第 1 章　初识 PHP

马上就要进入神秘的 PHP 殿堂学习了，对这个有些许陌生的名词，我们既新奇又憧憬，新奇于 PHP 到底为何物，憧憬于尽快能驾驭 PHP。从本章开始就揭开 PHP 的神秘面纱，共同去认识 PHP。

学习本章，可以获得以下知识点：

- PHP 的概念、前景和优势
- PHP 的历史
- PHP 同其他网络编程语言的异同
- PHP 常用代码开发工具

1.1　介绍 PHP

1.1.1　PHP 是什么

PHP 是 Hypertext Preprocessor（超文本预处理器）的递归缩写，根据 PHP 官方的定义：PHP 是一种广泛使用的开源的一般用途脚本语言，它特别适合于 Web 开发和嵌入到 HTML 中。该解释言简意赅，但还不能给我们一个 PHP 的直观印象，下面举个例子说明。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html4/loose.dtd">
<html>
      <head>
         <title>Example</title>
      </head>
      <body>
         <?php
             echo "Hi, I'm a PHP script!";
         ?>
      </body>
</html>
```

这是一个最简单的 PHP 网页例子，之所以称之为 PHP 网页，在于 HTML 里嵌入了“<?php”和“?>”之间的 PHP 代码。这段 PHP 代码的作用是输出字符“Hi, I'm a PHP script!”，用法是嵌入到 HTML 中并以“<?php”和“?>”标记出来。通过这个例子的讲解，大家应该对 PHP 有个直观的理解了吧。

1.1.2　PHP 的前景

选择一门语言学习，其前景如何是我们考虑的重要因素之一，毕竟 Web 语言数种，ASP、JSP、Perl 都可供选择，为什么要选择 PHP 来建设网站？PHP 有前景吗？

先看看下面的统计数据。据 NetCraft 统计，2002 年 PHP 就已经超越了 ASP、JSP、Perl 等语言的应用。在 TIOBE 语言流行程度统计中，PHP 稳固占据 5 种主流语言之一，2006 年超过了 41%的主要网站运行着 PHP（2100 万/5100 万）。新的资料表明，50%的 Web 2.0 网站采用 PHP 开发，PHP 已得到了包括 Sun、Adobe、Macromedia、Oracle、IBM、微软在内的主要厂商的认证和支持，在某种程度上说，PHP 语言和微软之.NET、Sun 帝国之 Java 正在三分天下。

再引述 PHPChina 的统计资料，中国的 PHP 应用在 2005 年后出现了明显的上升。Alexa 排名中国网站前 500 名中，有 262 家使用了 PHP 技术，占整体比例的 52.4%。Google 排名的 25 种行业网站的前 10 名网站中，采用 PHP 技术的网站增加到 192 家，占整体比例的 76.8%。

可见，有众多的网站投入 PHP 的怀抱，新型的 Web 2.0 网站对 PHP 如此垂青，这些数据，足以说明 PHP 非常有前景。

1.1.3 PHP 的优势

PHP 是一种很有前景的 Web 语言，因为其有着以下几方面的优势。

1．开放源代码

PHP 是开放的源代码，这就意味着可随意修改和扩展它的功能，还能够得到数百万的 PHP 程序员、数万个 PHP 开发团队的支持，可与他们一道分享心得，交流经验，这对程序员来说，是再美妙不过的事情了。

2．易于学习

PHP 的语法与 C、ASP、JSP 类似，对于熟悉上述语言之一的人来说，很短的时间就可以将 PHP 的核心语法掌握，如果又非常了解 HTML，就能立即把 PHP 代码融入网站的设计，使得站点呈现动态性和交互性。

3．数据库连接

PHP 的开发者们为了更适合 Web 编程，开发了许多外围的基库，这些库包含了更易用的层，这样就可以方便地利用 PHP 连接包括 Oracle、MS-Access、MySQL 在内的大部分数据库。

4．面向对象编程

基于 Web 的编程工作非常需要面向对象编程，PHP 能够满足这些需求，它面向对象，提供类和对象，支持构造器。

5．可扩展性

随着版本的更新，PHP 的功能在一步步扩展，同时由于它是开源项目，只要熟悉 PHP，自己完全能够对它的功能进行扩展。

提示：PHP 还有一些其他优势，这里不一一介绍，随着学习的深入，读者对 PHP 的强大功能会有一个全面认识。

1.1.4 PHP 的发展历史

PHP 最初由 Rasmus Lerdorf 于 1994 年发起，之后以 Personal Home Page Tools（PHP Tools）对外发布了第一个版本，在这个早期版本中，提供了访客留言本、访客计数器等简单的功能。

1995 年，PHP 第二个版本问世，定名为 PHP/FI（Form Interpreter）。PHP/FI 是一个专为个人主页/表单提供解释程序的程序，它有着 Perl 样式的变量，能自动解释表单变量，加入了对 MySQL 的支持，并可以嵌入到 HTML 中，已经包含了今天 PHP 的一些基本功能。这期间 Andi Gutmans 和 Zeev Suraski 加入到程序开发中，他们三个就是 PHP3 的创始人。

1998 年 6 月，第三版 PHP 正式发布。第三版 PHP 被定名为 PHP3，它类似于当今 PHP 语法结构的第一个版本，它最强大的功能就是其可扩展性，能够给最终用户提供数据库、协议和 API 等基础结构。PHP 的可扩展性还吸引了大量的开发人员加入并提交新的模块，后来的事实证明，这是 PHP 3.0 取得巨大成功的关键。

2000 年 5 月，PHP 官方发布了 PHP 4.0 正式版本。PHP 4.0 除了更高的性能以外，还包含了其他一些功能，比如：支持更多的 Web 服务器，输出缓存，更安全的处理用户输入的方法等。

2004 年 7 月，PHP5 问世。PHP5 算是一个里程碑式的版本，它采用 Zend II 引擎，完备了对象模型，改进了语法设计，终使得 PHP 成为一个设计完备、具有面向对象能力的脚本语言。目前 PHP 已经发布到 PHP 5.3 alpha1 版本。

1.2 PHP 与 ASP、JSP 的对比

PHP、ASP、JSP 是三种常用的网络开发语言，它们之间有着各自的特点，下面对这三种语言进行简单介绍和比较。

1.2.1 语言介绍

ASP（Active Server Pages）是微软发布的脚本语言，利用它可以执行动态的 Web 服务应用程序。ASP 的语法与 Visual Basic 非常类似，学过 Visual Basic 的人可以很快上手，它也是这几种脚本语言中较简单易学的开发语言。

JSP（Java Server Pages）是 Sun 公司推出的一种动态网页技术。JSP 技术是以 Java 语言作为脚本语言的，熟悉 Java 语言的人可以很快上手。JSP 本身虽然也是脚本语言，但是却和 PHP、ASP 有着本质的区别。PHP 和 ASP 都由语言引擎解释执行程序代码，而 JSP 代码却被编译成 Servlet 并由 Java 虚拟机执行，这种编译操作仅在对 JSP 页面的第一次请求时发生。

PHP 的语法和 Perl 很相似，但是 PHP 所包含的函数却远远多于 Perl。PHP 语法简单，非常易学易用，很利于快速开发各种功能不同的定制网站。由于其免费、开源，它往往和同样免费、开源的 Apache 和 MySQL 搭配使用，可以非常快速地搭建一套功能丰富的动态网站。

提示：上述几种语言都可作为制作网页的语言，另外还有 Java 等程序语言。

1.2.2 性能对比

上面简单介绍了三种语言，下面列出其语言特点之间的对比，以供参考。

其语言特点对比如表 1-1 所示。

表 1-1 性能对比

语言 / 性能指标	PHP	ASP	JSP
操作系统	均可	Win32	均可
Web 服务器	多种	IIS	多种
执行效率	快	快	极快
稳定性	佳	中等	佳
学习门槛	低	低	高
函数支持	多	较多	多
系统安全	好	差	好
升级速度	快	慢	中等
网页结合	好	好	差
开发时间	短	短	较长

ASP、PHP、JSP 三者都是面向 Web 服务器的技术，客户端浏览器不需要任何附加的软件支持。三者都能嵌入到 HTML 代码中，由语言引擎解释执行程序代码；HTML 代码主要负责描述信息的显示样式，而程序代码则用来描述处理逻辑。唯一不同的是 JSP 代码被编译成 Servlet，并由 Java 虚拟机解释执行。

1.3 PHP 常用开发工具

编写 PHP 程序，借助一些工具软件，会使编程工作事半功倍。PHP 开发工具有很多，常用的代码开发工具如 Zend Studio、PHP Edit，网页编辑器如 Frontpage、Dreamweaver，文本编辑器如 UltraEdit、EditPlus，甚至 Windows 自带的记事本，都可以书写源代码。

1.3.1 PHP 代码开发工具

Zend Studio 是强大的 PHP 应用程序开发工具，它包括编辑、调试等客户端及服务器组件，它的启动界面如图 1-1 所示。

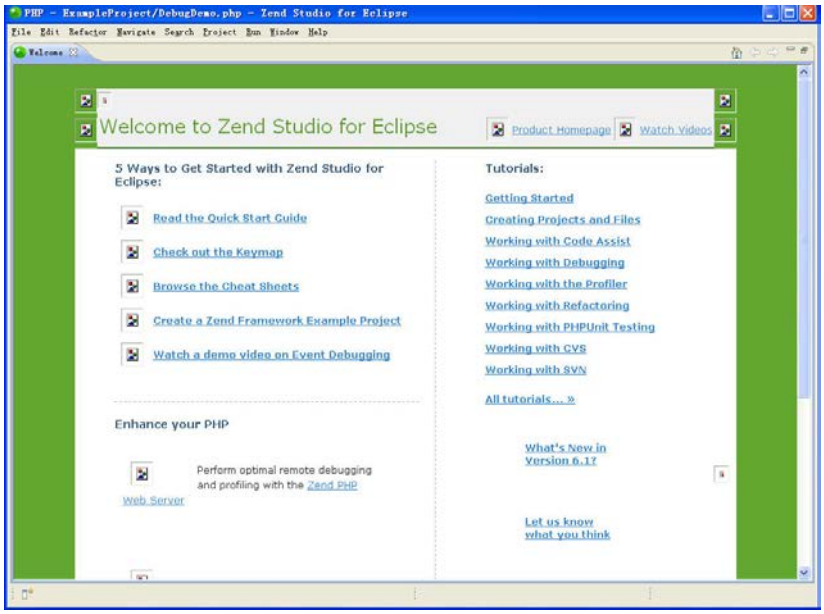

图 1-1　Zend Studio 启动界面

Zend Studio 的编辑功能非常适合用户编写 PHP 代码，它提供项目和文件管理器，能在 Zend Studio 中管理文件和文件夹；它提供代码补全功能，能实现 PHP、HTML、变量、关键字等在内的代码补全功能；它提供代码模板，能帮助用户快速准确地编写代码，对于将要编写的代码片段，模板可以很快地插入一个构架；它支持语法高亮，能将代码自动高亮显示，以突出不同的语法元素；它支持代码缩进，能控制 PHP 代码的格式。

Zend Studio 的调试功能也很方便，它通过控制 PHP 应用程序的运行过程，能够给出错误报告、变量值和输出值等重要调试信息；它提供 PHP 在线手册功能，能够通过 Zend Studio 浏览器窗口给出所查询的 PHP 函数的详细参考信息。Zend Studio 还提供其他辅助功能，诸如定位导航功能，能够提供多种方法来定位到想跳转到的文本和代码；自定义快捷键功能，能够根据通常的按键习惯自定义键盘快捷方式。Zend Studio 工程界面有多个窗口，如图 1-2 所示，可以看到文件管理器、检查器、编辑器、浏览器、调试输出等窗口，借助它我们就可以方便地编

写和调试 PHP 程序了。

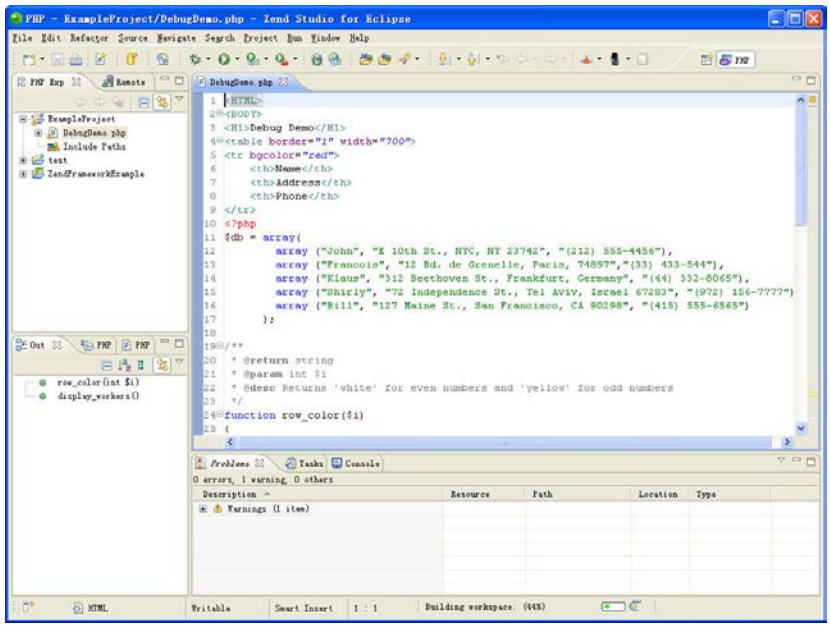

图 1-2　Zend Studio 工程界面

提示：Zend Studio 是一款较适合 PHP 网络开发的工具，其可以实现编译、检查等多种功能。

PHPEdit 是一款 Windows 平台下的 PHP 脚本开发工具，该软件为快速、便捷地开发 PHP 提供了很大方便。其功能包括：代码提示、集成 PHP 调试工具、帮助生成器、自定义快捷方式、快速标记等，采用这种开发工具可以方便地进行 PHP 代码设计，其工作界面如图 1-3 所示。

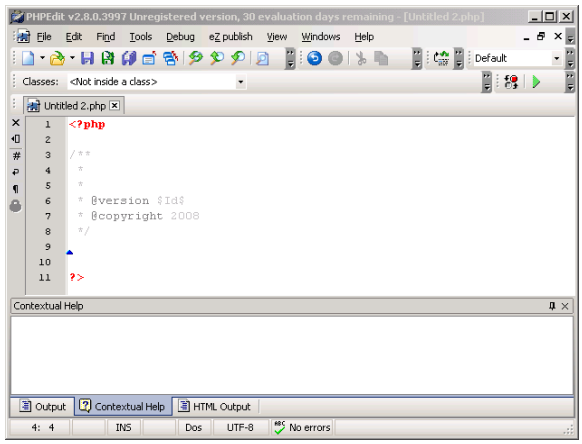

图 1-3　PHPEdit 工作界面

1.3.2　网页设计工具

FrontPage 是微软公司推出的一款制作网页的软件，简单易学，容易上手，使用者可以很容易地管理网站的内容、超链接、页面和发布。FrontPage 与 Microsoft Office 各软件无缝连接，

编辑时有三种模式：“普通”模式，用来编辑网页内容；“HTML”模式，用来观看 FrontPage 自动生成的 HTML 代码；“预览”模式，演示网页效果。FrontPage 的工作界面如图 1-4 所示。

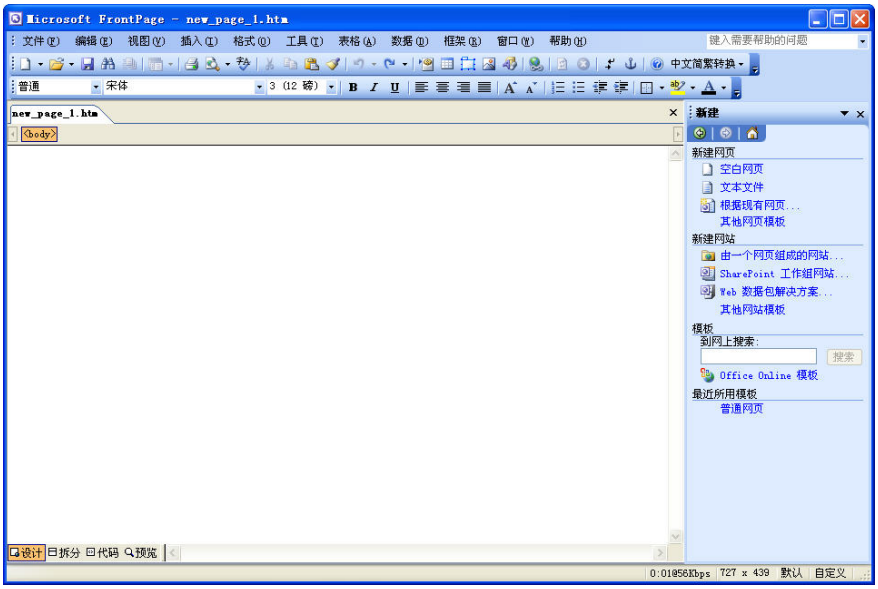

图 1-4　FrontPage 工作界面

FrontPage 对于初学者具有很大的优势：所见即所得，即使不懂 HTML 语言，也能制作出具有专业效果的网页；使用者很容易在网页中加入 ActiveX Controls、JavaScript、VBScript 等组件。但 FrontPage 也有自身的缺陷：对于新技术缺乏支持；生成代码冗余；对数据库支持不足。

Dreamweaver 是另一款网页设计软件，它有着非常友好的代码和设计模式界面，能够为网页设计者提供非常方便的辅助，其界面如图 1-5 所示。

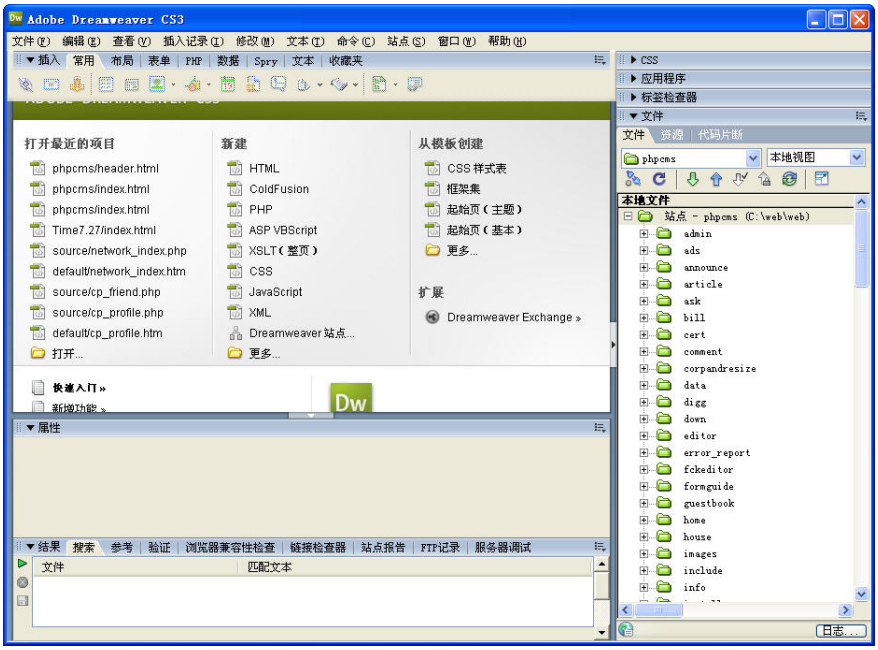

图 1-5　Dreamweaver 启动界面

Dreamweaver 具有很好的直观性与高效性，它的标准工作界面，如图 1-6 所示，它包括：标题栏、菜单栏、插入面板组、文档工具栏、标准工具栏、文档窗口、状态栏、属性面板和浮动面板组。

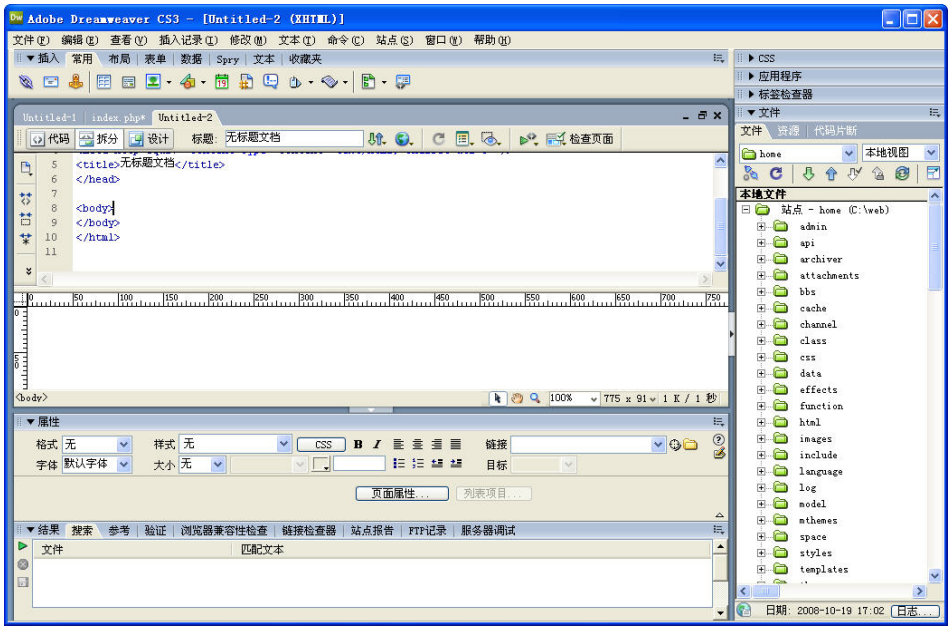

图 1-6　Dreamweaver 工作界面

提示：Dreamweaver 是一款比较方便的网页设计工具，其能够实现提示编写代码，适合于对网络语言不十分熟悉的用户。

1.3.3　文本编辑工具

UltraEdit 是一款功能强大的文本编辑器，对 C 语言、ASP、PHP、CSS 等都有很好的语法支持，能够支持宏，有列模式，对处理大容量文本、二进制格式文件很有优势。

UltraEdit 的界面如图 1-7 所示，上面是标题栏、菜单、工具栏，下部是驱动器文件列表和文本编辑区，显示类似资源管理器的文件目录和所编辑的文件内容。UltraEdit 具有十六进制编辑模式，可编辑二进制代码文件；支持同时编辑多个文件；支持多文件的查找/替换；支持拼写检查功能；支持多种文件格式，包括*.txt、*.doc、*.bat、*.ini、*.c、*.cpp、*.h、*.hpp 和*.java 等；支持宏功能，能提供宏记录、宏编辑、宏调用等多种宏处理，给文本编辑中经常重复的操作带来方便。

介绍了 UltraEdit，有必要介绍 UltraEdit 的姊妹软件 UltraCompare，它是一款文件内容比较工具软件，能进行文本模式、文件夹模式及二进制模式的比较，并且可以对比较的文件进行合并、同步等操作，支持撤销操作，它拥有书签与收藏夹功能，可以设置过滤。UltraCompare 的界面如图 1-8 所示。

EditPlus 也是一款功能强大的文本编辑器，它能支持文本、HTML、程序源代码编辑，支持 HTML、CSS、PHP、ASP、Perl、C/C++、Java、JavaScript 和 VBScript 的语法高亮显示，支持以正则表达式的方式进行查找与替换，可以在某一目录的所有文件中，进行字符串的查找。EditPlus 的界面如图 1-9 所示。

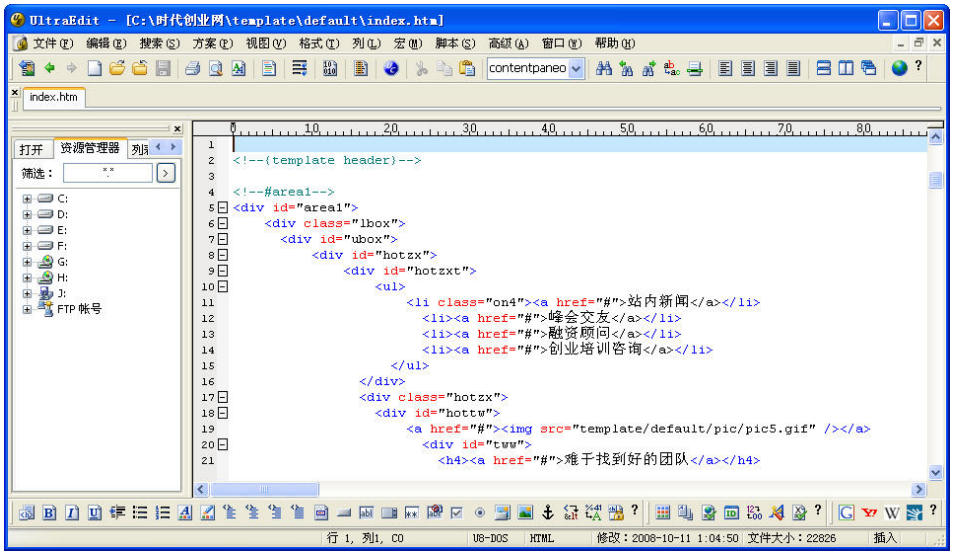

图 1-7　UltraEdit 编辑器

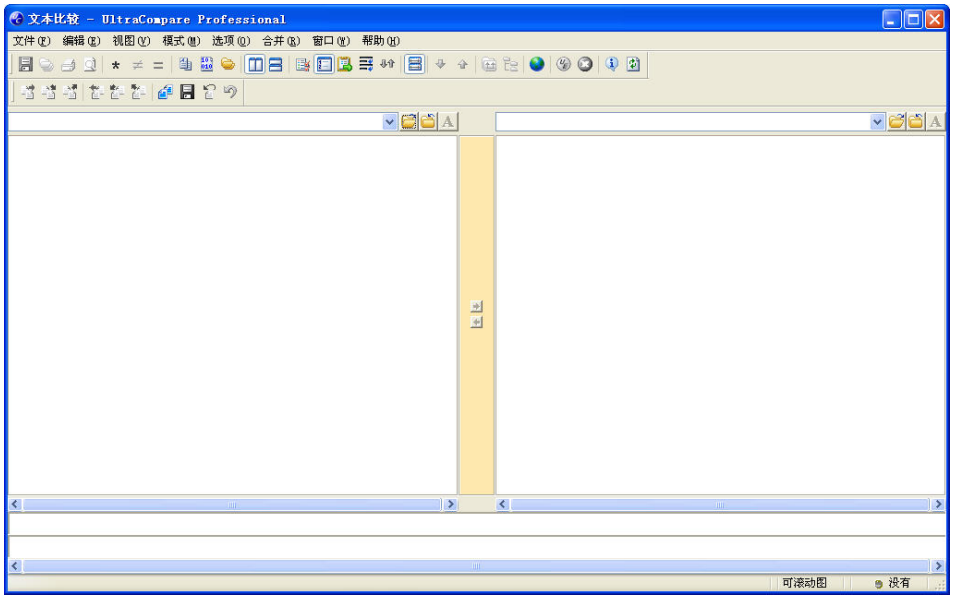

图 1-8　UltraCompare

记事本是 Windows 自带的小软件，通常在处理简单文本的时候会用到它。记事本占用内存极少，运行速度很快，但功能较简单，如图 1-10 所示。对于我们学习简单的 PHP 程序，记事本就足够了，本书的例子都是在记事本下输入的。

提示：在使用记事本编好程序存盘的时候，在文件名一栏手动输入文件名字，后缀为.php，保存类型应选择“所有文件”，否则系统会自动为文件加上后缀名.txt，使之成为文本文件，无法直接实现 PHP 的功能。

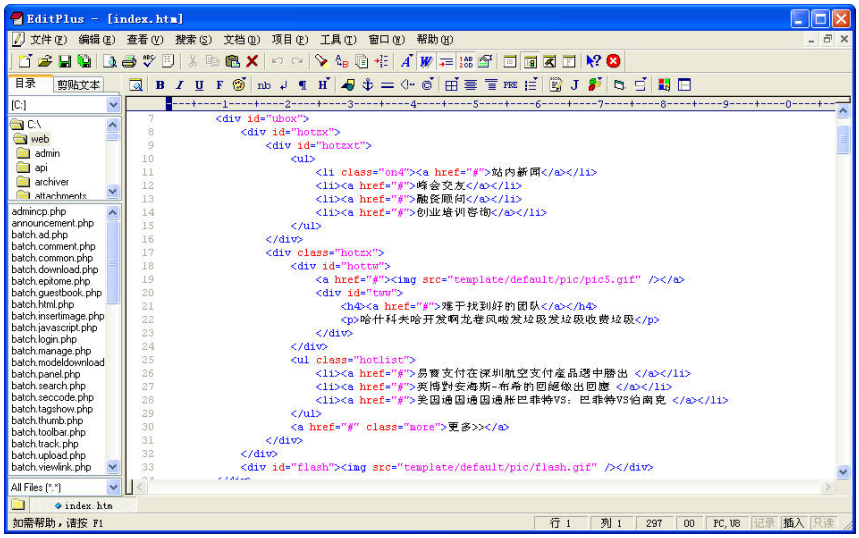

图 1-9　EditPlus 编辑器

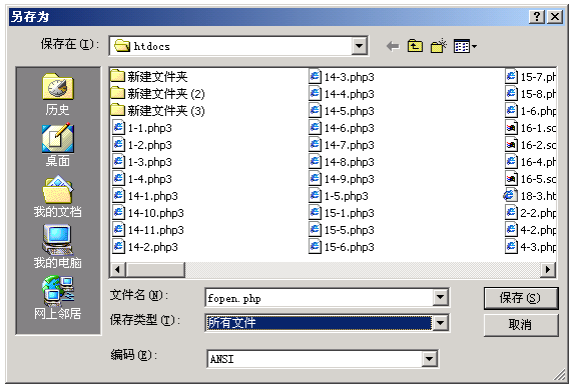

图 1-10　记事本

1.4　综合练习

1．利用 PHP 语言输出“21 天基础教程”。

提示：PHP 的语言特点如 1.1 节所述，输出“21 天基础教程”，只需将 echo 命令后的代码替换掉，其程如下：

```
<html>
   <head>
      <title>Example</title>
   </head>
   <body>
      <?php
          echo "21 天基础教程";
      ?>
   </body>
</html>
```

2．PHP 的优势是能同数据库结合，编写一个连接数据库的代码。

提示：PHP 连接数据库，首先要经过三个步骤：（1）设置初始变量；（2）连接数据库；（3）判断连接情况。其程序代码如下：

```
<?php
        /* 步骤一：设置初始变量 */
    $host="localhost";
    $user="root";
    $password="123456";
        /* 步骤二：连接 MySQL 服务器 */
    $conn=mysql_connect($host,$user,$password);
        //步骤三：判断连接结果
    if (!$conn){
        die ("连接数据库失败。".mysql_error( ));
        }
    else{
    echo  "MySQL 服务器: $host <br> 用户名称：$user  <br>";
    echo  "成功连接数据库。";
    }
?>
```

1.5 小结

本章介绍了 PHP 的概念、前景、优势和发展的历史，并同其他 Web 编程语言做了对比，使我们对 PHP 有一个全方位的认识。最后简单介绍了 PHP 常用的开发工具，读者可以尝试着使用这些工具软件，看哪一个适合自己，借助它来帮助我们更方便地学习 PHP。

1.6 习题

一、填空题

1．PHP 可以嵌入到_____语言中，以_____和_____标记出来。

2．PHP5 算是一个里程碑式的版本，其采用_____，完备了对象模型，改进了语法设计，使得 PHP 成为一个设计完备、具有面向对象能力的脚本语言。

3．第二版 PHP 问世，加入了对_____数据库的支持。

4．JSP 是 Sun 公司推出的一种_____技术。

5．对 PHP 代码开发工具，主要有_____、_____。

6．PHP 是一种很有前景的 Web 语言，其稳定性_____，安全性_____，学习门槛_____。

7．常用的两种网页设计工具是_____和_____。

8．Windows 自带的文本编辑工具是_____。

二、选择题

1．能嵌入到 HTML 代码中的语言是（　　）。

A．PHP　　B．ASP　　C．JSP　　D．都可以

2．下列语言哪种升级较快（　　）。

A．PHP　　B．ASP　　C．JSP　　D．都一样

3．微软发布的脚本语言是（　　）。

A．PHP　　B．ASP　　C．JSP　　D．都一样

4．PHP 的特点是（　　）。

A．不支持数据库　　B．不支持面向对象操作

C．不可扩展　　D．易于学习

三、简答题

1．什么是 PHP？PHP 的前景及优势如何？

2．PHP 常用开发工具有哪些？

四、编程题

1．上网了解 PHP 的语言特点及其发展情况。

2．查找一些用 PHP 构建的网站。

第 2 章　配置 Web 服务器

学习 PHP 语言，需要配置一个能够运行、解析 PHP 语言的 Web 服务器，这样才能开发调试 PHP 程序。本章针对配置 Web 服务器，分别介绍 Apache、PHP 和 MySQL 数据库在 Windows 平台下的安装和配置。学习本章可以掌握以下知识点：

- Apache 的安装与配置；
- PHP 的安装与配置；
- MySQL 数据库的安装；
- Web 服务器的测试。

2.1　准备工作

要想配置一台支持 PHP 语言的 Web 服务器，需要从硬件和软件两方面进行需求分析。下面就从这两方面来介绍 Web 服务器的配置。

2.1.1　选择硬件

配置一台 Web 服务器，从硬件角度来讲可分为两条路线，一个是专业级，另一个是业余调试级。简单地说，专业级就是从构建一台真正的 Web 服务器出发，按服务器标准配置专业级硬件。即意味着有支持 24 小时/365 天的服务器级 CPU、主板和散热机箱，以及 RAID 的冗余备份系统，软件和硬件防火墙，标准空调托管环境等标配。业余调试级 Web 服务器以目前主流的个人电脑配置就可满足其硬件要求，即至少 256MB 内存、奔腾级 CPU、10Mb/s 网卡。

对于 PHP 初学者来说，配置一个专业级服务器来学习 PHP 有点大炮打蚊子的意味，用我们手头的电脑配置一个支持 PHP 的 Web 服务器才是明智之举。下面就来介绍一下在个人电脑上如何配置一个 Web 服务器。

2.1.2 选择操作系统

要配置 Web 服务器，确定了硬件需求后，还需要为它选择操作系统，即要确定该服务器是运行在 Windows 或 Linux 下。若要使该服务器运行在 Windows 下，需要硬件平台上装有 Windows 操作系统，这就意味着花费一笔钱向 Micosoft 购买该软件；若要使该服务器运行在 Linux 下，则不要这笔额外的花费，因为 Linux 是开源免费的。由于国内大多数用户（尤其 PHP 初学者）都熟悉 Windows 操作系统，这里就把我们这个 Web 服务器定位为运行在 Windows 操作系统下的服务器，所有的软件都是按运行在 Windows 操作系统下准备和安装的。

对于 Windows 操作系统大家都已很熟悉了，为构建 Web 服务器系统，我们可以选择在个人电脑上安装 Windows Server 或 Windows XP 系列操作系统。Micosoft 的 Windows Server 服务器操作系统现在有主流的 Windows Server 2003 和 Windows Server 2008，它们专为中小企业服务器专业设计，性能和安全性很好。Windows XP 系统在国内有最多的用户，广大用户也最熟悉其操作，在这样的系统下也是能够构建 Web 服务器的，虽然它没有在 Windows Server 下性能卓著，但对于 PHP 调试已足够了。基于大多数 PHP 初学者的习惯问题，该服务器就定位运行在 Windows 系统下。

警告：如果是 Linux 系统用户按本章安装方法安装，可能会引起错误。如果你是其他系统用户，请查阅相关资料。

2.1.3 选择服务器软件

在 Windows XP 操作系统下，原则上只要安装服务器软件就建立了 Web 服务器。也可以选择 XP 自带的 IIS 服务器，或选择第三方服务器软件。IIS 在 Windows XP 安装盘里自带，默认没有安装，如需使用，只要添加即可。IIS 默认支持 ASP 语言，对于 PHP 是不能解析的，需要另外配置才可以。由于 IIS 不是支持 PHP 语言的主流，它对 PHP 支持的高级功能大打折扣，例如 PHP 压缩解析、重写、多域名支持等都不如 PHP 主流服务器软件做起来得心应手，所以在这里还是舍弃了以 IIS 作为服务器软件的做法，而是转向 PHP 的主流服务器软件——Apache。

Apache 为开放源代码软件，允许任何人对其修改、扩充、更新，使其可以运行在几乎所有广泛使用的计算机平台上。在所有的 Web 服务器软件中，Apache 已占据绝对优势，远远领先排名第二的 Microsoft IIS，是目前世界上使用最为广泛的 Web 服务器。

2.1.4 选择 PHP

PHP 是免费开源软件，可以到 PHP 官方站点 www.php.net 去下载使用。PHP 版本很多，鉴于 PHP 5 性能稳定，功能齐全，又向下兼容，这里确定选择 PHP 5 系列来安装。登录 PHP 官方网站，单击导航链接的“downloads”打开 PHP 的下载页面，该页面列出了最新的 PHP 安装资源包，单击“Windows Binaries”中的“PHP 5.2.2 zip package”，下载完成之后，得到一个名为“php-5.2.2-Win32.zip”的压缩包，这就是需要的 PHP 安装资源包。在这个资源包中，包括了安装和配置 PHP 服务器的一切文件，以及大量 PHP 扩展函数库。通过此压缩包的名字能看出其版本号为 5.2.2，适用平台为 Win32，即 Windows 系列。

至此，就完成了 PHP 软件资源包的准备工作。

注意：很多网站提供的 PHP 安装资源包并不规范和全面，有些 PHP 安装软件是由第三方开发的，并没有经过 PHP 官方的认可和授权。有的安装包为了缩小体积还

去掉了大部分扩展函数库，给以后的使用带来不便。因此，建议直接去 PHP 官方网站下载 PHP 安装资源包，并用手工方法安装和配置 PHP 服务器，只有这样，才能保证所使用的 PHP 完整、全面和权威。

2.1.5 选择数据库

对于开发网站来说，使用数据库是开发专业站点必不可少的，但是数据库类别林林总总，Access、SQL Server、Oracle、MySQL 各有优势，PHP 语言也都支持这些数据库，选择哪个呢？考虑到 Access、SQL Server 和 Oracle 都不是开源免费软件，需要购买才能使用，而 MySQL 是开源软件，选择 MySQL 是最节省成本的做法，而且 Apache、PHP 和 MySQL 是绝配，它们的组合有着最佳的性能和最广大的用户支持群。最终看来，还是应该选择 MySQL 作为我们的 Web 服务器数据库。

MySQL 是一个真正的多用户、多线程 SQL（结构化查询语言）数据库服务器，它以客户机/服务器的结构实现。MySQL 有如下优势。

- 速度快。开发者声称 MySQL 数据库可能是目前能得到的最快的数据库。可访问 MySQL 官方站点上的性能比较页 http://www.mysql.com/benchmark.html，上面有 Oracle 与 MySQL、PostgreSQL 与 MSQL 实测比较，可测试它的速度性能。
- 连接性和安全性。MySQL 是完全网络化的，其数据库可在因特网上访问，因此，可以和任何地方的任何人共享数据库，而且 MySQL 还能进行访问控制，能够控制特定用户不允许其访问数据。
- 可移植性。MySQL 可运行在各种版本的 UNIX 系统及其他非 UNIX（如 Windows 和 OS/2）系统上，从家用 PC 到高级服务器都可运行 MySQL。
- 支持 SQL 语言。MySQL 支持这种现代数据库系统都选用的语言。
- 容易使用。MySQL 是一个高性能且相对简单的数据库，易于操作。
- 成本优势。MySQL 对多数个人用户来说是免费的。

目前，MySQL 已经发布到 5.0 系列，可以到 MySQL 官方网站 http://www.mysql.com 下载得到，这里下载的是 mysql-5.0.22-win32.exe，从文件名分析它的版本是 5.0.22，适用于 Windows 平台。

2.2 Apache 的安装

前面完成了服务器配置的准备工作，从本节开始将安装配置服务器，下面讲解 Apache 的安装与配置。

2.2.1 安装 Apache

Apache 在 Windows 平台上的安装非常简单，只需要启动安装程序，按照提示逐步进行即可。

注意：在安装 Apache 之前，必须保证电脑里没有安装 IIS 或其他服务器软件，如果已经安装了，建议将其卸载，因为可能会由于端口冲突而导致 Apache 无法启动，当然也可以设置使各个服务器软件工作在不同端口，但是在绝大多数情况下只需要一个服务器软件就足够了。

① 打开下载的 Apache 安装文件，双击启动安装可执行程序，首先会看到一个欢迎画面，如图 2-1 所示。

② 单击“Next”按钮，继续安装。进入版权声明界面，出现 Apache 许可协议，阅读完毕许可协议之后选择“I accept the terms in the license agreement”单选按钮，表示接受许可协议中的条款，如图 2-2 所示。

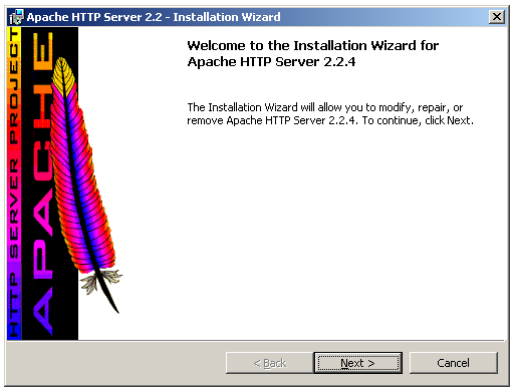

图 2-1　欢迎界面

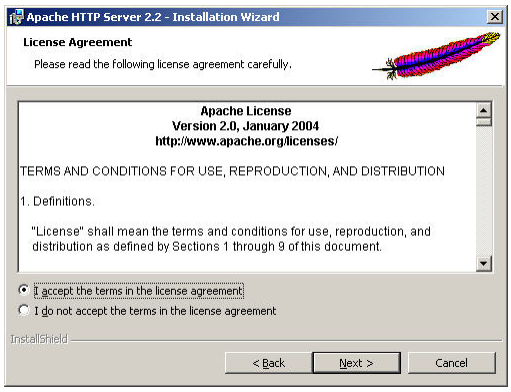

图 2-2　版权声明界面

警告：用户必须同意画面所列版权声明，如不同意将停止安装。

③ 单击“Next”按钮，出现如图 2-3 所示的服务器注意事项。

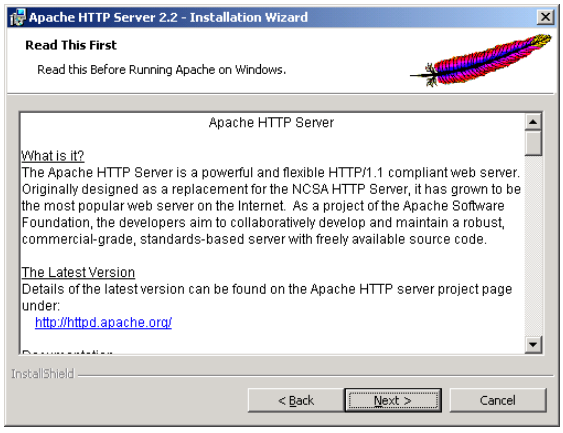

图 2-3　服务器注意事项

④ 阅读完毕后，单击“Next”按钮，出现服务器信息设置界面，如图 2-4 所示。

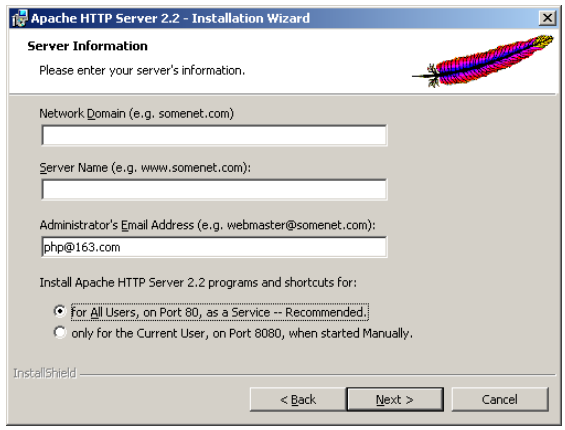

图 2-4　服务器信息设置界面

此处要求设置一些服务器基本信息，分别是网站域名、服务器名、管理员邮箱，以及 Apache 的工作方式。如果只是在自己的机器上使用 Apache，那么前三个选项可以保持空白，不需要设置。第四个选项有两种选择，建议选择第一项，即“针对所有用户，工作在 80 端口，安装为服务”，“安装为服务”的意思是将 Apache 安装为 Windows 的一个服务，当机器启动时自动启动 Apache。

⑤ 设置完成之后，单击“Next”按钮进入下一步，这时出现安装类型窗口，在这里有 Typical 和 Custom 即“典型安装”和“用户自定义安装”两种选择，如图 2-5 所示。建议不太熟悉 Apache 的初学者使用“典型安装”。

⑥ 单击“Next”按钮，出现 Apache 安装位置选择窗口。Apache 默认被安装到 C:\Program Files\Apache Software Foundation\Apache2.2\目录下。如果希望安装在其他位置，可以单击“Change”按钮来选择另外一个位置，这里采用默认位置，如图 2-6 所示。

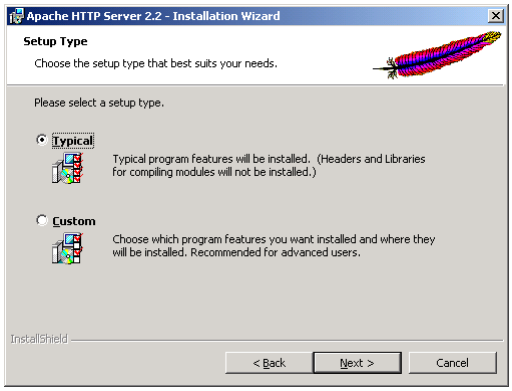

图 2-5　安装类型窗口

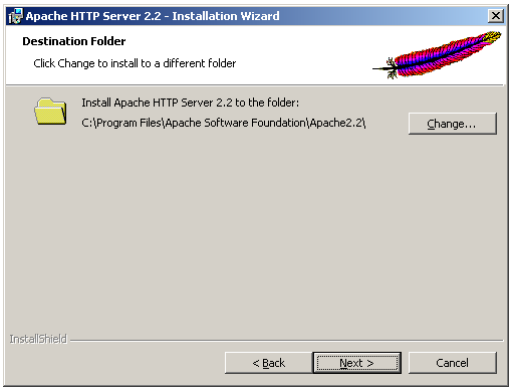

图 2-6　安装位置选择窗口

⑦ 单击“Next”按钮，这时出现“安装准备已就绪”窗口，如果不需要对前面的步骤做修改，就可以直接单击 Install 按钮开始安装 Apache。安装开始后会出现安装进度条，如图 2-7 所示。

⑧ 全部安装完成后，则会出现安装成功的提示窗口，单击“Finish”按钮结束安装程序，如图 2-8 所示。至此，Apache 的安装就完成了。

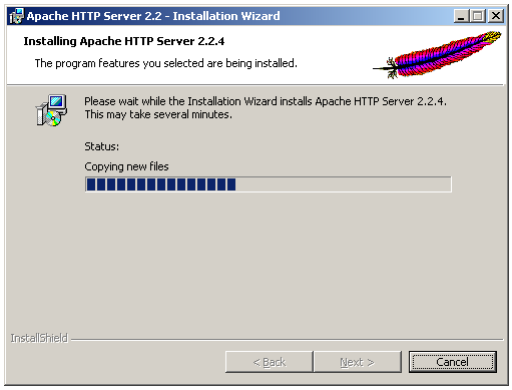

图 2-7　安装过程

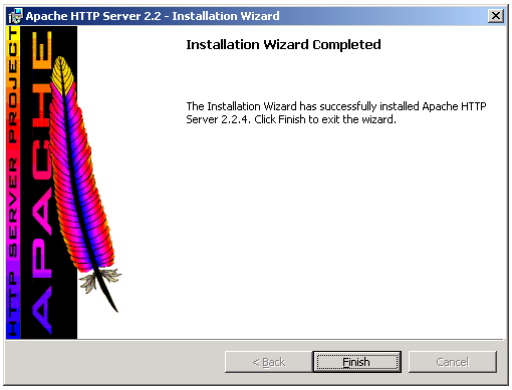

图 2-8　完成安装

⑨ 验证 Apache 是否安装成功，打开浏览器，在地址栏中输入 http://lcalhost/或者 http://127.0.0.1，这时如果看到如图 2-9 所示的页面，说明 Apache 已经成功安装并开始工作了。

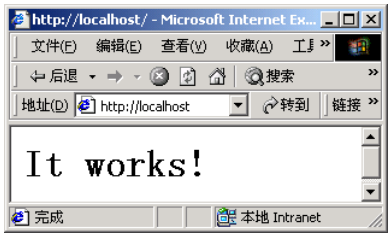

图 2-9　测试 Apache

2.2.2　配置 Apache

此时网站已经能够解析静态网页了，但是所有的程序必须放在 Apache 的 htdocs 文件夹下才能被解析。如果用户想自己建立网站根目录，则需要对 Apache 的 http.conf 文件进行如下设置。

查找 DocumentRoot，下面有这样一行：

```
DocumentRoot"C:/Program Files/Apache Group/Apache2/htdocs"
```

这就是网站的默认根目录，可以修改为希望设置的目录。查找“This should be changed to whatever you set DocumentRoot to”。在它下面有这样一行：

```
<Directory "C:/Program Files/Apache Group/Apache2/htdocs">
```

把 C:/Program Files/Apache Group/Apache2/htdocs 改成要设置的目录，如 C:\web。

这样，网站的根目录就变为 C:\web 了。

注意：第二项必须改，否则会出现 403 错误。

2.3 PHP 的安装与配置

前面已经准备好了 PHP 资源包，接下来进行 PHP 的安装与配置。

2.3.1 安装 PHP

将之前下载的 PHP 安装资源包 php-5.2.2-Win32.ip 解压缩，会得到一个文件夹，这里面放着运行 PHP 所需要的所有文件，解压后的文件如图 2-10 所示。将这些文件复制到 PHP 的安装目录，PHP 的安装目录可以根据自己的需要随意设置，这里放置到 C:\php5。

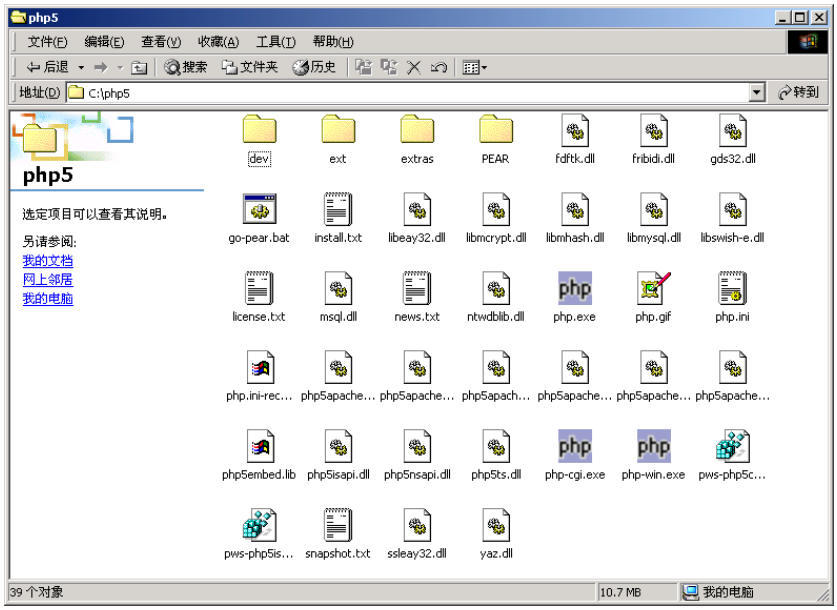

图 2-10　安装后 PHP 的文件

之后进行如下操作：将 php5 目录下的 php.ini-dist 或 php.ini.recommended 文件，重命名为 php.ini 并复制到系统盘的 Windows 目录下（以 C:\windows 为例）；将 php 目录下的 php5ts.dll、libmysql.dll 复制到 C:\windows\system32 下；再将 php5\ext 目录下的 php_gd2.dll、php_mysql.dll、php_mbstring.dll 三个文件复制到 C:\windows\system32 下。

至此，PHP 的安装就完成了。

2.3.2 配置 PHP

安装了 PHP，并不是说明 PHP 就能运行了，而是需要经过配置 PHP 才能够起作用。在 Windows 目录下找到 php.ini 文件，然后用记事本打开，查找以下字符串：extension=php_gd2.dll、extension=php_mysql.dll 和 extension=php_mbstring.dll，将这三项前面的“;”去掉，使 PHP 支持处理图像、MySQL 函数库和宽字符。

如果想使 PHP 支持其他功能，只要在 php.ini 里进行相应的配置就行了。例如要使 PHP 支持以“<?”短标记作为 PHP 代码开始的标记，只需找到 short_open_tag 参数，将其值设为 on，如图 2-11 所示。

如果不允许 PHP 自动将外部提交的数据注册为全局变量，只需找到 register_globals 参数，将它设置为 off，如图 2-12 所示。

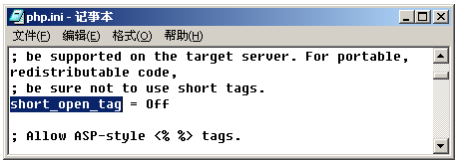

图 2-11 设置 short_open_tag 参数

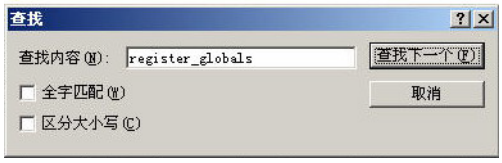

图 2-12 设置 register_globals 参数

如果需要 PHP 控制文件上传的信息，只需找到 file_uploads、upload_tmp_dir 和 upload_max_filesize 三个参数进行设置，如图 2-13 所示。这三个参数分别表示是否允许文件上传（on/off）、上传文件的暂存路径、上传文件的最大字节数。可以根据需要来修改和填写，也可以使用默认值。

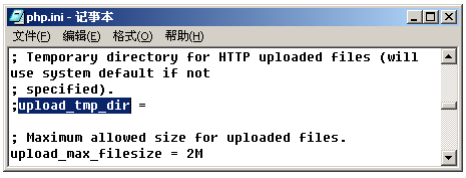

图 2-13 设置上传文件参数

以上介绍了 php.ini 文件的配置，这些设置对调试 PHP 程序来讲已经足够了。

提示：upload_tmp_dir 最好填写，以保证将来可以顺利地用 PHP 实现上传文件功能。

2.4 MySQL 数据库的安装与测试

前面已经准备好了 MySQL 安装软件，下面安装 MySQL 数据库。

2.4.1 安装 MySQL

① 双击执行 mysql-5.0.22-Win32.exe 程序，出现如图 2-14 所示的欢迎界面。

② 单击“Next”按钮，继续安装，出现安装选择窗口。它有三个选项可供选择，“Typical”（典型安装）、Complete（完全安装）和 Custom（自定义安装），建议初学者选择“典型安装”，如图 2-15 所示。

③ 单击“Next”按钮，出现的对话框显示安装类型和安装的路径，如图 2-16 所示。默认的安装路径为“C:\Program Files\MySQL\MySQL Server 5.0\”。

④ 单击“Install”按钮，出现安装进度条。安装进度完成后，出现 MySQL 注册窗口，提示创建一个 MySQL 通行证或者登录 MySQL 通行证，如图 2-17 所示。根据需要选择增加一个免费的 MySQL 账号或跳过注册账号。

图 2-14　欢迎界面

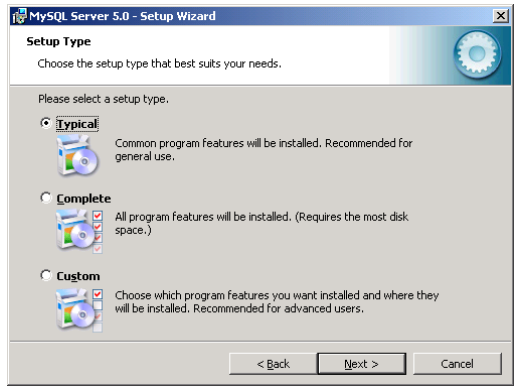

图 2-15　安装选择窗口

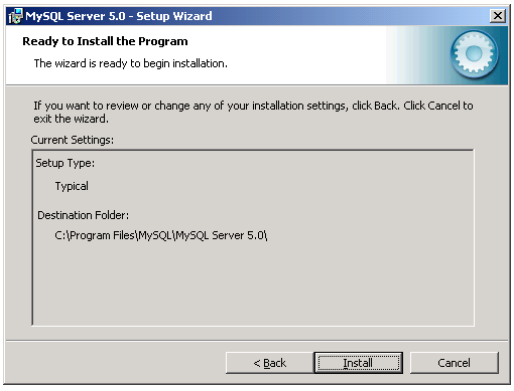

图 2-16　准备安装界面

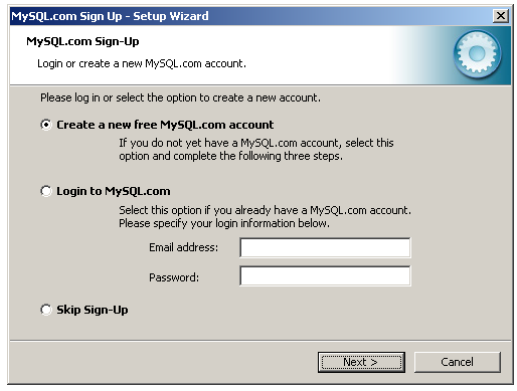

图 2-17　MySQL 注册窗口

⑤ 选择下方的“Skip Sign-UP”单选按钮，单击“Next”按钮，选择跳过。至此，MySQL 5.0 的安装已经完成，勾选“Configure the MySQL Server now”项以便随后进行 MySQL Server 的配置，然后单击下方的“Finish”按钮完成安装过程，如图 2-18 所示。随后即会自动进入 MySQL Server 配置过程，如图 2-19 所示是其配置过程的欢迎界面。

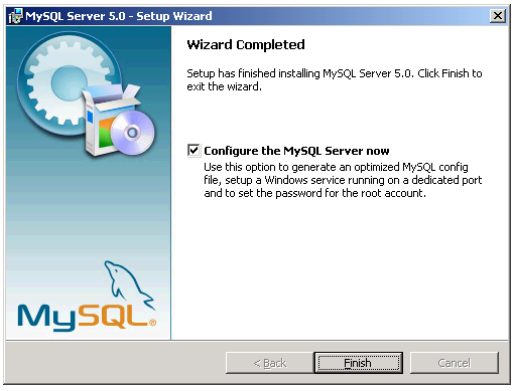

图 2-18　完成安装

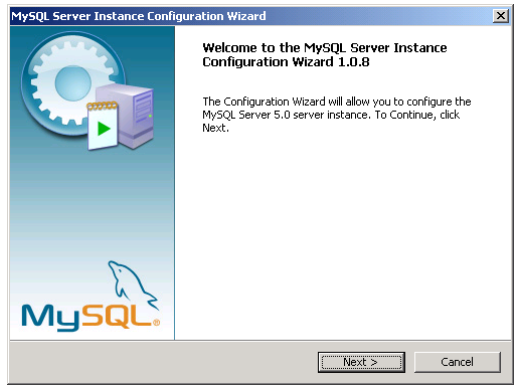

图 2-19　欢迎界面

提示：如果注册一个免费用户，需要连接网络，但也完全是免费安装。

⑥ 单击“Next”按钮，显示界面如图 2-20 所示，选择“Standard Configuration”选项。

⑦ 单击“Next”按钮，直至进入配置路径界面，如图 2-21 所示，按该图示进行配置，然后单击“Next”按钮。

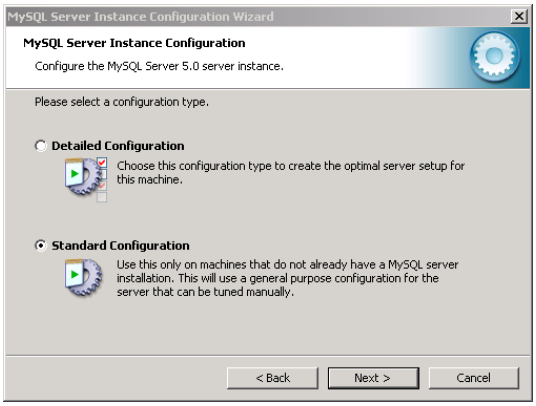

图 2-20　配置类型界面

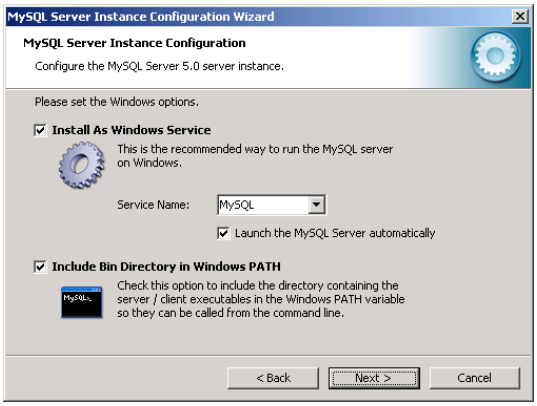

图 2-21　配置路径

⑧ 进入安全设置界面，设置“root”用户口令，输入要设置的密码（这里输入的密码为“123456”），单击“Next”按钮，如图 2-22 所示。

图 2-22　设置密码

⑨ 之后出现如图 2-23 所示的界面，最后单击下方的“Execute”按钮执行配置程序，运行结束即完成了 MySQL Server 的配置，如图 2-24 所示。

注意：如果机器上装有防火墙，在运行到第三项“Start service”时可能会弹出网络访问请求，此时应予放行。

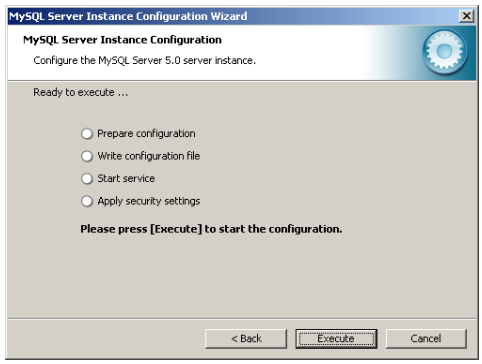

图 2-23　设置界面

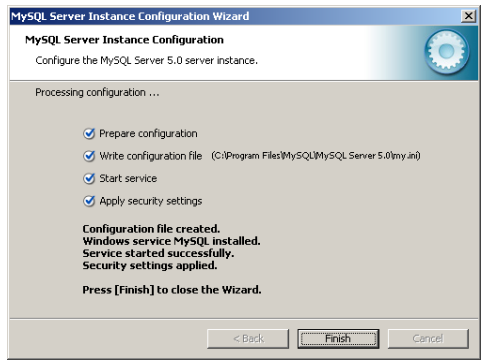

图 2-24　MySQL Server 配置完成

2.4.2　测试 MySQL

MySQL 已经安装好了，需要测试它是否正常工作。具体做法是：选择“开始”|“程序”|“MySQL”|“MySQL Server 5.0”|“MySQL Command Line Client”菜单项，若出现需要输入密码的对话框，说明 MySQL 安装成功了。

2.5　建立关联

虽然安装了 Apache、PHP 和 MySQL，但它们还不能协同工作，原因是它们还没建立起关联。这里介绍几个建立关联的方法，使这三个软件组成一个功能强劲的服务器。

2.5.1　设置 Apache 支持 PHP 网页

Apache 目前已经可以正常运行，但它只能解析静态网页，此时还无法解释 PHP 网页。要想让 Apache 能够解析 PHP 网页，必须将 Apache 和 PHP 建立关联，具体操作为：打开 Apache 的配置文件 httpd.conf，如果为默认安装，其位置为 C:\Program Files\Apache Software Foundation\Apache2.2\conf。

查找# LoadModule foo_module modules/mod_foo.so。

在此行后加入下面这一行：

```
LoadModule php5_module C:/php5/php5apache2_2.dll
```

查找 AddType application/x-gzip .gz .tgz，在此行后加入如下这一行：

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

查找 DirectoryIndex index.html index.html.var，将其修改成如下代码：

```
DirectoryIndex index.html index.html.var index.php
```

这样 apache 就可以解释 php 文件了，index.php 也可以充当默认页面了。

2.5.2　测试 Apache 与 PHP 的关联

按照上面的步骤修改完配置之后，重新启动 Apache 使之生效。可按如图 2-25 所示操作，单击 Restart 重新启动。

下面测试服务器是否已经支持 PHP 程序，打开记事本，输入以下代码：

```
<?php                          //PHP 语言开始标记
    phpinfo();                 //调用 phpinfo()函数
?>                             // PHP 语言结束标记
```

使用记事本，通过选择“文件”|“另存为”菜单项，将文件名保存为 phpinfo.php，保存路径为网站根目录，如图 2-26 所示。

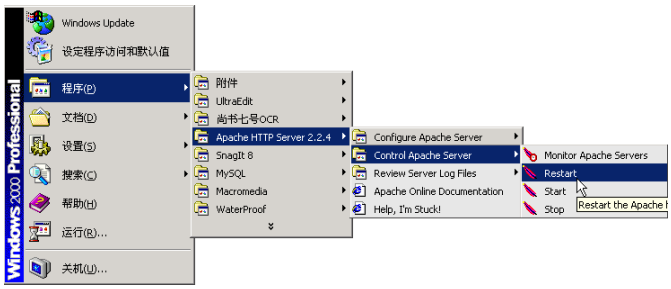

图 2-25　重启 Apache

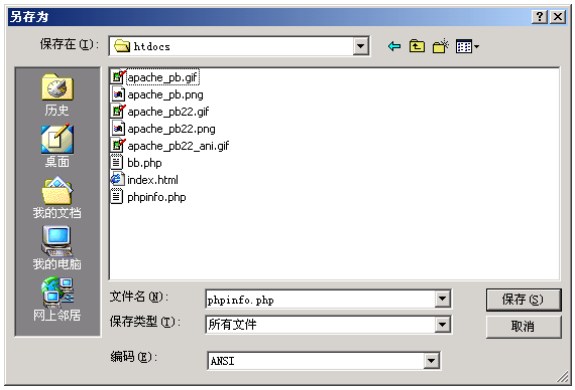

图 2-26　保存 phpinfo.php 文件

保存完文件，打开 IE 浏览器输入 http://lcalhost/phpinfo.php 并按回车键，如果看到如图 2-27 所示的画面，则说明 Apache 和 PHP 已经建立关联了。

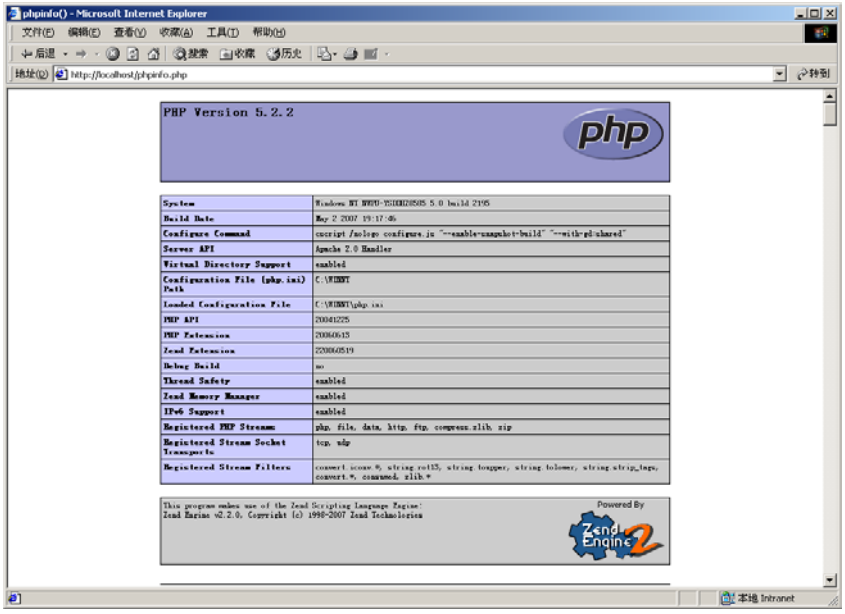

图 2-27　测试 PHP 文件

提示：通过 phpinfo()函数能看出服务器所使用的系统、PHP 版本等信息。

2.5.3 测试调用 MySQL

在 Apache 和 PHP 建立关联的前提下，下面测试 Apache、PHP 和 MySQL 是否协同工作。打开记事本，输入以下代码：

```
01  <?php                                                    //PHP 开始标记
02      $link=mysql_connect("localhost","root","123456");    //123456 改成你的
mysql 密码
03      if(!$link) echo "失败!";                             //输出字符串“失败!”
04      else echo "成功!";                                   //输出字符串“成功!”
05      mysql_close();                                       //关闭数据库
06  ?>                                                       // PHP 结束标记
```

选择“文件”|“另存为”菜单项，将文件名保存为 test.php，保存到网站根目录。打开 IE 浏览器输入“http://lcalhost/test.php”并按回车键，如果看到如图 2-28 所示的情况，则说明 MySQL 已经安装成功了。

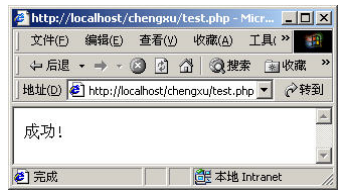

图 2-28 测试 MySQL 数据库

至此，一个基于 Apache+PHP+MySQL 结构的 Web 服务器就建立完成了，以后就可以开始 PHP 的学习之旅了。

2.6 综合练习

1．安装完成后，将下列代码输入到记事本中，然后将文件保存为名为 ceshi.php，保存路径为网站根目录。

```
01  <html>                                  <!--html 标签开始-->
02    <head>                                <!--头标记开始-->
03      <title>PHP 实例</title>             <!--显示标题-->
04    </head>                               <!--头标记结束-->
05    <body>                                <!--body 标签开始-->
06      <?php                               //PHP 开始标记
07          echo "开始学习 PHP";           //输出“开始学习 PHP”
08      ?>                                  //PHP 结束标记
09    </body>                               <!-- body 标签结束-->
10  </html>                                 <!--html 标签结束-->
```

打开 IE 浏览器输入 http://lcalhost/ceshi.php 并按回车键，即可看到如图 2-29 所示的画面。

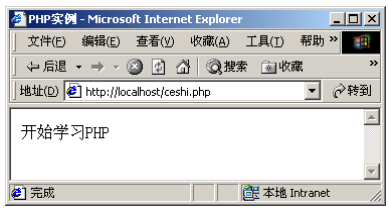

图 2-29 测试

2．安装 MySQL 数据库，将下列代码输入到记事本中，然后将文件保存为 cesql.php，保存路径为网站根目录。

```
<?php
        /* 步骤一：设置初始变量 */
    $host="localhost";                                          //变量赋值
    $user="root";                                               //变量赋值 P
    $password="123456";                                         //变量赋值
        /* 步骤二：连接 MySQL 服务器 */
    $link=mysql_connect($host,$user,$password);                 //连接数据库
        //步骤三：判断连接结果
    if (!$link){                                                //判断连接
        die ("连接数据库失败。".mysql_error( ));                 //输出错误信息
        }
    else{                                                       //连接成功情况
    echo  "MySQL 服务器: $host <br> 用户名称：$user  <br>";     //输出服务器和用户名
    echo  "成功连接数据库。";                                   //输出“成功连接数据库。”
    }
?>
```

打开 IE 浏览器输入 http://lcalhost/cesql.php 并按回车键，即可看到如图 2-30 所示的画面。

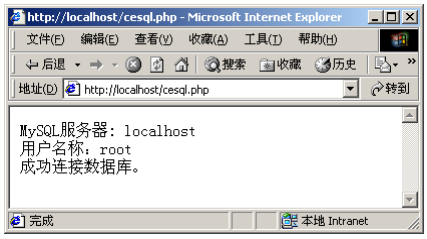

图 2-30　连接 MySQL

2.7　小结

本章首先介绍了 Windows 平台下 Web 服务器的搭建方法，从需求实际出发，确定了服务器的 Apache+PHP+MySQL 的结构形式。接着介绍了三个组件的安装和配置方法，最终使之协同工作，测试通过。通过本章的学习，读者应能熟练配置 Apache+PHP+MySQL 结构的 Web 服务器。

如果想了解更多的关于服务器安装配置知识，可参阅《PHP Web 开发快速入门及实例精选》（陆昌辉等编著：电子工业出版社，2008）和《PHP 5 权威编程》（（美）古曼兹等著：电子工业出版社，2007）。

2.8　习题

一、填空题

1．PHP 可选择的 Web 服务器是_____、_____。
2．PHP 一般选择的数据库是_____。
3．PHP 安装后要配置_____文件以支持相应的功能。
4．PHP 的输出命令是_____。
5．一般使用_____函数测试 PHP 是否安装成功。
6．在 php.ini 中，支持宽字符的文件是_____。

7．要使 PHP 支持以“<?”短标记作为 PHP 代码开始的标记，需找到_____参数，将其值设为 on。

8．连接数据库的函数是_____。

二、选择题

1．以下站点可以下载到 PHP 的是（　　）。

A．http://www.php.net　　B．http://www.apache.org

C．http://www.mysql.com　　D．http://www.asp.net

2．以下站点可以下载到 Apache 的是（　　）。

A．http://www.php.net　　B．http://www.apache.org

C．http://www.mysql.com　　D．http://www.asp.net

3．以下站点可以下载到 mysql 的是（　　）。

A．http://www.php.net　　B．http://www.apache.org

C．http://www.mysql.com　　D．http://www.asp.net

4．下列哪个文件是支持数据库（　　）。

A．extension=php_gd2.dll　　B．extension=php_mysql.dll

C．extension=php_mbstring.dll　　D．extension=php_fdf.dll

三、简答题

1．简述 PHP 的运行环境。

2．简述 PHP 及其服务器安装步骤。

四、编程题

1．了解 PHP 的一些配置文件。

2．了解一些关于 zend 等的安装知识。

第 3 章　HTML 基础

PHP 是一种可嵌入式语言，通常嵌入到 HTML 的程序中，生成动态的 HTML 页面，所以学习 PHP 语言之前，应该先掌握 HTML 语言基础。本章首先从 HTML 语言开始介绍，使读者具备一定语言基础，为以后的 PHP 语言学习做好准备。

学习本章，可以获得以下知识点：

- 掌握 HTML 常用标签和属性；
- 了解 HTML 文档的结构；
- 掌握字体标签；
- 了解段落标签间的区别。

3.1　HTML 简介

HTML 是 Hyper Text Markup Language（超文本语言标记）的英文缩写，HTML 语言是一

种超文本标记语言，就是说其不需要编译，可以直接由浏览器执行（属于浏览器解释型语言）。用 HTML 语言编程的文件称为 HTML 文档，HTML 文档在浏览器中以 Web 页面的形式显示。

3.1.1 了解 HTML 语言

上网对于人们来说应该是相当熟悉了，如今已成为生活中不可缺少的一部分。但是除了专业人士以外，大部分人上网都是在浏览、欣赏网页，而对其实现过程鲜有了解。

网页的原始内容不只是由许多图片和文字组合而成，而是由语言程序构筑成的，利用语言程序来构建文字、图片等对象。在浏览器中，可以利用查看源文件命令看到构筑这个网页的语言。例如，如图 3-1 所示的网页页面，通过“查看”|“源文件”命令可以看到其语言构成，如图 3-2 所示是其网页的 HTML 文件。

图 3-1 网页

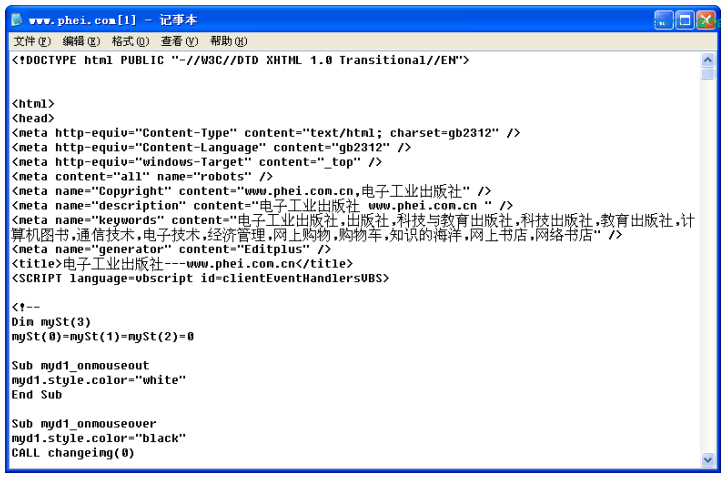

图 3-2　HTML 文件

3.1.2　HTML 语言实例

HTML 语言有其自己的语言风格，下面读者可以通过一个典型的 HTML 语言实例了解其语言风格特点。

【范例 3-1】打开记事本，将示例代码 3-1 所示的 HTML 语言代码输入记事本中。

示例代码 3-1

```
01  <html>                                  <!--html 标签开始-->
02      <head>                               <!--头标记开始-->
03          <title>第一个 HTML 文件</title>   <!-- 文件标题-->
04      </head>                              <!-- 头文件结束-->
05      <body text=red>                      <!-- body 标签开始-->
06      这是我编写的第一个程序!               <!-- 显示内容-->
07      </body>                              <!--body 标签结束-->
08  </html>                                  <!--html 标签结束-->
```

打开记事本，选择“文件”|“另存为”菜单项，将文件保存为 3-1.html，保存形式如图 3-3 所示。

图 3-3　保存代码

【运行结果】打开 IE 浏览器，在地址栏中输入 http://localhost/3-1.html，并按回车键，即可看到示例代码的运行结果如图 3-4 所示。

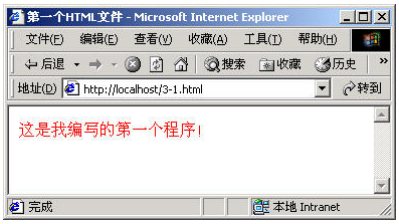

图 3-4　代码运行结果

【代码解析】上述代码是简单的 HTML 代码，有 HTML 标签，<body></body>之间是要输出的内容。

提示：上面代码是一些 HTML 程序，是超文本语言，不需要编译即可直接在浏览器上显示。打开所保存的文件夹，找到刚才保存的文件，双击可直接打开查看，其效果与图 3-4 完全一样。

3.2　HTML 语法

HTML 是一种标记性语言，组成 HTML 语法的元素只有 HTML 标签与 HTML 属性，下面分别介绍这两种元素。

3.2.1　HTML 标签

标签是 HTML 语言中最基本的单位，就如同山、水、动物、植物等组成了丰富多彩的自然界，精彩纷呈的网页也是由最基本的 HTML 标签组成的。标签用来指定一个给定信息的名，形如<html>、<title>、<a>、 都是 HTML 标签，分别指定 HTML 开头、文件标题、超链接、换行等信息。标签由一个开始尖括号（<>）和一个结束尖括号所组成，起分隔或标记文本的作用，其结构形式如下：

```
<标签名>内容</标签名>
```

其标签有以下特点：

- 通常要用两个尖括号括起来，以“<”开始，以“>”结束。
- 标签都是闭合的（闭合就是指标签的最后要有一个“/”来标示结束），但不一定成对出现。比如<body>和</body>是一对标签（<body>是开始标签，</body>是结束标签，在开始和结束标签之间可以有内容），但
就是单独出现的（注意要在最后加上/，以标示其是独立的）。
- 标签大小写无关，例如<body>跟<BODY>表示的意思是一样的，建议使用小写。

【范例 3-2】使用标签 用文字标明这是第几段文字，划分为 4 段，其代码如示例代码 3-2 所示。

示例代码 3-2

```
01  <html>                                    <!--html 标签开始-->
02      <body>                                 <!-- body 标签开始-->
03          <p>这是第一段。<p>                  <!--p 标签-->
```

```
04          <p>这是第二段。</p>                    <!--p 标签-->
05          <p>这是第三段。</p>                    <!--p 标签-->
06          <p>在 HTML 里，用 p 来定义段落。</p>    <!--p 标签-->
07      </body>                                   <!-- body 标签结束-->
08  </html>                                       <!--html 标签结束-->
```

【运行结果】将文件保存为 3-2.html，打开浏览器，在地址栏输入 http://localhost/3-2.html，程序执行结果如图 3-5 所示。

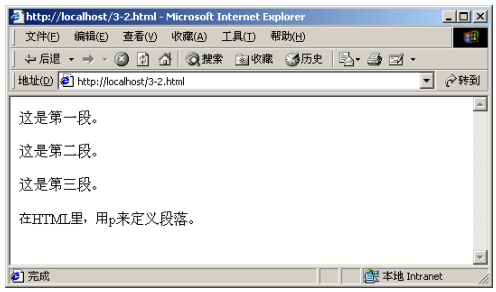

图 3-5　标签示例

【代码解析】上述代码是简单的 HTML 代码，第 03～06 行使用<p>标签换行显示段落内容。

注意：查看程序结果的运行方式是 http://localhost/后加保存的文件名，查看方式同上，以下不再赘述。

3.2.2　HTML 属性

HTML 属性是 HTML 标签的一部分，用来描述特定对象的特性。就如同一部汽车，有价格、品牌、颜色等属性，可以用价格为 20 万元来描述特定的一部汽车。HTML 属性与其类似。HTML 属性一般的结构形式为：

```
<标签名 属性名1="属性值" 属性名2="属性值" ...属性名N="属性值"> </标签名>
```

属性具有以下特点。

- 标签可以拥有多个属性。
- 属性由属性名和属性值成对出现。

【范例 3-3】利用链接标签 a 编写一段 HTML 代码，通过设置属性，链接到 http://www.baidu.com 页面。其程序如示例代码 3-3 所示。

示例代码 3-3

```
01  <html>                                       <!--html 标签开始-->
02      <head>                                    <!--头标记开始-->
03          <title>属性</title>                   <!--显示标题-->
04      </head>                                   <!--头标记结束-->
05      <body>                                    <!--body 标签开始-->
06      标签<a>是超链接标签.使用 href 属性，可以定义链接的位置(URI)代码:
07      <a href="http://www.baidu.com/">          <!-- a 标签开始-->
08      尽情学习                                  <!--链接内容 -->
09      </a>                                      <!-- a 标签结束-->
10      <body>                                    <!-- body 标签结束-->
11  </html>                                       <!-- html 标签结束-->
```

【运行结果】将文件保存为 3-3.html，打开浏览器，在地址栏输入 http://localhost/3-3.html，

程序执行结果如图 3-6 所示。

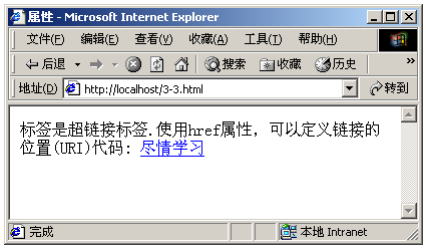

图 3-6　属性示例

【代码解析】程序第 07 行运用<a>标签，定义一个超链接，标签以<a>开始，以结束，标签只对其之间的内容有效。属性名是 href，属性值为 http://www.baidu.com。单击“尽情学习”链接就能进入链接界面。

提示：标签中可以使用多个属性，也可以不使用属性。

3.2.3　HTML 注释

HTML 注释用来解释程序的功能或注意事项，以便后来查找。HTML 的注释形式为：

```
<!--注释的内容-->
```

每个注释语句的内容前后必须有“--”识别，在“<!”和“>”之间出现空格是允许的。注释可以是任何内容、但不会显示在浏览器上。

【范例 3-4】在浏览器上输出语句“显示此内容”，语句“这内容你看不到。”不在浏览器上显示，其程序如示例代码 3-4 所示。

示例代码 3-4

```
01  <html>                                          <!-- html 标签开始-->
02      <head>                                       <!--头标记开始-->
03          <title>注释</title>                      <!--显示标题-->
04      </head>                                      <!--头标记结束-->
05      <body>                                       <!--body 标签开始-->
06      <!--                                         <!--注释开始-->
07      <font  color="red">这内容你看不到。</font>   <!--字体标签-->
08      -->                                          <!-- 注释结束-->
09      <font  color="red">显示此内容</font>         <!-- 字体标签-->
10      </body>                                      <!--body 标签结束-->
11      </html>                                      <!--html 标签结束-->
```

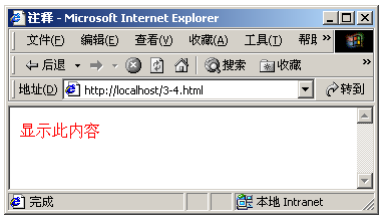

图 3-7　注释示例

【运行结果】将文件保存为 3-4.html，打开浏览器，在地址栏输入 http://localhost/3-4.html，程序执行结果如图 3-7 所示。

【代码解析】在第 03 行<!--和第 05 行-->之间的部分不能显示在浏览器上，其使用的是注释内容，浏览器上只能看到红色的“显示此内容”。

3.3 HTML 文档基本结构

编写 HTML 程序要按照其基本结构形式编写，一个完整的 HTML 文件结构由 HTML 主体标记、头部标记、主体区标记组成，下面分别介绍这几个标记。

3.3.1 HTML 主体标记

主体标签是 HTML 开始和结束的标记，其结构形式为：

```
<html>….</html>
```

HTML 文件中包含<head>和<body>标记。HTML 文档中所用的内容都应该在这两个标记之间，一个 HTML 文档总是以<html>开始，以</html>结束。例如：

```
<html>
<head>
    <title>HTML 主体标记</title>
</head>
<body>
   HTML 文件的正文写在这里……
</body>
</html>
```

程序是以<html>开头以</html>结尾的 HTML 结构程序，包含有<head>、<body>等标签。

3.3.2 HTML 头部标记

HTML 文件的头部标记用于放置页面的标题及文件信息等内容，通常将其两个标记之间的内容统称为 HTML 的头部，其结构形式为：

```
<head>…</head>
```

HTML 头部标记主要包括页面的一些基本描述的语句，其他如引用的 JavaScript 和 CSS，一般也定义在 HEAD 头元素中，其常用的头部标记有<title>、<meta>、<base>、<style>等组成，其含义如下所示。

- <title>：是显示在浏览器标题栏上的文件标题，用于说明文件的性质。每个 HTML 文档都应该有标题，在 HTML 文档中，标题文字位于<title>和</title>之间。<title>和</title>位于 HTML 文档的头部，也就是<head>和</head>之间。
- <meta>：该标记的功能是定义页面中的信息。其标记通过属性来定义文件信息的名称、内容等，其能够提供文档的关键字、描述等多种信息。但这些文件信息并不会出现在浏览器页面的显示之中，只会显示在源代码中。
- <base>：该标记可以设定 URL 地址，一般常用来设定浏览器中文件的绝对路径。在浏览器中浏览的时候这些位置会自动附在绝对路径后面，成为完整的路径。
- <style>：设定 CSS 层叠样式表的内容。

为更好地理解上述标签的作用，下面通过一个实例来具体使用这些标签。例如：

```
01  <head>                                                    <!--头标记开始-->
02      <title>头部标记</title>                                <!--显示标题-->
03      <meta name="description" content="php 基础教程">       <!--设置内容-->
04      <meta name="keywords " content="php,html ,教程 基础"> <!--设置关键字-->
05      <meta name="author" content="php ">                   <!--设置作者 -->
06      <style>                                               <!--样式开头标记>
07          body {background-color:white; color:black;}       <!--内容样式 >
```

```
08          h1 {font: 18pt arial bold;}                    <!--字体样式>
09      </style>                                           <!--样式结束标记>
10  </head>                                                <!--头标记结束-->
```

程序第 01 行以<head>标记头部标记开始，第 02 行<title>与</title>之间的内容显示在浏览器的标题栏上，第 03～05 行设置 META 属性，分别设定主要内容，关键字和作者信息。第 06～09 行设置 CSS 样式。定义背景颜色，字体等。

提示：CSS 样式一般用来设定网站的布局。

3.3.3 HTML 主体区标记

HTML 文件的主体区标记，绝大多数 HTML 内容都放置在这个区域里面，通常位于</head>标记之后，</html>标记之前，其结构形式为：

```
<body>…</body>
```

<body>标示出文件主体区，主体区是网页的主要部分，如网页中显示的新闻、图片、音乐、视频等内容，都是<body>和</body>之间的部分。通过<body>标签可以设置主体部分的背景颜色、文字大小、链接颜色等属性，<body>元素常用属性如表 3-1 所示。

表 3-1 <body>标记常用属性

属 性	描 述
text	设定文字颜色
bgcolor	设定背景颜色
background	设定背景图像
link	设定默认链接颜色
alink	设定单击时的链接颜色
vlink	设定访问后的链接颜色

text 属性用来设定整个页面文字的颜色，Bgcolor 属性用来设定页面的背景颜色，其基本语法为：

```
<body text= VALUE bgcolor= VALUE >
```

通过 text 属性定义文字的颜色，VALUE 指的就是颜色的值，设置颜色的方式有两种，一种是直接用文字指定设定的颜色，如红、黄等，另一种是指定颜色编码，如 FFFFFF、CCCCCCC、000000 等，bgcolor 属性的设置方法同 text 的一样。

链接 link 是网页中最基本的元素之一，在浏览器的默认情况下，链接文字、普通文字访问过的文字用不同的颜色区分开，这样有助于用户判断是否需要打开该链接，同时也能使页面炫目。

其基本语法为：

```
<body link="COLO_VALUE" alink="COLOR_VALUE" vlink="COLOR_VALUE">
```

通过 link、alink、vlink 分别设置链接、单击时和访问后的颜色，COLOR_VALUE 为链接显示的颜色。

【范例 3-5】通过 link、alink、vlink 分别设置链接、单击时、访问后的不同的颜色，以示区别。其程序如示例代码 3-5 所示。

示例代码 3-5

```
01  <html>                                                    <!--html 标记开始-->
02      <head>                                                <!--头标记开始-->
03          <title>页面文字链接</title>                        <!--显示标题-->
04      </head>                                               <!--头标记结束-->
05      <body bgcolor=#336699 text=#FFFFFFF link=# cccccc
06          alink="#00FF00" vlink="# FF0000" >                <!--body 标记开始-->
07      <center>                                              <!-- 居中对齐-->
08      <H2> 设定不同的链接颜色<H2>                           <!-- 标题文字-->
09      <a href="http://www.sina.com">默认的链接颜色</a>      <!-- 链接颜色-->
10      <p>                                                   <!-- 段落标记-->
11      <a href="http://www.baidu.com">默认的链接颜色</a>     <!-- 链接标记-->
12      <p>                                                   <!-- 段落标记-->
13      <a href="http://www.163.com">默认的链接颜色</a>       <!-- 链接标记-->
14      </center>                                             <!-- 居中对齐-->
15      </body>                                               <!--body 标记结束-->
16  </html>                                                   <!--html 标记结束-->
```

【运行结果】将文件保存为 3-5.html，打开浏览器，在地址栏输入 http://localhost/3-5.html，程序执行结果如图 3-8 所示。

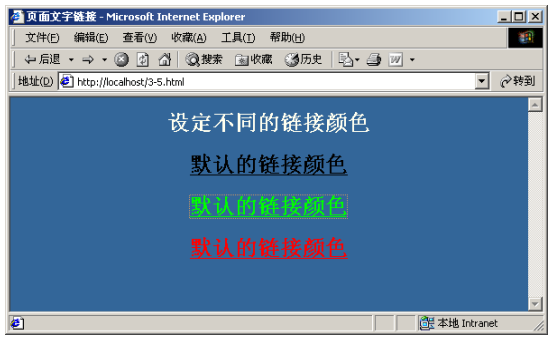

图 3-8　链接颜色

【代码解析】代码第 05 行定义了背景颜色、文字颜色、链接的颜色，第 09、11、13 使用标签<a>分别设置了三个超链接。

警告：Text属性设定的文字属性对整个主体区内起作用，但如果对某段文字单独定义颜色，则Text属性所定义的颜色将失效。

3.4　字体标签

一般情况下，文字在网站中出现的频率是最高的，所以文字的处理对整个页面来说显得非常重要，对文字的处理主要是指文字的字体大小、颜色、排版等的设置。

3.4.1　标题字标记

标题字标记是用来标识标题的型号的，用<H>来表示。一般标题字有 6 种不同的型号，每一种型号在字号上都有明显的区别，一般用标题字来强调段落要表现的内容。在 HTML 中定义了六级标题，从一级到六级，每级标题的字体大小依次递减，其基本语法为：

```
01  <H1>……<H1>                                               <!--一级标题-->
```

```
02  <H2>……<H2>                    <!--二级标题-->
03  <H3>……<H3>                    <!--三级标题-->
04  <H4>……<H4>                    <!--四级标题-->
05  <H5>……<H5>                    <!--五级标题-->
06  <H6>……<H6>                    <!--六级标题-->
```

一级标题使用最大的字号表示，六级标题使用最小的字号表示。

【范例 3-6】按照不同级别的标题字输出“PHP 基础教程”，查看其效果。其程序如示例代码 3-6 所示。

示例代码 3-6

```
01  <html>                          <!-- html 标记开始-->
02      <head>                      <!--头标记开始-->
03      <title>页面标题字体</title>  <!--显示标题-->
04      </head>                     <!--头标记结束-->
05      <body>                      <!-- body 标记开始-->
06          <H1> PHP 基础教程</H1>  <!--字体标记-->
07          <H2> PHP 基础教程</H2>  <!--字体标记-->
08          <H3> PHP 基础教程</H3>  <!--字体标记-->
09          <H4> PHP 基础教程</H4>  <!--字体标记-->
10          <H5> PHP 基础教程</H5>  <!--字体标记-->
11          <H6> PHP 基础教程</H6>  <!--字体标记-->
12      </body>                     <!-- body 标记结束-->
13  </html>                         <!-- html 标记结束-->
```

【运行结果】将文件保存为 3-6.html，打开浏览器，在地址栏输入 http://localhost/3-6.html，程序执行结果如图 3-9 所示。

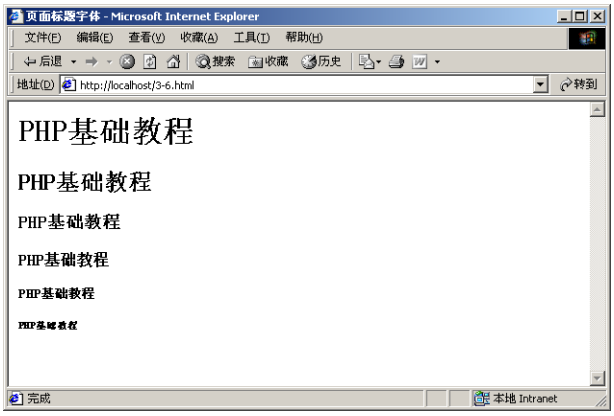

图 3-9　标题字

【代码解析】代码 06～11 行使用 6 种不同的标题字标记，其在浏览器上显示不同大小的标题字。

注意：标题字只是字体标签的一种，字体标签有很多种，恰当地使用标签能使页面美观大方。

3.4.2　标题字的对齐属性

标题字可以在页面中实现水平方向向左、中、右的对齐方式，便于文字在页面中的编排。

在标题标记中，最主要的属性是 ALIGN 属性，用于定义标题段落的对齐方式，使页面更整齐。

其基本与语法为：

```
< ALIGN=LEFT>        标题居左对齐
< ALIGN=CENTER>      标题居中对齐
< ALIGN=RIGHT>       标题居右对齐
```

【范例 3-7】在页面中以三种不同的对齐方式显示“对齐方式”标题字。其程序如示例代码 3-7 所示。

示例代码 3-7

```
01  <html>                                          <!-- html 标记开始-->
02     <head>                                       <!--头标记开始-->
03      <title>对齐方式</title>                      <!--显示标题-->
04      </head>                                     <!--头标记结束-->
05    <body>                                        <!-- body 标记开始-->
06      <H1> 对齐方式</H1>                          <!--字体标记-->
07      <H2 ALIGN=CENTER> 对齐方式</H2>             <!--字体标记-->
08      <H3 ALIGN=LEFT> 对齐方式</H3>               <!--字体标记-->
09      <H4 ALIGN=RIGHT>对齐方式</H4>               <!--字体标记-->
10    </body>                                       <!-- body 标记结束-->
11  </html>                                         <!-- html 标记结尾-->
```

【运行结果】将文件保存为 3-7.html，打开浏览器，在地址栏中输入 http://localhost/3-7.html，程序执行结果如图 3-10 所示。

图 3-10　对齐方式

【代码解析】代码第 06 行设置了标题字标记，第 07 行对标题字设置居中对齐，第 08 行对标题字设置左对齐，第 09 行对标题字设置右对齐。

3.4.3　文字的修饰标记

在 HTML 文件中，可以加入多种文字的修饰标记，能够使文字呈现不同的样式，常见的文字修饰标记如表 3-2 所示。

表 3-2　文字修饰标记

标　记	描　述
<b>	粗体
<i>	斜体
	斜体
<u>	下画线

续表

标　记	描　述
<s>	删除线
<sup>	上标
<sub>	下标
<big>	大号字
<small>	小号字
<var>	声明变量
<samp>	等宽
tt	打印字体

这些文字修饰标记使文本产生多种效果，不过要与具体环境结合起来才能产生很好的搭配效果。下面通过实例演示其文字修饰标记的使用效果。

【范例 3-8】利用文字修饰标记，粗体显示“文字修饰标记”，斜体显示“下面是一些代数方程实例”，列出方程，删除错误的文字“PHP 是一门难学的语言。”。注意要合理搭配才能收到好的效果。其程序如示例代码 3-8 所示。

示例代码 3-8

```
01  <html>                                                    <!-- html标记开始-->
02      <head>                                                <!--头标记开始-->
03      <title>文字修饰标记</title>                            <!---显示标题-->
04      </head>                                               <!--头标记结束-->
05      <body>                                                <!--body 标记开始-->
06      <center><b> 文字修饰标记</b></center>                  <!--文字修饰标记-->
07      <I>下面是一些代数方程实例：</I><br>                    <!--文字修饰标记-->
08      3X<sup>2</sup>+2X+3=17<br>                            <!--文字修饰标记-->
09      5X<sub>1</sub>+2X<sub>2</sub>=16<br>                  <!--文字修饰标记-->
10      下面的文字是<small>错误</small>的,将会被<big>删除:</big> <!--文字修饰标记-->
11      <br>                                                  <!--换行标记-->
12      <s>PHP 是一门难学的语言。</s>                          <!--文字标记-->
13      </body>                                               <!--body 标记结束-->
14  </html>                                                   <!--html 标记结束-->
```

【运行结果】将文件保存为 3-8.html，打开浏览器，在地址栏输入 http://localhost/3-8.html，程序执行结果如图 3-11 所示。

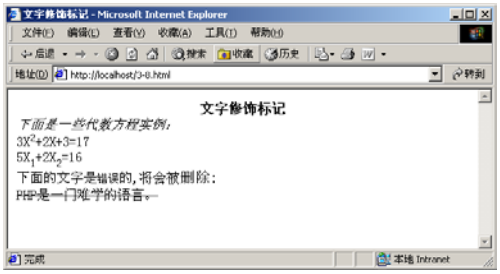

图 3-11　文字修饰

【代码解析】代码第 06 行设置了标题文字居中对齐，第 07 行使用标签<I></I>设置斜体字，第 08、09 行使用上下标书写方程，第 10 行分别使用大号和小号字体来起强调作用，第 11 行使用删除线删除文字。

提示：可以对一段文字，采用多种效果，如采用黑体的同时还可以采用斜体。

3.4.4 字体标记

常用的字体标记包括字体类别、字号、颜色等。如果希望更改字体的类别、字号和颜色，可以使用<font>标记，其包含的标签如表 3-3 所示。

表 3-3 字体标记标签

属 性	描 述
face	字体
size	字号
color	颜色

其语法结构如下：

```
<font face="font_name" size="value" color="value"></font>
```

face：指的是使用字体的类别，任何安装在 Windows 系统中的文字都可以显示在浏览器中。对操作的电脑，选择“开始”|“设置”|“控制面板”菜单项，双击“字体”选项，可以看到安装的所有字体，如图 3-12 所示。

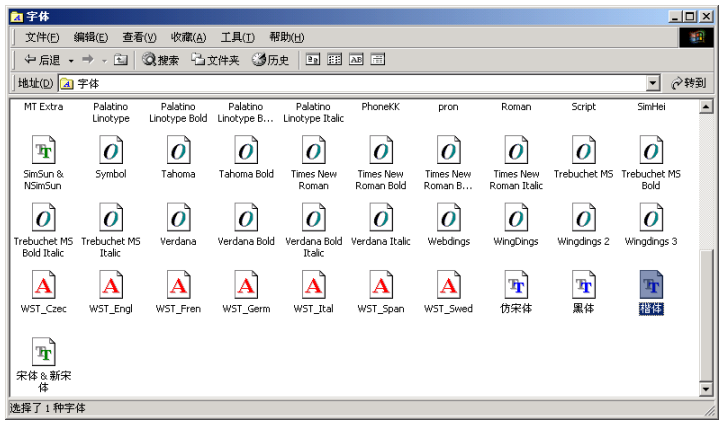

图 3-12 字体类别

一般来说，在网页中不应该使用过于特殊的字体，因为浏览此网页的人不一定都装有此种字体。如果浏览器带有一些特殊字体，而在本机上没有安装此种字体，则只能以普通的默认字体来显示。

- size：字体的字号。字号没有绝对的大小标准，其大小只是相对于默认字体而言的。
- color：文字的颜色。丰富的字符颜色毫无疑问能够极大地增强文档的表现力。

【范例 3-9】使用不同的字体标签，在浏览器上显示“PHP 基础教程”不同效果，程序如示例代码 3-9 所示。

示例代码 3-9

```
<html>                                          <!-- html 标记开始-->
    <head>                                      <!--头文件开始-->
        <title>字体标记</title>                  <!--文件标题-->
        </head>                                 <!--头标记结束-->
```

```
05      <body>                                                    <!--body 标记开头-->
06      <font face=宋体 size=5 color=red >PHP 基础教程</font><br>
                                                                  <!--字体标记-->
07      <font face=黑体 size=3 >PHP 基础教程</font>               <!--字体标记-->
08      <font face="times new roman" size=3 color=#000000>PHP</font>
                                                                  <!--字体标记-->
09      <font face="隶书" size=3 color=#cccccc>基础</font>        <!--字体标记-->
10      <font face="隶书" size=3 color=#00ffff>教程</font>        <!--字体标记-->
11      </body>                                                   <!--body 标记结尾-->
12  </html>                                                       <!--html 标记结尾-->
```

【运行结果】将文件保存为 3-9.html，打开浏览器，在地址栏输入 http://localhost/3-9.html，程序执行结果如图 3-13 所示。

图 3-13　字体标记

【代码解析】代码第 06、08、09、10 行分别设置了字体的类别、大小和颜色，这些都能在浏览器上清楚地表现出来。

3.5　段落标记

段落标记主要用于定义段落布局，包含行中断标签、分段标签、不换行标签等多种标签，以适应不同的分段需要。

3.5.1　段落标记

标记表示段落的开始，其可以成对使用即 … ，也可以单独使用 ，这两种形式都能标记一个段落。其具体用法通过下列实例讲解。

【范例 3-10】利用段落标记 对文字“经典广告词　不管是一大步还是一小步，都是世界的脚步——带你前行”，进行段落的标记。其程序如示例代码 3-10 所示。

示例代码 3-10

```
01  <html>                                                       <!-- html 标记开始-->
02      <head>                                                   <!--头文件开始-->
03      <title>段落文字</title>                                  <!--文件标题-->
04      </head>                                                  <!--头标记结束-->
05      <body>                                                   <!-- body 标记开头-->
06      <p> 经典广告词</p>                                       <!--段落标记-->
07       不管是一大步还是一小步，都是世界的脚步<p>              <!--段落标记-->
08      <p>——带你前行                                           <!--段落标记-->
09      </body>                                                  <!-- body 标记结尾-->
10  </html>                                                      <!--html 标记结尾-->
```

【运行结果】将文件保存为 3-10.html，打开浏览器，在地址栏中输入 http://localhost/3-10.html，程序执行结果如图 3-14 所示。

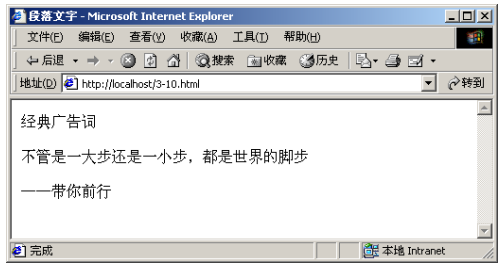

图 3-14　段落标记

【代码解析】代码第 06 行运用了段落标记 进行下一段的分段，第 07 行利用段落标记 进行下一段的分段，第 08 行开头使用段落标记 ，由于段落标记 不能连续使用，因此在浏览器上只显示了一个段落的标记。

提示： 标签不能连用，如果同时使用多个 标签，其效果与使用一个 标签的效果相同。

3.5.2　行中断标签

段落与段落之间一般是隔行换行的，这样有时显得文字的行间距过大，此时可以使用行中断换行标记，完成文字的紧凑换行显示。其语法结构如下：

```
<br>
```

【范例 3-11】对“孔子曰：苟有用我者，期月而已，三年有成。”使用行中断标签换行，分段显示。其程序如示例代码 3-11 所示。

示例代码 3-11

```
01  <html>                                                    <!--html 标记开始-->
02      <head>                                                <!--头文件开始-->
03      <title>行中断标记</title>                              <!--显示标题字-->
04          </head>                                           <!--头文件结尾-->
04      孔子曰：<br>                                          <!--换行标记-->
05           苟有用我者，期月而已，三年有成。  <!--显示内容-->
06      </body>                                               <!--头文件结束-->
07  </html>                                                   <!--html 标记结尾-->
```

【运行结果】将文件保存为 3-11.html，打开浏览器，在地址栏中输入 http://localhost/3-11.html，程序执行结果如图 3-15 所示。

图 3-15　行中断标记

【代码解析】代码第 04 行使用强制换行标签
进行换行，第 05 行用几个 添加了几个空格。

3.5.3 不换行标签

不换行标签<nobr>，其语法结构形式如下：

```
<nobr>…..</nobr>
```

<nobr>标签的作用是强制浏览器不换行，如果不使用此标签，浏览器会自动将超出浏览显示宽度文字换行显示，以适应浏览器的显示窗口。

【范例 3-12】对长句子“这是一本专业详尽的 PHP 书籍，浅显易懂，适合于初级阶段学习。”分别进行强制不换行显示和默认处理做对比，对比其效果。其程序如示例代码 3-12 所示。

示例代码 3-12

```
01  <html>                                          <!-- html 标记开始-->
02      <head>                                      <!--头文件开始-->
03      <title>文字不换行</title>                    <!--显示标题-->
04      </head>                                     <!--头文件结束-->
05      <body>                                      <!-- body 标签开始-->
06      <p>这是一本专业详尽的 PHP 书籍，浅显易懂，
07                          适合于初级阶段学习。</p>  <!--分段标记-->
08      <nobr>这是一本专业详尽的 PHP 书籍，浅显易懂，
09                      适合于初级阶段学习。</nobr>   <!--不换行标记-->
10      </body>                                     <!--body 标签结束-->
11  </HTML>                                         <!-- html 标记结尾-->
```

【运行结果】将文件保存为 3-12.html，打开浏览器，在地址栏中输入 http://localhost/3-12.html，程序执行结果如图 3-16 所示。

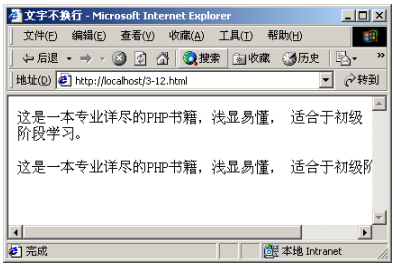

图 3-16 不换行标签<NOBR>

【代码解析】代码的第 06、08 行输入了两行相同的文字，其差别就在于是否使用了不换行标签，最终的显示效果如图 3-15 所示。

注意：一个换行使用一个标签
，多个换行可以连续使用多个
标记。

3.6 综合练习

1．用 HTML 语言编写苏轼的词“念奴娇·赤壁怀古”中的一段，要求设置背景颜色、字体大小、颜色等，要有段落层次感、有超链接。其词内容为：大江东去，浪淘尽，千古风流人物。故垒西边，人道是，三国周郎赤壁。乱石穿空，惊涛拍岸，卷起千堆雪。

提示：使用 3.1 节所讲过的内容标签、属性，输入一段文字，使得页面看起来有条理，能够吸引人的目光。程序如示例代码 3-13 所示。

示例代码 3-13

```
<html>                                                    <!-- html 标记开始-->
    <head>                                                <!--头文件开始-->
    <title>我的家园</title>                                <!--显示标题-->
    </head>                                               <!--头文件结束-->
    <body bgcolor="#336699">                              <!--body 标签开始-->
    <center>                                              <!--居中开始标记-->
    <font size=7 face=黑体><a  href="http://www.baidu.com">念奴娇·赤壁怀古
    </a></font><p>                                        <!--设置字体标记-->
    <font size=2 face=宋体>作者：苏轼</font><p>           <!--设定字体和段落-->
    <font size=4 face=楷体>大江东去，浪淘尽，千古风流人物。<br>故垒西边，人道是，
    三国周郎赤壁。<br>乱石穿空，惊涛拍岸，卷起千堆雪。</font>
        </center>                                         <!--居中结尾标记-->
    </body>                                               <!--body 标签结束-->
</html>                                                   <!-- html 标记结尾-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/3-13.html，程序执行结果如图 3-17 所示。

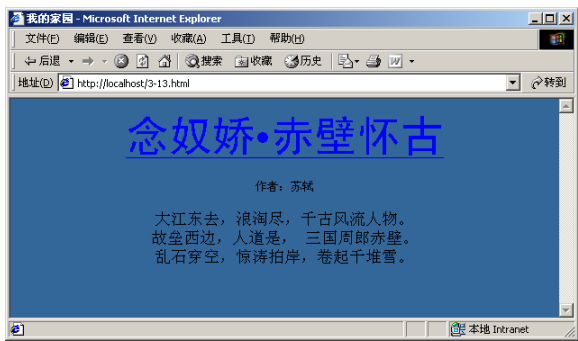

图 3-17　程序执行结果

2．用 HTML 编写 PHP 从第 1 章到第 5 章的目录，列出其参考书及其出版社。设定其内容为“书籍”，关键词为：“php，html，教程，基础”，作者为“php”等内容。

参考书：

HTML 从入门到精通　　电子工业出版社

网站制作基础必学 HTML　　电子工业出版社

最新 HTML 4.0 网页制作教程　　电子工业出版社

提示：可根据上面介绍的 HTML 语言标签，编写一个结构紧凑、层次分明的目录。程序如示例代码 3-14 所示。

示例代码 3-14

```
<html>                                                    <!-- html 标记开始 02-->
    <HEAD>                                                <!--头标记开始-->
        <title>PHP 书目录</title>                         <!--显示标题-->
        <meta name="description" content="书籍">          <!--设置内容-->
        <meta name="keywords " content="php,html ,教程 基础">
                                                          <!--设置关键字-->
        <meta name="author" content="电子工业">           <!--设置作者 -->
        <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
    </HEAD>                                               <!--头标记结束-->
    <body>                                                <!--body 标签开始-->
```

```
11          <center><font ><a  href="http://www.baidu.com"><H1>PHP 书目录</H1>
12                  </a></font><p>                              <!-- 设置字体和段落
-->
13      </center><p>                                            <!--居中结尾标志-->
14      <a name="C1"><b>第 1 章</b> <i>初识 PHP</i></a>
15      </br>                                                   <!--段落标记-->
16      <a name="C2"><b>第 2 章</b> </a><i>安装</i>         <!--设定超链接-->
17          </br>                                               <!--段落标记-->
18          <a name="C3"><b>第 3 章</b> <i>HTML 基础</i></a><br>
19          <a name="C4"><b>第 4 章</b></a> <i>PHP 基础知识</i></br>
20          <a name="C5"><b>第 5 章</b></a> <i>常用流程控制</i></br>
21          <hr>                                                <!--横线标记-->
22          <p>                                                 <!--段落标记-->
23          参考书 <p>                                          <!--段落标记-->
24          HTML 从入门到精通
25          <a href="mailto:info@sina.com?subject=Hello">电子工业出版社</a>
26          </br>                                               <!--段落标记-->
27          网站制作基础必学 HTML <a href="mailto:info@sina.com?subject=Hello">
28          电子工业出版社</a></br>
29      最新 HTML 4.0 网页制作教程 <a href="mailto:info@sina.com?subject=Hello">电子
30          工业出版社</a>                                      <!--设定超链接-->
31      </body>                                                 <!--body 标签结束-->
32  </html>                                                     <!--html 标记结尾-->
```

【运行结果】打开 IE 浏览器，在地址栏中输入 http://localhost/3-14.html，程序执行结果如图 3-18 所示。

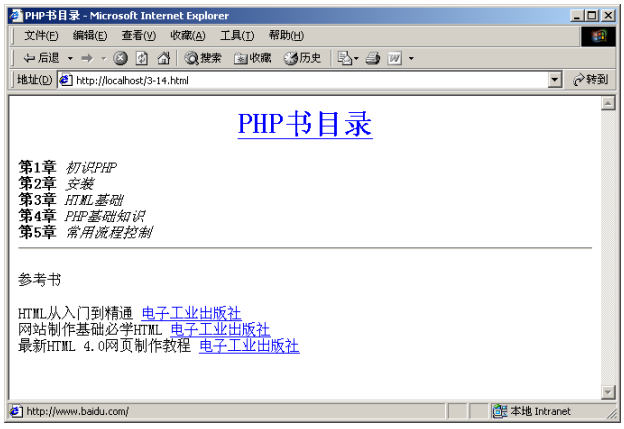

图 3-18　示例执行代码

3.7　小结

本章介绍了 HTML 的结构形式和常用标签，HTML 是网络程序开发的基础内容，也是编程语言的基础，掌握本章内容会对今后 PHP 语言的学习有很大的促进作用。如果读者对本章内容还有疑问，可参考《HTML 从入门到精通》（[美]Aark Andrews 著，蒋明、程昱译．北京：机械工业出版社，1997）和《网站制作基础必学 HTML》（原奕等．北京：机械工业出版社，2005）。

3.8 习题

一、填空题

1．_____是 HTML 语言中最基本的单位。
2．标签由一个_____和一个_____所组成，起分隔或标记文本的作用。
3．一个 HTML 文档总是以_____开始，以_____结束。
4．标题字标记是用来标识标题的型号的，用_____来表示。
5. 更改字体的类别、字号和颜色可以使用_____标记，其包含的标签有_____、____和____。
6．在 HTML 中，段落主要由标记_____定义。
7．行中断标记为_____。
8．在 HTML 中使用标记_____来强制浏览器不换行显示。

二、选择题

1．标题右对齐的方法是（　　）。

A．<align=LEFT>　　B．<align =CENTER>
C．<align =RIGHT>　　D．<align =MIDDLE>

2．下面选项中属于字体标记标签的是（　　）。

A．face　　B．size　　C．color　　D．LEFT

3．HTML 的头部标记有（　　）。

A．<title>　　B．<meta>　　C．<base>　　D．<style>

4．HTML 语法中，文字修饰标记有数种，其中表示下画线的标记为（　　）。

A．　　B．<u>　　C．<s>　　D．<sup>

三、简答题

1．简述 HTML 主体区标记有哪些。
2．简述文字的修饰标记及特性。
3．列举 HTML 的段落标记。

四、编程题

1．用 HTML 语言编写一个网页，要求主体区标记完备，段落层次清晰。
2．编写一个 HTML 页面，要求能够表现所有的字体标记和修饰标记。

第 4 章　PHP 基础知识

前面介绍了 HTML 语言的基本语法和结构，以后各章将开始进入真正的 PHP 语言学习阶段。本章主要从 PHP 基础知识入手，逐步深入学习 PHP 语法结构，有步骤地培养编程思想。

学习本章，可以获得以下知识点：

- 了解 PHP 语言的标识符和注释；

- 掌握 PHP 的常用数据类型；
- 学会运用 PHP 的变量；
- 了解定义 PHP 常量的方法；
- 掌握运算符及各运算符的优先级。

4.1 PHP 的基本语法

语法是一门语言的基础。PHP 也有语法，它的基本语法主要包括其语言风格、标识符、注释和输出命令等，以下详细介绍 PHP 的基本语法。

4.1.1 语言风格

语言风格是一种编程语言形成的约定或习惯，PHP 也有一套自己的书写习惯和约定，它在使用的时候一般嵌套到 HTML 语言中，利用标签来区分是 HTML 或 PHP 语言，下面通过实例来演示。

【范例 4-1】打开记事本，将示例代码 4-1 所示的程序代码键入到记事本中，然后将文件保存为 4-1.php 格式。

示例代码 4-1

```
01      <html>                                  <!--html 标签开始-->
02          <head>                              <!--头文件标记开头-->
03              <title>Hello World !</title>    <!--显示标题-->
04          </head>                             <!--头文件结尾-->
05          <body>                              <!--body 标签开始-->
06      <?php                                   //php 开始标记
07              echo "Hello World!";            //输出
08      ?>                                      <!--php 结束标记-->
09          </body>                             <!--body 标签结束-->
10  </html>                                     <!--html 标签结束-->
```

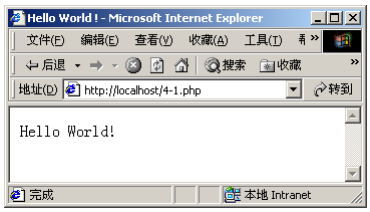

图 4-1 “hello world”运行结果

【运行结果】打开 IE 浏览器，在地址栏中输入 http://localhost/4-1.php，查看其运行结果，即可看到示例代码的运行结果如图 4-1 所示。

【代码解析】上述代码大部分是 HTML 标识符，只有第 06～08 行，即“<?php”和“?>”之间嵌入的代码才是 PHP 语句，充分体现了 PHP 作为 HTML 脚本语言的特点。程序中的“;”表示一条语句的结束，echo 语句用于向浏览器输出，它是 PHP 中的标准输出语句。

注意：如果语句声明是 PHP代码的最后一行，也就是说它后面是 PHP 代码结束标记，这时也可以不加分号。

4.1.2 PHP 在 HTML 中的嵌入

PHP 在大多数情况下都是嵌入在 HTML 文档中使用的，“<?php”和“?>”是 PHP 的分界符，其中包含的代码是 PHP 代码，PHP 服务器解析的是这段代码。用一个形象的比喻，PHP 服务器相当于运动员，“<?php”相当于运动员的起跑线，“?>”相当于运动员的终点线，运动

员要根据起跑线和终点线的标志跑完所包含的路段，PHP 服务器就是根据“<?php”和“?>”标志来判断要解析哪些程序的。PHP 共允许 4 种类型的分界符，其如下表示：

- “<?php”和“?>；
- “<?”和“?>” ；
- “<script language="php"”和“</script>”；
- “<%”和“%>”。

第 1 种方法是最常用的，它用“<?php”和“?>”直接标示 PHP 程序，只要支持 PHP 语言的服务器就能支持这种分界符。

第 2 种方法是在小于号后加上问号，同前者的区别是在前面的“<?”后没有 PHP 字符，在“<?”之后的就是 PHP 的程序代码。在程序代码结束后，加入问号和大于号两个符号即可结束 PHP 的代码了。如果要使服务器支持此种分界符，需将 php.ini 文件中 short_open_tag 的值设为 on。

第 3 种方法是用分界符“<script language="php"”和“</script>”来标示 PHP 程序，类似于写 JavaScript 的方式。

第 4 种方法对于从 Windows NT/2000 平台的 ASP 投向 PHP 的用户来说更熟悉，只要用 PHP 3.0.4 或更高版本的服务器都可以用“<%”开始，“%>”结束。要使用此格式，要将 php.ini 文件中的 asp_tags 的值设为 on，如图 4-2 所示。

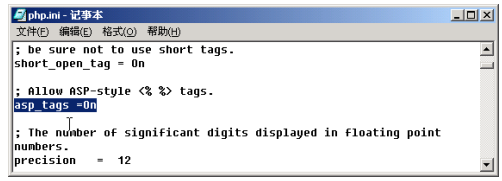

图 4-2 php.ini 文件设置

【范例 4-2】分别利用“<?php”和“?>”、“<?”和“?>”、“<script language="php"”和“</script>”、<%”和“%>这 4 种类型的分界符，将一段 PHP 程序嵌入到 HTML 程序中，并输出这各自格式的分界符，程序如示例代码 4-2 所示。

示例代码 4-2

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>PHP 标签</title>                           <!--文件标题-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        <center><H1> PHP 的四种标记</H1></center>         <!--居中显示标题-->
<?php                                                     //PHP 开始标记
        echo "利用< ?php   ? >标记";                      //输出
?>                                                        <!--PHP 结束标记-->
    <hr>                                                  <!--水平线标签-->
<?                                                        //PHP 开始标记
    echo "利用< ?  ? >标记";                              //输出
?>                                                        <!--PHP 结束标记-->
    <hr>                                                  <!--水平线标签-->
<%                                                        //PHP 开始标记
    echo "利用< %  % >标记";                              //输出
%>                                                        <!--PHP 结束标记-->
    <hr>                                                  <!--水平线标签-->
        <script language="php">                           //php 开始标记
            echo  "利用 Script 的写法";                   //输出
        </script>                                         <!--php 结束标记-->
    </body>                                               <!--body 标签结束-->
</html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-2.php，查看其运行结果，即可看到示例代码的运行结果如图 4-3 所示。

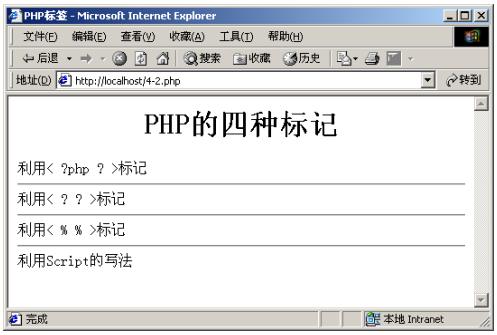

图 4-3　PHP 嵌入 HTML

【代码解析】代码 07～09 行是利用“<?php”、“?>”标记的 PHP 程序，第 10、14、18 行是一个 HTML 标签，在浏览器上显示一条横线。第 11～13 行和第 19～21 行是分别利用“<?”、“?>”和“<script language="php"”、“</script>”标记的 PHP 程序。

注意：代码 4-2 中，在“<”和“？ ”之间使用了空格，因此其不能被识别为标记。

4.1.3　标识符

标识符（identifier）即是一个事物的名字，就像每个人都要有自己的名字。标识符的命名要满足以下规则：

- 标识符可以由一个或多个字符组成，必须以字母或下画线开头。此外，标识符只能由字母、数字、下画线字符和从 127 到 255 的其他 ASCII 字符组成。
- 标识符区分大小写，函数例外。因此，变量$price 不同于变量$Price。
- 标识符名不能与任何 PHP 预定义关键字相同。
- 变量名可以与函数名相同。

下面举一些合法标识符和一些非法标识符的例子，如表 4-1 所示。

表 4-1　标识符

合　法	非　法
my_function	this&that
size	!counter
_someword	4ward

4.1.4 注释

PHP 注释的作用是解释其代码的功能，其注释风格和 HTML 不一样，它有两种注释形式，分别是单行注释和多行注释。单行注释如下所示。

```
// 注释内容......
# 注释内容......
```

多行注释如下所示。

```
/*
注释内容
......
*/
```

提示：建议读者在编写程序时养成良好的注释习惯，写好一段程序很容易忘记其内容，做好注释可以方便以后查找。

【范例 4-3】输出“使用//注释单行”，“使用/*　　*/的多行注释”，“使用#只能注释单行”的三种注释方法，并进行单行和多行注释。程序如示例代码 4-3 所示。

示例代码 4-3

```
01  <html>                                          <!-- html 标签开始-->
02      <head>                                      <!--头标记开始-->
03          <title>PHP 注释</title>                 <!--文件标题-->
04      </head>                                     <!--头文件结束-->
05      <body>                                      <!-- body 标签开始-->
06  <?php                                           //PHP 开始标记
07      echo "使用//注释单行<br>";                  //单行注释
08      /* 这是一个
09                多行注释 */
10      echo "使用/*    */的多行注释<br>";
11      echo "This is yet another test<br>";        //输出
12      echo '使用#只能注释单行';                   #单行注释
13  ?>                                              <!--PHP 结束标记-->
14      </body>                                     <!-- body 标签结束-->
15  </html>                                         <!-- html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-3.php，查看其运行结果，即可看到示例代码的运行结果如图 4-4 所示。

图 4-4　PHP 注释

【代码解析】程序第 08 行和第 09 行采用多行注释，程序第 07 行和第 11 行采用单行注释，echo 后面引号内的是输出内容。

注意：多行注释内容不可嵌套，如果使用形如“/*……/*……*/……*/”的注释将会产生错误。

4.1.5 echo 输出命令

PHP 并不是浏览器可以直接执行的语言，即浏览器无法解释 PHP 的内容。要使浏览器显示 PHP 所要表达的内容，必须通过 PHP 提供的几个输出命令，PHP 提供了 echo()和 print()来实现输出。前面的程序已多次使用 echo()语句，其基本的结构形式为：

```
viod echo (string argument1 [,….string argument])
```

echo()语句经常用于输出简单句，其形式简单。echo()语句不能用在复杂的表达式中，因为其返回一个静态值，其简单用法如下：

```
echo "变量 a 的值为：$a"
```

通过上例可以输出变量 a 的值。

4.1.6 print 输出命令

PHP 除了支持 echo()输出命令，还提供了如 print()输出命令，用法和 echo 输出命令基本相同。其基本的结构形式为：

```
boolean print (argument)
```

print()语句负责为用户提供反馈，能显示原始字符串和变量。能提供同 echo()语句相同的功能，而且比 echo()语句的功能更强大。

【范例 4-4】在网页中，经常要对变量的值进行输出，将 5 赋给变量$a，使用 print()语句和 echo()输出“PHP 基础教程”和变量 a 的值，并比较两者的用法。程序如示例代码 4-4 所示。

示例代码 4-4

```
<html>                                          <!--html 标签开始-->
    <head>                                      <!--头标记开始-->
        <title>PHP 输出命令</title>              <!--文件标题-->
    </head>                                     <!--头文件结束-->
    <body>                                      <!-- body 标签开始-->
<?php                                           //PHP 开始标记
    echo  "PHP 基础教程。<br>";                 //输出
    print "PHP 基础教程。<br>";                 //输出
    print ("PHP 基础教程。<br>");               //输出
    $a=5;                                       //定义变量
    echo "变量 a 的值为$a<br>";                 //输出命令
    print "变量 a 的值为$a<br>";                 //输出命令
    echo '变量 a 的值为$a<br>';                  //输出命令
    print '变量 a 的值为$a<br>';                 //输出命令
?>                                              <!--PHP 结束标记-->
    </body>                                     <!-- body 标签结束-->
</html>                                         <!-- html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-4.php，查看其运行结果，即可看到示例代码的运行结果如图 4-5 所示。

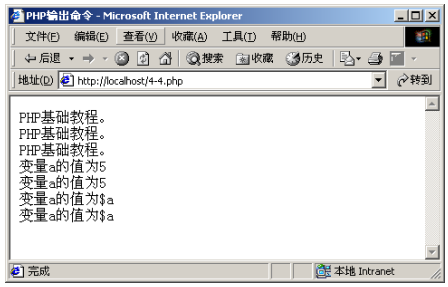

图 4-5　输出语句比较

【代码解析】代码第 10 行给一个变量$a 赋值，第 11～14 行对变量值进行输出，但采用单引号时对引号内的值不识别，直接输出，故不能正常输出 a 的值。

注意：PHP 是一门可嵌入式语言，但也是门独立的语言，前面可以不加 HTML 语言。为节省篇幅，在今后的章节中，一般不在 HTML 中嵌入，而使用独立的 PHP 语言。

4.2　数据类型

数据类型用来区分不同的数据，指定了什么样的数据能够在程序中进行存储和操作，数据类型就像一个容器，规定里面存放液体、固体或气体。由于数据在存储时所需要的容量各不相同，不同的数据就必须要分配不同大小的内存空间来存储。PHP 包含多种数据类型，下面将分别介绍。

4.2.1　布尔型

布尔型也就是逻辑型，支持逻辑运算。它是 PHP4 中新增的类型，常在判断中运用，其值只有两个：True（真）和 False（假）。其值可以直接设置，形式如下：

```
$a=true;                    //$x 设置为真
$b=false;                   //$y 设置为假
```

除了像上面直接设置之外，还可以强迫其他数据类型转换成布尔型。因为布尔型只有真值与假值，在下列情况下，其值将会被设置为 false。

```
布尔值=false
整数值=0
浮点数=0.0
空字符串
空数组
没有变量成员的对象。
变量值为空
NULL
```

例如：

```
$a= false;                  //$x 设置为假
$b=NULL;                    //$y 设置为假
$c=NULL;                    //$y 设置为假
```

除了上述情况以外，其余情况值将会被转换为 true，一般来说，1 为 true，0 为 false。这些转换将在以后的判断语句中经常遇到。

4.2.2 整型

整型即是整数，用 integer 表示。PHP 整型值可以以十进制、十六进制或八进制的方式表示数值。整型数的支持字长和平台有关，在 32 位的操作系统中，PHP 中整型的有效范围为-2,147,483,648~+2,147,483,647。

其整数可用以下三种形式表示。

- 十进制整数：例如 12、-50、0。
- 八进制整数：以 0 开头的数是八进制数。如 0123 表示八进制的 123，其值相当于十进制的 83。-011 表示八进制-11，其值相当于十进制的-9。
- 十六进制整数：以 0x 开头的数是 16 进制数。如 0x123，代表十六进制数 123，其值相当于十进制的 291。-0x12 等于十进制的-18。

【范例 4-5】分别设置变量 a、b、c、d、e 的值为 12345、-12345、12345（八进制）、12345（十六进制）、ABCD（十六进制），并输出变量的值，程序如示例代码 4-5 所示。

示例代码 4-5

```
01  <?                                                          //PHP 开始标记
02      $a = 12345;                                             //十进制数
03      $b = -12345;                                            //一个负数
04      $c = 012345;                                            //八进制数
05      $d = 0x12345;                                           //十六进制数
06      $e = 0XABCD;                                            //十六进制数
07      echo "a=$a <br> b=$b <br> c=$c <br> d=$d <br> e=$e";    //输出
08  ?>                                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-5.php，查看其运行结果，即可看到示例代码的运行结果如图 4-6 所示。

警告：在PHP中 不能 将整型设置为无符号型，否则将发生错误，因为PHP不支持无符号整数。

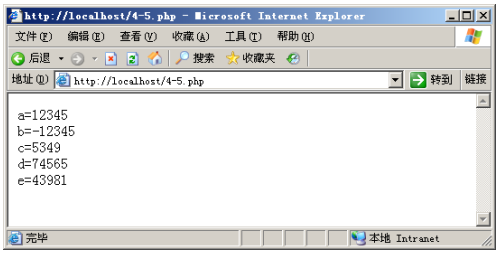

图 4-6 整型数据

【代码解析】程序第 02～06 行分别给变量$a、$b、$c、$d、$e 赋值，这是几种整数的表示方法。第 07 行输出各变量的值，输出的全是十进制数值。如果有其他进制的数，要先转化为十进制数才输出。

4.2.3 浮点型

浮点型主要用于表示带有小数的数值，有 float 和 double 两种形式。浮点型（也叫浮点数）可以用以下语法定义：

```
$a = 1.234;
$a= 1.2e3;
$a = 7E-10;
```

第一种是比较熟悉的表示方法，第二种是以指数的方法表示的。PHP 的浮点型类似于 C 语言中 double 型，在 32 位的操作系统中，有效范围是 1.7e-308~1.7e+308。浮点型变量显示时所用的十进制个数由 php.ini 文件中的 precision 定义，预定义值为 14，即浮点数最长为 14 个数字。

【范例 4-6】设置浮点型变量$a = 1.234，$b = -1.234，$c = 1.234e-2，$d =1.234e2，并对其进行输出，比较其结果的异同。程序如示例代码 4-6 所示。

示例代码 4-6

```
01  <?                                                          //PHP 开始标记
02      $a = 1.234;                                             //变量赋值
03      $b = -1.234;                                            //变量赋值
04      $c = 1.234e-2;                                          //变量赋值
05      $d = 1.234e2;                                           //变量赋值
06      echo "a=$a <br> b=$b <br> c=$c <br> d=$d <br>";       //输出变量
07  ?>                                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-6.php，查看其运行结果，即可看到示例代码的运行结果如图 4-7 所示。

【代码解析】程序第 02～05 行给变量赋予浮点型变量，第 06 行输出变量 a、b、c、d 的值。

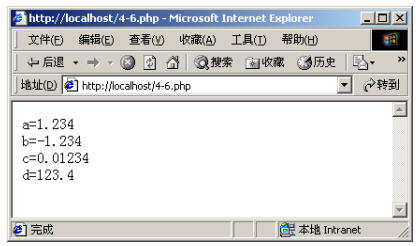

图 4-7　浮点型变量

4.2.4　数据类型的转换

将一个值转换为整型（integer），可以用（int）或（integer）强制转换。不过大多数情况下都不需要强制转换，因为当运算符、函数或流程控制需要一个 integer 参数时，值会自动转换。还可以通过函数 intval()来将一个值转换成整型。

【范例 4-7】在编写程序中，如果不注意数据类型的范围，有可能造成错误。将变量$large_number 的值先后赋予 1000000000、2147483648、80000000（八进制）、50000000000，利用 var_dump 查看其输出的值及其类型，分析输出的结果。程序如示例代码 4-7 所示。

示例代码 4-7

```
01  <?php                                                       //PHP 开始标记
02      $large_number=1000000000;                               //变量赋值
03      var_dump($large_number);                                //输出变量
04          //输出为：int(1000000000)
05      $large_number=2147483648;                               //变量赋值
06          //输出为：float(2147483648)
07      var_dump($large_number);                                //输出变量
08          //同样也适用于十六进制表示的整数
09      var_dump(0x80000000);                                   //输出变量
10          //输出为：float(2147483648)
11      $million = 1000000;                                     //变量赋值
12      $large_number = 50000*$million;                         //变量赋值
13          //输出为：float(50000000000)
14      var_dump($large_number);                                //输出变量
15  ?>                                                          //PHP 结束标记
```

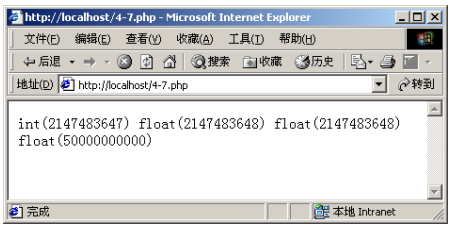

图 4-8　数据类型的转化

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-7.php，查看其运行结果，即可看到示例代码的运行结果如图 4-8 所示。

【代码解析】程序第 02 行定义一个整型变量，第 03 行用 var_dump()函数输出变量及类型。第 05 行定义的变量值超出了整型范围，所以输出浮点型。第 09 行判断十六进制的 80000000 的数据类型。第 12 行将变量$large_number 的值变为 50000000000。第 14 行判断变量$large_number 的数据类型。

注意：如果给定的数超出了 integer 的范围，将会被解释为 float；同样，如果执行的运算结果超出了 integer 范围，也会返回 float。

4.2.5　字符串

字符串就是包括在一对双引号或一对单引号之间的一串字符。引号必须匹配，以单引号开始，必须以单引号结束；以双引号开始，必须以双引号结尾。字符串内部也可能会出现单引号或双引号。下面是一组字符串的实例：

```
"there are many things"
"there are 'many things' "
'there are"many things" '
```

上面是三组字符串，从上面可以看出，字符串是包括在一对单引号或双引号之内的一串字符，字符之中也可以含有单引号或双引号。另外，字符串中有时需要使用特殊字符，则可以用反斜杠（\）表示，常见的特殊字符及其功能如表 4-2 所示。

表 4-2　特殊字符

字 符 形 式	功　能
\n	换行并归 0
\br	换行
\t	跳格
\'	单引号
\"	双引号
\$	$符号
\r	回车

【范例 4-8】分别给变量 a、b、c 赋予 Hello\n、PHP\r
、How are you，含有特殊字符的字符串，输出变量，查看输出结果的变化。程序如示例代码 4-8 所示。

示例代码 4-8

```
01  <?                                                    //PHP 开始标记
02      $a = "Hello\n";                                   //变量赋值
03      $b = "PHP\r <br>";                                //变量赋值
04      $c = "How are you.";                              //变量赋值
05      echo $a, $b, $c;                                  //输出变量
06      echo "<br>";                                      //输出变量
07      echo "\$a=$a \t \$b=$b \t \$c=$c";                //输出变量
08  ?>                                                    //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-8.php，查看其运行结果，即可看到示例代码的运行结果如图 4-9 所示。

【代码解析】代码第 02～04 行给变量赋值，第 05 行和第 07 行输出这些变量。变量值是含有特殊字符的字符串，其含义在表 4-1 中已列出，可以通过源文件查看其编译后的效果，编译后的结果如图 4-10 所示。

图 4-9　字符串类型

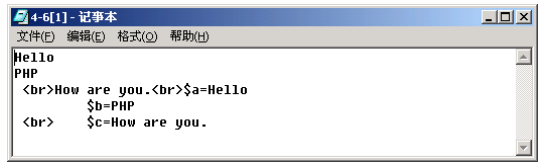

图 4-10　查看源文件

提示：这里只简单介绍一下字符串，有关字符串的其他操作将在第 8 章中详细讲述。

4.2.6　NULL 类型

NULL 是一个比较特殊的类型，一个变量被设置为 NULL，就表示这个变量没有值。NULL 类型唯一可能的值就是 NULL。在下列情况下一个变量被认为是 NULL：

- 被赋值为 NULL。
- 尚未被赋值。
- 被 unset()。

NULL 类型只有一个值，就是大小写敏感的关键字 NULL。下面的程序没有任何输出：

```
01  <?php                                  //PHP 开始标记
02  $var1 = NULL;                          //变量赋值
03  $var2 = 5;                             //变量赋值
04  unset($var2);                          //清除变量值
05  echo "$var1 $var2 $var3";              //输出结果
06  ?>                                     //PHP 结束标记
```

因为程序第 02 行$var1 被设置为 NULL，$var1 的值为空；第 03 行 unset()函数清除变量内容，$var2 为空；第 05 行$var3 没有被定义，也为空，所以没有任何输出。

4.3　变量

变量在程序中经常使用，简单来讲变量就像一个容器，先给这个容器取上名字，然后利用赋值运算符给这个容器装上东西。下面详细介绍其具体的解释和应用。

4.3.1　变量名

一个程序中因为各种计算的不同而需要不同个数的变量，此时就必须对每一个不同的变量给予不同的名称以示区别。变量名必须遵循以下规则：

- 变量必须以$符号开头。
- 第二个字符必须是字母或下画线。

- 以后可以是字符数字等的任意组合。

下面给出几个合法的变量名的示例：

```
$price  $my_string  $name  $_year  $no  $No
```

为了让程序较易读懂，变量名的定义一般都有其特定含义，例如$_year 一般是和年份有关的变量名。

4.3.2 定义和赋值

变量是一个数据存储空间的表示，将数据指定给变量，就是将数据存储至对应的内存空间，调用变量，就是将对应的内存空间的数据取出来使用，声明变量就是申请一个内存空间。

PHP 中变量能够被赋予以下几种类型：

- 整型；
- 浮点型；
- 字符串；
- 布尔型；
- 对象；
- 数组。

其形式如下所示。

```
01  $a=1;
02  $b=1.1;
03  $c="Hello";
```

第 01 行设置变量$a 的值为 1，数据类型为整数类型。第 02 行设置变量$b 的初始值为 1.1，数据类型为符点型。第 03 行为字符串变量。所以，变量的初始值不但决定变量的内容，同时还决定数据的变量类型。上述代码直接设置变量的初始值，这种设置初始值还可以通过表达式来实现。下面的实例通过表达式给变量赋值：

```
01  $a=2;
02  $b=6;
03  $c=$a*$b+2;
04  $d=$e;
```

代码第 01，02 行设置变量 a，b 的值，代码第 03 行通过表达式给变量 c 赋值，第 04 行将变量 e 的值赋给变量 d。

4.3.3 输出变量

变量的输出一般利用 PHP 提供的输出语句，通过变量名即可直接调用其变量值，其具体用法通过下面的实例来讲解。

【范例 4-9】定义变量$a、$b、$c 的值分别为 1、1.1、Hello。给变量$d、$e 分别赋予表达式 1+6、 $d+1。输出其运算结果，比较两种输出变量的方式。程序如示例代码 4-9 所示。

示例代码 4-9

```
01  <?php                                                  //PHP 开始标记
02      $a=1;                                              //变量赋值
03      $b=1.1;                                            //变量赋值
04      $c="Hello";                                        //变量赋值
05      echo  "第一种方式设置变量值<br>";                  //输出变量
06      echo "a=$a <br> b=$b <br> c=$c <br>";             //输出变量
07      $a=2;                                              //变量赋值
```

```
08      $b=6;                                          //变量赋值
09      $c=$a*$b+2;                                    //变量赋值
10      $d=1+6;                                        //变量赋值
11      $e=$d+1;                                       //变量赋值
12      echo "第二种方式设置变量值<br>";                //输出变量
13      echo "c=$c <br>d=$d <br> e=$e";                //输出变量
14  ?>                                                 //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-9.php，查看其运行结果如图 4-11 所示。

【代码解析】代码第 02～06 行是简单的变量赋值和变量输出，第 07～11 行是含有逻辑运算的变量赋值，第 13 行输出变量的值。

提示：输出变量的值时，不要对变量名加引号，如 echo $a。

4.3.4 变量类型自动转换

变量的数据类型是指其数据值的数据类型，其支持 4.2 节提到的 4 种变量类型和 2 种复合类型。在 PHP 中，变量类型在变量赋值时便会自动确定，在以后的使用中，其类型也可随时根据需要强制改变。变量的自动转换主要依靠操作符，如用“=”可自动转换为字符，用“+=”可转化为数字。在将字符串转化为数字时，如果字符串中含有“e”“E”“.”或“、”则其将转化为浮点型数据，否则转化为整型。

【范例 4-10】给变量$var 赋予字符串 1，然后将其强制加 1，再加上浮点数 1.1，最后利用 array()函数将其转化为数组，通过 gettype()函数查看其数据类型的变化，程序如示例代码 4-10 所示（gettype—获取变量的类型）。

示例代码 4-10

```
01  <?php                                              //PHP 开始标记
02      $var="1";                                      //变量赋值
03      echo "$var is a ".gettype($var)."<br>";        //输出变量类型
04      $var+=1;                                       //类型转换
05      echo  "$var is a ".gettype($var)."<br>";       //输出
06      $var+=1.1;                                     //变量赋值
07      echo  "$var is a ".gettype($var)."<br>";       //输出
08      $var=array($var);                              //转化为数组
09      echo  "$var is a ".gettype($var)."<br>";       //输出
10  ?>                                                 //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-10.php，查看其运行结果，即可看到示例代码的运行结果如图 4-12 所示。

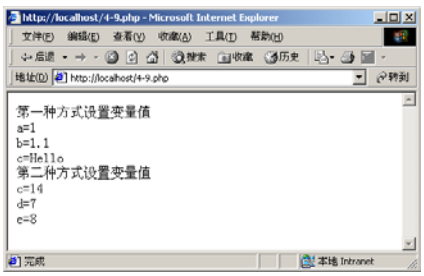

图 4-11　变量输出

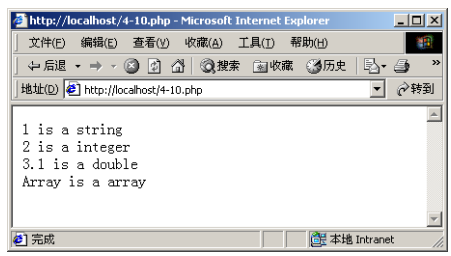

图 4-12　类型转换

【代码解析】程序第 02 行定义一个变量，变量值为字符串。第 03 行输出字符串的类型，gettype()函数是获得字符串类型的函数。第 04 行将原变量的值加 1，获得新的变量，这时变量的值被强制转化为整型，得到了整型变量。同理，第 06 行和第 08 行都进行了强制转换，第 10 行是将变量转化为数组的形式，arr 是用来转化为数组的。

注意：强制将某个变量转换到指定的类型时，可以对该变量进行“cast”操作，或者使用函数“settype()”。

4.3.5 默认变量

在上网时，经常会发现自己使用的浏览器名称、IP 地址等关于自己的信息会显示在浏览器上。而对方网站借助通信协议取得这些信息，这些信息可以轻而易举地显示在网页上。

在 PHP 中也有这样的默认变量，在介绍默认变量之前先说明一下 PHP 的参数设置。在 php.ini 中有一个参数设置 register_globls，它用于设置环境变量、GET 变量、POST 变量、Cookie 变量，以及服务器变量是否在程序中可以直接使用。若其值为 on 就表示这个变量可以直接使用。如果其值是 off，则程序不能直接使用，但可以借助$_ENV、$_GET、$POST、$_Cookie 和 $_SERVER 数组取得变量内容。例如，使用$_SERVER 数组变量可以获得服务器上的相关信息。

如表 4-3 列出的是$_SERVER 数组中的变量包含的内容。

表 4-3 $_SERVER 数组内变量

变 量 名	意 义
PHP_SELF	目前执行的文件名称
SERVER_NAME	服务器名
SERVER_SOFTWARE	服务器使用的软件
DOCUMENT_ROOT	文档的根目录
HTTP_USER_AGENT	用户相关信息
REMOTE_ADDR	远程用户的地址
REMOTE_PORT	远程用户的连接端口

提示：在取得变量内容时，以变量名称为数组指针即可取得信息。例如，SERVER_NAME 可以用$_SERVER[‘SERVER_NAME’]取得信息。

【范例 4-11】输出一些默认变量 PHP_SELF，SERVER_NAME，SERVER_SOFTWARE，DOCUMENT_ROOT，HTTP_USER_AGENT，REMOTE_ADDR，REMOTE_PORT 的值。程序如示例代码 4-11 所示。

示例代码 4-11

```
01  <?php                                                                  //PHP 开始标记
02      echo "目前执行的文件名称". $_SERVER['PHP_SELF']."<br>";             //输出变量值
03      echo "服务器名" . $_SERVER['SERVER_NAME']."<br>";                   //输出变量值
04      echo "服务器软件".$_SERVER['SERVER_SOFTWARE']."<br>";               //输出变量值
05      echo "文档的根目录" . $_SERVER['DOCUMENT_ROOT']."<br>";             //输出变量值
06      echo "用户相关信息" . $_SERVER['HTTP_USER_AGENT']."<br>";           //输出变量值
07      echo "远程用户的地址". $_SERVER['REMOTE_ADDR']."<br>";              //输出变量值
08      echo "远程用户的连接端口". $_SERVER['REMOTE_PORT']                  //输出变量值
09  ?>                                                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-11.php，查看其运行结果，如图 4-13 所示。

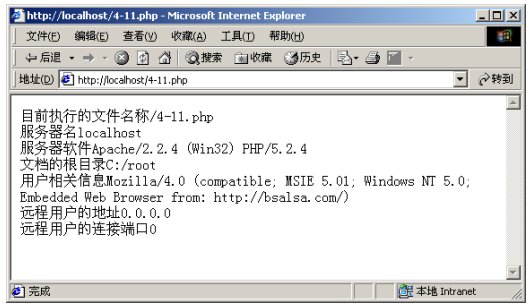

图 4-13　默认变量

【代码解析】上述代码中的默认变量的含义可以从表 4-3 中直接查找，代码分别输出这些变量所代表的值。

4.4　常量

常量就是一个简单值的标识符（名字），代替一个有意义的值。常量如同其名称所暗示的，在程序执行期间该值不能改变。PHP 的常量包含两种：一种是默认常量，另一种是自定义常量。

4.4.1　默认常量

PHP 本身提供了很多预先定义好的常量供设计者使用，称之为默认常量。如表 4-4 列出了常用的默认常量和其含义。

表 4-4　默认常量

默认常量名	含　义
_ FILE_	当前正在分析着的脚本的文件名。如果用在分析一个被 include 或 require 包含的文件，则该常数给出的是 include 所包含的文件名，而不是当前文件名
_ LINE _	当前正在分析着的行在脚本中的行数。如果用在一个被 include 或 require 包含的文件中，则该常数给出的是（行）在被 include 包含的文件中的位置
PHP_VERSION	一个描述当前用着的 PHP 处理器的版本的字符串
PHP_OS	正在运行本 PHP 处理器的操作系统的名字
TRUE	真值
FALSE	假值
E_ERROR	指示一个不可恢复的语法错误
E_WARNING	指示如下的一种状态： PHP 知道某处出错了，但仍可以继续运行；这些错误能被脚本自身捕获
E_PARSE	PHP 在脚本的一个语病中被阻塞了，不可恢复
E_NOTICE	出现了可能是一个错误也可能不是的情况；（这种情况下）运行会继续

这些系统常量可以帮助我们很好地了解系统当前的情况，而且这些常量可以随时调用。例如_FILE_预定义常量，可以直接得到目前操作的文件名。

【范例 4-12】通过默认常量 PHP_OS 和 PHP_VERSION，查看操作系统和所使用的 PHP 版本，其程序如示例代码 4-12 所示。

示例代码 4-12

```
01  <?php
02      echo "你的系统是：".PHP_OS."<p>";
03      echo "目前使用的 PHP 版本是：".PHP_VERSION."<P>";
04  ?>
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-12.php，查看其运行结果，即可看到其结果如图 4-14 所示。

【代码解析】代码第 02 行利用默认常量显示其当前的操作系统，第 03 行查看其使用的 PHP 版本。

4.4.2 自定义常量

预定义的常量常常满足不了需要，PHP 还提供了 define()函数自行定义所需要的常量，其结构形式如下：

```
define('name',number)
```

第一个参数为常量名，第二个参数为赋给常量的值，值可以是数值，也可以是字符串。

【范例 4-13】将值 3.14159 赋给一个常量 PI，并输出该常量的值。程序如示例代码 4-13 所示。

示例代码 4-13

```
01  <?                                          //PHP 开始标记
02      define("PI","赋 3.14159 给 PI");        //定义常量
03      echo PI;                                //输出常量值
04  ?>                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-13.php，查看其运行结果，即可看到示例代码的运行结果如图 4-15 所示。

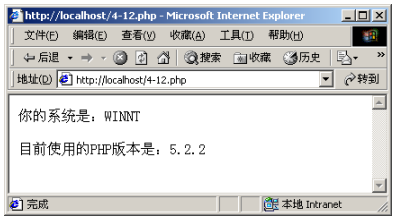

图 4-14 默认常量

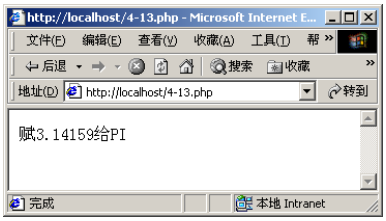

图 4-15 定义常量

【代码解析】代码第 02 行定义一个常量 PI，将一个字符串赋给了常量 PI，第 03 行输出常量值。

注意：常量前没有$符号。

4.5 运算符及表达式

前面介绍了几种数据类型，下面介绍如何利用运算符进行各种运算。运算符同数据类型结合起来，就构成了表达式。运算符种类很多，主要有算术运算符、赋值运算符、字符串运算符、位运算符等，下面分别介绍这些运算符。

4.5.1 算术运算符

算术运算符用来处理四则运算的符号，这是最简单、也是最常用的符号，尤其是对数字的处理，几乎都会使用到算术运算符号，其符号和意义如表 4-5 所示。

表 4-5 算术运算符

符　号	意　义
+	加法运算
−	减法运算
*	乘法运算
/	除法运算
%	取余运算

【范例 4-14】将 5 和 3 分别赋予变量 a，b，利用算术运算符求得变量$a 和变量$b 之和、差、乘积、商和余数。程序如示例代码 4-14 所示。

示例代码 4-14

```
01  <?                                          //PHP 开始标记
02      $a=5;                                   //定义变量
03      $b=3;                                   //定义变量
04      echo $a."+".$b."=";                     //输出表达式
05      echo $a+$b."<br>";                      //输出结果
06      echo $a."-".$b."=";                     //输出表达式
07      echo $a-$b."<br>";                      //输出计算结果
08      echo $a."*".$b."=".$a*$b."<br>";        //输出变量值
09      echo $a."/".$b."=";                     //输出表达式
10      echo $a/$b."<br>";                      //输出计算结果
11      echo $a."%".$b."=";                     //输出表达式
12      echo $a%$b;                             //输出计算结果
13  ?>                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-14.php，查看运行结果，即可看到其结果如图 4-16 所示。

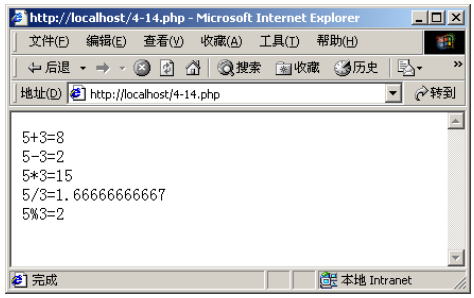

图 4-16 算术运算符

【代码解析】程序第 02 行和第 03 行分别定义两个变量，第 04～10 行是利用算术运算符，求得两变量之间的关系，并输出结果。

提示：利用“ / ”时，无论是整型或是浮点型除法，计算结果都以浮点型表示。

4.5.2 赋值运算符

基本的赋值运算符是“=”，一开始可能会以为它是“等于”，其实不是的，实际上意味着把右边表达式的值赋给左边的运算数。

除了基本赋值运算符外，还有一些复合运算符，常见的复合运算的符号及其意义如表 4-6 所示。

比如，“$a*=$b”等价于“$a=$a*$b”，其他赋值运算的等价关系可依此类推，复合运算符使得运算式更加简洁。

表 4-6 复合运算符

符 号	意 义
+=	将左边的值加上右边的值赋给左边
-=	将左边的值减去右边的值赋给左边
*=	将左边的值乘以右边的值赋给左边
/=	将左边的值除以右边的值赋给左边
%=	将左边的值对右边取余数赋给左边
.=	将左边的字符串连接到右边

【范例 4-15】给变量 a，b 分别赋予 3 和 Hello，利用赋值运算符使变量 a，b 输出“8”和“Hello PHP”。程序如示例代码 4-15 所示。

示例代码 4-15

```
01  <?php                          //PHP 开始标记
02      $a = 3;                    //定义变量
03      $a += 5;                   // $a = $a + 5;
04      $b = "Hello ";             //定义变量
05      $b .= "PHP";               // "Hello PHP !";
06      echo "a=$a<br>";           //输出结果
07      echo "b=$b";               //输出变量值
08  ?>                             //PHP 结束标记
```

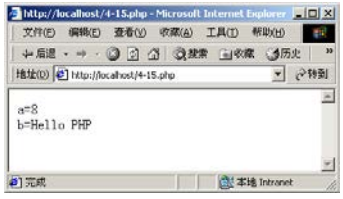

图 4-17 赋值运算符

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-15.php，查看运行结果，即可看到其结果如图 4-17 所示。

【代码解析】代码第 03 行、第 05 行利用赋值运算符对变量$a、$b 赋值，相当于$a=$a+5，$b=Hello PHP。第 06、07 行输出变量。

4.5.3 自增自减运算符

PHP 提供了自增运算符（++）和自减运算符（--）。自动增量运算符的功能很简单，就是让它的操作数自动加 1。相反，自动减量运算符就是把它的操作数自动减 1。如果设一个变量$a，运用自增或自减运算符的形式共有下列 4 种：

```
$a++;
++$a;
$a--;
--$a;
```

在一般单独的表达式中，$a++和++$a 所实现的效果是一样的，都使变量$a 的值加 1。但在一些复杂的表达式中，特别是赋值表达式中，自增和自减运算符的位置不一样会影响计算结

果。例如现有表达式$b=$a++与表达式$b=++$a，变量$a 的初值是 1，执行前后变量的值变化情况如图 4-18 所示。

图 4-18　自增运算符

自减运算符同上面的自增运算符一样，具有相同的运算规律，只是数值减 1。

【范例 4-16】将 5 赋给变量 a，输出$a++的值，再查看变量$a 的值。然后在变量 a 的值赋予 5，输出++$a 的值，再查看变量$a 的值。同样方法计算自减运算，比较自增或自减 4 种运算符的差异。程序如示例代码 4-16 所示。

示例代码 4-16

```
<?php                                                    //PHP 开始标记
    echo "<h3>自增 a++ </h3>";                           //输出标题字
    $a = 5;                                              //变量赋值
    echo "Should be 5: " . $a++ . "<br />\n";            //自增变量
    echo "Should be 6: " . $a . "<br />\n";              //输出变量
    echo "<h3>自增++a </h3>";                            //输出标题字
    $a = 5;                                              //变量赋值
    echo "Should be 6: " . ++$a . "<br />\n";            //自增变量
    echo "Should be 6: " . $a . "<br />\n";              //输出变量
    echo "<h3>自减 a-- </h3>";                           //输出标题字
    $a = 5;                                              //变量赋值
    echo "Should be 5: " . $a-- . "<br />\n";            //自减变量
    echo "Should be 4: " . $a . "<br />\n";              //变量输出
    echo "<h3>自减--a </h3>";                            //输出标题字
    $a = 5;                                              //变量赋值
    echo "Should be 4: " . --$a . "<br />\n";            //自减变量
    echo "Should be 4: " . $a . "<br />\n";              //输出变量
?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-16.php，查看运行结果，即可看到其结果如图 4-19 所示。

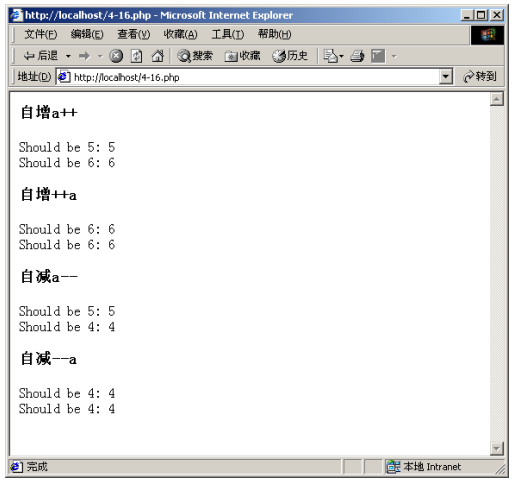

图 4-19　自增自减运算

【代码解析】代码第 03 行给变量$a 赋值，第 04 行输出变量$a 的值，并加 1。第 05 行输出变量$a 的值，此时变量$a 的值经过了加 1 变为 6。第 08 行对变量$a 加 1，然后输出 a 的值。自减运算符同自增相同。

提示：使用自增自减运算可以简化输入，提高运算的效率，提高程序的可读性。

4.5.4 位运算符

位运算是以二进制为单位的算法，即把现有的数转换成二进制数来进行运算，主要的位运算符如表 4-7 所示。

表 4-7 位运算符

符 号	意 义
&	按位与
|	按位或
^	按位异或
<<	按位左移
>>	按位右移
~	按位取反

计算机内部都是 0 和 1 的二进制，位运算即是利用二进制做运算的，具体用法通过下面的范例来讲解。

【范例 4-17】将变量 a、b、c 分别赋予 12、9、2，进行按位与、按位或、按位左移运算，其程序如示例代码 4-17 所示。

示例代码 4-17

```
01  <?php                                  //PHP 开始标记
02      $a=12;                             //变量赋值
03      $b=9;                              //变量赋值
04      $c=2;                              //变量赋值
05      echo "a=$a b=$b c=$c";             //输出变量
06      echo "<br>\$a & \$b =";            //输出结果
07      echo $a & $b;                      //按位与
08      echo "<br>\$a | \$b =";            //输出结果
09      echo $a | $b;                      //按位或
10      echo "<br>\$a<<\$c=";              //输出结果
11      echo $a<<$c;                       //按位异或
12  ?>                                     //PHP 结束标记
```

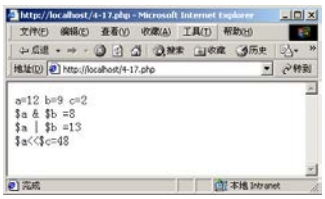

图 4-20 位运算

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-17.php，查看运行结果，即可看到其结果如图 4-20 所示。

【代码解析】代码 02～04 行给变量赋值，第 07 行输出按位与的值，第 09 行输出按位或的值，第 11 行输出按位异或的值。

提示：位运算的计算要按二进制进行计算，首先把变量由十进制数转化为二进制数，例如 12 转化为 1100，即 $2^0\times0+2^1\times0+2^2\times1+2^3\times1=12$。同理 9 的二进制数为 1001，通过位运算，即可得到代码的正确计算结果。

4.5.5 逻辑运算符

逻辑运算符主要有以下 6 种，其运算时只有真（true）及假（false）两个值，即无论哪种逻辑运算结果只有真和假。如表 4-8 所示，列出了主要的 6 种运算符及其用法。

表 4-8 逻辑运算符

操 作 符	用 法	说 明
and（与）	$a and $b	$a 和$b 同时为真，则结果为真
&&（与）	$a && $b	$a 和$b 同时为真，则结果为真
or（或）	$a or $b	$a 或$b 有一个为真，则结果为真
‖（或）	$a ‖ $b	$a 或$b 有一个为真，则结果为真
xor（异或）	$a xor $b	$a 和$b 不同时为真，则结果为真
!（非）	! $a	$a 为假，则结果为真

其计算表达式可表示为：

```
0 and 0=0  0 and 1=0  1 and 0=0  1 and 1=1
0 or 0=0  0 or 1=1  1 or 0=1  1 or 1=1
0 xor 1=1  1 xor 0=1  1 xor 1=0
!0=1  !1=0
```

符号 0 代表假，1 代表真。

4.5.6 关系运算符

PHP 的关系运算符用来表达数据之间大小关系的运算符，主要有以下 6 种运算符。

表 4-9 关系运算符

操 作 符	用 法	说 明
==（等于）	$a == $b	$a 等于$b 结果为真
!=（不等于）	$a != $b	$a 不等于$b 结果为真
<（小于）	$a < $b	$a 小于$结果为真
>（大于）	$a > $b	$a 大于$b 结果为真
<=（小于等于）	$a <= $b	$a 小于或等于$b 结果为真
>=（大于等于）	$a >= $b	$a 大于或等于$b 结果为真

【范例 4-18】利用关系运算判断 0 和 a 的关系，并输出判断结果。其程序如示例代码 4-18 所示。

示例代码 4-18

```
<?php                                    //PHP 开始标记
    var_dump(0 == "a");                  // 0 == 0 -> true
    var_dump("1" == "01");               // 1 == 1 -> true
    switch ("a") {                       //选择语句
        case 0:                          //选择情况
        echo "0";                        //输出结果
```

```
07            break;                                    //跳出选择语句
08            case "a":                                 //选择情况
09            echo "a";                                 //输出语句
10            break;                                    //跳出选择语句
11        }
12    ?>                                                //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-21.php，查看运行结果，即可看到其结果如图 4-21 所示。

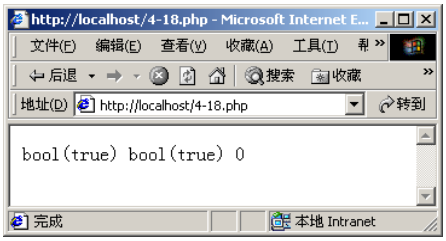

图 4-21　关系运算

【代码解析】程序第 02～03 行利用函数 var_dump()起判断作用，返回真。第 04 行是一个 switch 语句，判断 a 的值。switch 语句将在第 5 章中详细介绍。

4.5.7　其他运算符

除了上述介绍的几种运算符之外，还有很多常会遇到的运算符。例如，字符串的连接、抑制错误信息等运算符，如表 4-10 所示，列出了比较常用的运算符。

表 4-10　其他运算符

符　　号	含　　义
&	取地址
@	不显示错误信息
?:	三目运算
$	变量
.	连接字符串
,	逗号运算符
->	引用对象的方法和属性
=>	数组赋值

在表达式中比较特殊的是三目运算符“？:”，其结构形式为：

```
(expr1)?(expr2):(expr3);
```

如果表达式 expr1 的运算结果为 True，则执行 expr2。如果表达式 expr1 的计算结果为假，则执行表达式 expr3。其运算法则类似于后面讲的 if…else 循环语句。另外，这种表达式可以使程序更简洁，更有效率。

```
01  $a = array("a" => "apple", "b" => "banana");
02  $b = array("a" => "pear", "b" => "strawberry", "c" => "cherry");
```

上面的实例利用运算符=>给数组变量赋值。第 01 行分别将“apple”、“banana”赋给变量 a，b 构成数组元素，第 02 行同第 01 行类似。

4.5.8 运算符优先级

前面提到了大量的运算符，当这些运算符同时出现在同一个表达式时，就不得不考虑其运算的先后和优先级的问题了。优先级，即是结合的先后顺序问题，一般谁的优先级高就先算谁。如果不考虑运算的先后，很可能会造成错误。运算符的优先级决定着运算的次序，如表 4-11 列出了几种常用运算符的优先级。在以后的运用中，如果是复杂的混合运算，要特别注意运算符的优先级问题。如果不注意，可能会造成很大错误。

表 4-11　运算符优先级

优 先 级	结合方向	运 算 符	附加信息
1	非结合	new	new
2	左	[	array()
3	非结合	++ --	递增 / 递减运算符
4	非结合	! ~ - （int） （float） （string） （array） （object） @	类型
5	左	* / %	算数运算符
6	左	+ - .	算数运算符和字符串运算符
7	左	<< >>	位运算符
8	非结合	< <= > >=	比较运算符
9	非结合	== != === !==	比较运算符
10	左	&	位运算符和引用
11	左	^	位运算符
12	左	|	位运算符
13	左	&&	逻辑运算符
14	左	||	逻辑运算符
15	左	? :	三元运算符
16	右	= += -= *= /= .= %= &= |= ^= <<= >>=	赋值运算符
17	左	and	逻辑运算符
18	左	xor	逻辑运算符
19	左	or	逻辑运算符
20	左	,	多处用到

提示：左表示表达式从左向右求值，右表达式则相反。

【范例 4-19】利用混合运算分别将变量 a、b 赋予 3 * 3 % 5 和 true ? 0 : true ? 1:2，查看计算结果，再将变量 a、b 的值赋予 1、2，再利用关系式将其值变为 5。其程序如示例代码 4-19 所示。

示例代码 4-19

```
<?php                                   //PHP 开始标记
    $a = 3 * 3 % 5;                     //（3 * 3） % 5 = 4
    $b = true ? 0 : true ? 1 : 2;       //（true ? 0 : true） ? 1 : 2 = 2
    echo $a;                            //输出
    echo "<br>";                        //输出换行
    echo $b;                            //输出变量 b 的值
    echo "<br>";                        //输出换行
    $a = 1;                             //设置变量
```

```
09      $b = 2;                          //设置变量
10      $a = $b += 3;                    // $a =（$b += 3） -> $a = 5, $b = 5
11      echo $a;                         //输出
12      echo "<br>";                     //输出
13      echo $b                          //输出
14  ?>                                   //PHP 结束标记
```

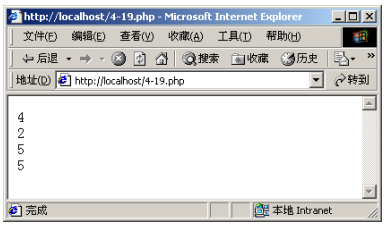

图 4-22　运算符优先级

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-19.php，查看运行结果，即可看到其结果如图 4-22 所示。

【代码解析】程序第 03 行为混合三目运算，表达式为 true，则执行表达式 3。表达式的值仍为 true，则执行另一组的表达式 3，其值为 2。第 10 行相当于$b=$b+3，$a=$b，所以表达式的值为 5。

4.5.9　表达式

表达式就是由操作数、运算符等所组成的合法序列。简单地说，变量或常量通过运算符连接后就形成了表达式。例如：

```
$a++
```

上式就为一个表达式，变量$a 同自增运算符结合到一起构成了自增表达式。表达式也可能很复杂，可以是很多运算符的结合。根据表达式中的运算符的不同，可以把表达式分为算术表达式、赋值表达式、位运算表达式、逻辑表达式、关系表达式等。其中赋值运算是运用较多的，而逻辑表达式、关系表达式也会在以后的章节中大量运用。

4.6　综合练习

1．在编写网页时，经常遇到一些数据问题，如数据记录，编写公式等，经常会用到数据类型的转换问题。下面对变量$var 进行数据类型的转换，给变量$var 赋予字符串 0，查看其数据类型。然后进行自增运算，数值加 1 后，再加上 float 型的 1.3，分别查看每次运算后的类型。

提示：变量的转换可以通过运算符来实现，也可以通过一些函数，强制转换。其程序如示例代码 4-20 所示。

示例代码 4-20

```
01  <?                                   //PHP 开始标记
02      $var = "0";                      //变量赋值
03      echo gettype($var)."<br>";       //输出“string”
04      echo "var 的值为:".$var."<br>";  //输出变量值
05      $var++;                          //自增运算
06      echo gettype($var)."<br>";       //整数
07      echo "var 的值为:".$var."<br>";  //输出变量值
08      $var+= 1;                        //加 1
09      echo gettype($var)."<br>";       //整数
10      echo "var 的值为:".$var."<br>";  //输出变量值
11      $var = $var+1.3;                 //类型转换
12      echo gettype($var)."<br>";       //双精度数
13      echo "var 的值为:".$var."<br>";  //输出变量值
14  ?>                                   ///PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-20.php，查看运行结果，即可看到其结果如图 4-23 所示。

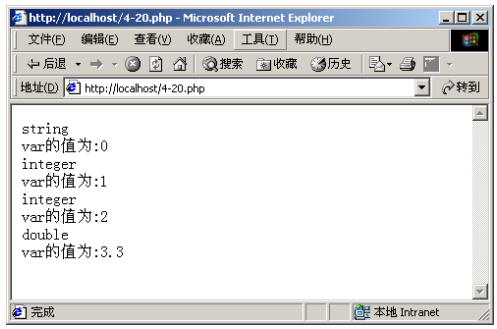

图 4-23　类型互换

2．给变量 a 赋予值 5，利用$b=$a 和$c=& $a 两种赋值方式给变量赋值，先将 b 的值加 1，再将 a 的值加 1，分别输出变量的值，比较其区别。

提示：先定义变量$a 的值，然后通过运算改变变量$b 的值，最后改变变量$a 的值。其程序如示例代码 4-21 所示。

示例代码 4-21

```
<?php                                                    //PHP 开始标记
  $a=5;                                                  //变量赋值
  $b=$a;                                                 //按值赋值
  $c=&$a;                                                //按址赋值
  echo "初始值：<br>";                                    //输出
  echo "\$a=$a <br> \$b=$b <br> \$c=$c <p>";             //输出变量值
  $b++;                                                  //自增运算
  echo "$b 加后：<br>";                                   //输出
  echo "\$a=$a <br> \$b=$b <br> \$c=$c <p>";             //输出变量值
  $a++;                                                  //自增运算
  $a++;                                                  //自增运算
  echo "$a 加后：<br>";                                   //输出
  echo "\$a=$a <br> \$b=$b <br> \$c=$c ";                //输出变量值
?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/4-21.php，查看运行结果，即可看到其结果如图 4-24 所示。

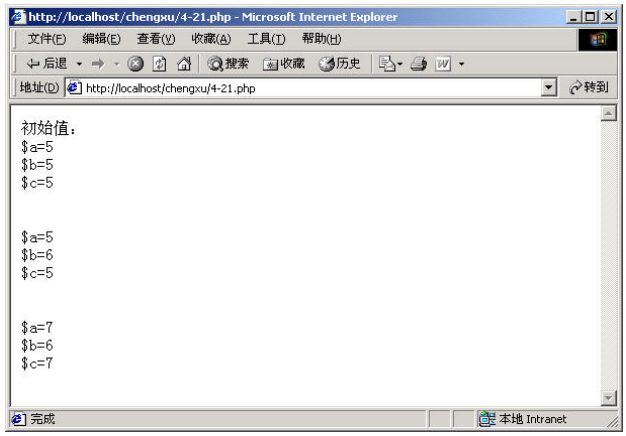

图 4-24　变量赋值

4.7 小结

本章是学习 PHP 语言的基础，由于 PHP 是一门嵌入式语言，所以先讲述了一些 HTML 语言知识，而后介绍了 PHP 语言的语法知识。语法是一门语言基础，了解一门语言首先要了解其语法特点。第 3 节介绍了 PHP 提供的数据类型，PHP 提供了包括整型、字符型等在内的多种数据类型。最后介绍了运算符和表达式，这是处理数据的基础。

如果读者对本章内容还有疑问，可参考《精通 PHP 5 应用开发》（秦涛，曾文玉．北京：人民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民。北京：中国水利水电出版社，2003）。

4.8 习题

一、填空题

1．PHP 提供了_____和_____函数来实现输出。

2．浮点型主要用于表示带有小数的数值，有_____和_____两种形式。

3．PHP 通过_____函数来将一个值转换成整型。

4．运算符种类很多，主要有_____、_____、_____、_____等。

5．根据表达式中的运算符的不同，可以把表达式分成_____、_____、_____、_____、_____等。

6．变量必须以_____符号开头，第二个字符必须是_____或_____。

7．PHP 的位运算符主要有_____、_____、_____、_____、_____和_____。

二、选择题

1．赋值运算符“%=”的意义是（　　）。

A．将左边的值对右边取余数赋给左边　　B．将左边的值除以右边的值赋给左边

C．将右边的值对左边取余数赋给左边　　D．将右边的值除以左边的值赋给左边

2．逻辑运算符“xor”的意义是（　　）。

A．$a 和$b 不同时为真，则结果为假　　B．$a 和$b 同时为真，则结果为真

C．$a 和$b 不同时为真，则结果为真　　D．以上都不对

3．选择下面程序的运行结果（　　）。

```
01 <?php
02 $a=5;
03 $b=3;
04 echo $a/$b%$b."<br>";
05 ?>
```

A．1　　B．0　　C．1.7　　D．无输出

4．选择下面程序的运行结果（　　）。

```
01<?php
02  $a=1;
03  $a++;
04  $c=&$a;
05  $b=$c++;
06  echo "\$a=$a <br> \$b=$b <br> \$c=$c ";
07  ?>
```

A．$a=3
$b=2
$c=3

B．$a=3
$b=2
$c=2

C．$a=2
$b=2
$c=3

D．$a=2
$b=2
$c=2

三、简答题

1．简述 PHP 的语言风格。
2．简述 PHP 的数据类型有哪些。
3．简述 PHP 的运算符及其优先级。

四、编程题

1．编写一段程序，统计今天网站卖的商品，计算其总价值、平均价格和每个顾客的购买率。
2．编写一个进行变量转化的程序。

第 5 章　常用流程控制

前面几章已经介绍了 PHP 语言的构成元素，本章将运用这些元素来实现一定逻辑结构的流程控制。元素好比汽车，控制语句好比公路，公路控制着车流方向。所有的流程都是由三种最基本的结构构成的，即：顺序结构、分支结构和循环结构。在 PHP 语言中比较常用的流程控制语句主要有 if、for、switch 等语句，这些语句将在本章中详细介绍。

学习本章，可以获得以下知识点：

- if、if…else 语句的使用；
- switch 语句的使用；
- for 语句的使用；
- while 和 do…while 语句的功能和区别；
- break 和 continue 语句的功能和区别。

5.1　if 语句的使用

if 语句根据判断语句的值，有条件地执行一组语句，即当程序遇到一个二岔路口，需要做出选择时，通过 if 语句选择执行的方向，所以其被称做条件控制语句。if 语句实现选择，其基本结构有三种，下面分别介绍这三种结构。

5.1.1　只含 if 的语句

只含 if 的语句，其语句做单纯的判断，然后决定是否执行，可解释为“若发生了某事，则怎样处理”。其基本结构为：

```
if(判断语句)
{
    执行语句体
}
```

判断语句通常都用逻辑运算符号，如果值为非 0，则执行相应语句体，否则就跳过该语句，继续向下执行。其执行流程图如图 5-1

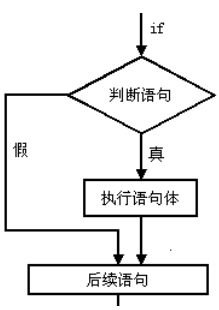

图 5-1　if 语句执行流程图

所示。

【范例 5-1】给变量 a、b 分别赋值 3、5，比较 a、b 两个数的大小。如果 a 大于 b，输出：a 大于 b，并将 b 的值赋给 a。如果 a 不大于 b，输出：a 小于等于 b。其程序如示例代码 5-1 所示。

示例代码 5-1

```
01  <?php                                    //PHP 标记开始
02      $a=3;                                //给变量赋 a 值
03      $b=5;                                //给变量赋 6 值
04      if ($a > $b) {                       //比较 a、b 的大小
05          echo "a 大于 b";                 //输出 a 大于 b
06          $a = $b;                         //变量 b 的值赋给 a
07      }
08      echo ""a 小于等于 b";                //输出 a 小于等于 b
09  ?>                                       //PHP 标记结束
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-1.php，查看运行结果，即可看到其结果如图 5-2 所示。

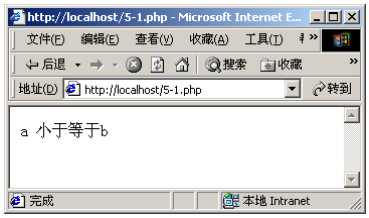

图 5-2　if 条件语句执行结果

【代码解析】代码 02～03 行给变量$a、$b 赋值。第 04 行判断 a、b 的大小，如果为真，执行 05～06 行输出 echo 语句。如果为假，则跳过 05、06 行，执行第 08 行，输出“a 小于等于 b”。

注意：如果只有一句执行语句，则可省略大括号。当执行语句多于一句时，大括号不能省略，否则将产生错误。if 语句可以无限制地嵌套其他 if 语句，但要注意不能有逻辑错误。

5.1.2　if...else 结构语句

if 语句的第一种结构可以选择是否执行某语句体，而第二种 if...else 结构语句必须在两个语句体中选择其中一个来执行。if...else 语句为选择性语句，可以解释成“若发生了某事则怎样处理，否则该如何解决”。其基本结构为：

```
if(判断语句)
{
        执行语句体 1
}
else
{
执行语句体 2
}
```

上述结构表示：如果判断语句的值为非 0（ture）即真，则执行 if 后执行语句体 1，执行完语句体 1 后，从执行语句体 2

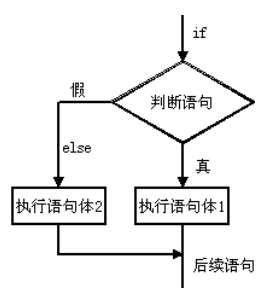

图 5-3　if…else 流程控制图

后开始继续向下执行。如果表达式的值为 0（false）即假，则跳过执行语句体 1 而执行 else 后的执行语句体。其执行流程图如图 5-3 所示。

【范例 5-2】判断变量$date 今天是否是星期日，如果是，输出“今天可以出去玩了”。如果不是，则输出“待在家学习”。其程序如示例代码 5-2 所示。

示例代码 5-2

```
01  <?php                                    //PHP 开始标记
02      $date=sunday;                        //给变量赋值
03      if($date==sunday){                       //判断
04          echo "今天可以出去玩了";           //输出“今天可以出去玩了”
05      }
06      else{                                //如果不是周日
07          echo "待在家学习";                //输出“待在家学习”
08      }
09  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-2.php，查看运行结果，即可看到其结果如图 5-4 所示。

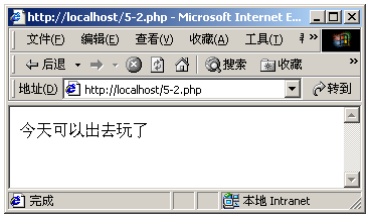

图 5-4　if…else 语句执行结果

【代码解析】上述程序结构非常简单，03～08 行是一个完整的 if…else 结构。第 03 行判断变量$date 的值，如果为真，执行第 04 行，输出“今天可以出去玩了”。如果为假，则执行第 06 行，输出“待在家学习”。

5.1.3　嵌套的 if...else 结构

前面的两种分支机构都只能实现两路分支，当程序结构里还有 if...else 语句时，就能实现多路分支，此结构称为嵌套的 if...else 结构。其基本结构如下：

```
If（判断语句 1）{
执行语句体 1
}
else if(判断语句 2){
  执行语句体 2
}
else if...
else...
```

上述结构表示：如果判断语句 1 的值为真，则执行语句体 1。否则转入后面的 else…if 语句，判断语句 2 若为真，则执行语句体 2。否则，继续转入后面的语句，直到某一执行语句被执行，跳出整个 if...else 循环。这种循环嵌套可以不含 else 语句，即只含有 if、else…if 语句。其循环流程图如图 5-5 所示。

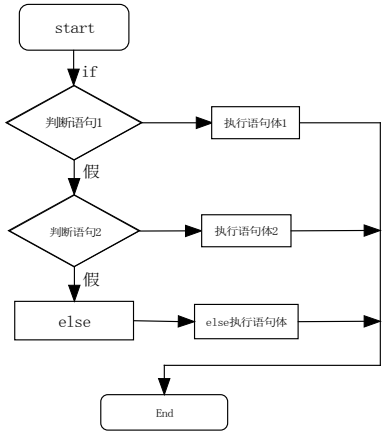

图 5-5　流程控制图

【范例 5-3】判断学生的成绩，如果大于 80 则输出“成绩优秀”。大于 60 输出“及格了”。小于 60 大于 30 输出“没有通过考试”。否则输出“成绩有误”。其程序如示例代码 5-3 所示。

示例代码 5-3

```
01  <?php                                    //PHP 开始标记
02      $score=73;                           //变量赋值
03          if ($score >=80){                //判断成绩是否大于 80
04              echo  "成绩优秀";             //输出“成绩优秀”
05          }
06          else if ($score>=60){            //判断成绩是否大于 60
07              echo "及格了";                //输出 “及格了”
08          }
09      else if($score>=30){                 //判断成绩是否大于等于 30
10          echo "没有通过考试!";            //输出“没有通过考试”
11      }
12      else                                 //如果不符以上情况
13      echo "成绩有误";                     //输出“成绩有误”
14  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-3.php，查看运行结果，即可看到其结果如图 5-6 所示。

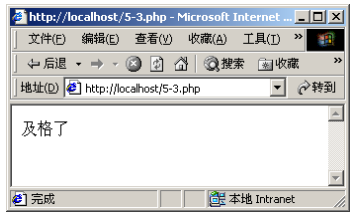

图 5-6　if…else 嵌套执行结果

【代码解析】上述示例代码是一个 if…else 嵌套，第 03、06、09 行分别对变量进行判断，当为真时则执行此种情况下的执行语句体。

提示：嵌套的 if...else 结构简单，层次清晰，比较易于掌握，但注意不能有逻辑上的错误。

5.1.4 if 语句多种嵌套

在 if 语句中可以嵌套一个或多个 if 语句，以实现对多个参数的判断，这就是 if 语句的多种嵌套。其结构形式如下：

```
if()
  if()   语句体 1
  else  语句体 2
else
  if()   语句体 3
  else  语句体 4
```

应当注意 if 和 else 的配对关系，从内层开始，else 总是与它上面最近的 if 配对，和代码缩进无关。缩进的作用只是使代码富有层次感，美观易读，对目标代码的生成毫无影响。

【范例 5-4】利用 if 嵌套语句，判断性别变量$gender 和年龄变量$age。如果变量$gender 的值为 male，且年龄大于 18，则输出“Hi! .sir”，否则输出“you are a boy.”。如果变量$gender 的值不是 male，且年龄大于 18，输出“how are you”，否则输出“you are a girl.”。其程序如示例代码 5-4 所示。

示例代码 5-4

```
01  <?php                                          //PHP 开始标记
02      $gender=male;                              //变量赋值
03      $age=13;                                   //变量赋值
04      if ($gender=="male"){                      //判断性别是否为男士
05          if($age<18){                           //判断年龄是否小于 18
06          echo  "you are a boy.";                //输出结果
07          }
08          else if ($age>=18){                    //如果是男士且年龄大于等于 18
09          echo "Hi! .sir";                       //输出结果
10          }
11      }
12      else{                                      //如果不是男士
13          if($age<18){                           //判断如果不是男士且年龄小于 18
14          echo  "you are a girl.";               //输出结果
15          }
16          else {                                 //如果不是男士且年龄不小于 18
17          echo "how are you";                    //输出结果
18          }
19      }
20  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-4.php，查看运行结果，即可看到其结果如图 5-7 所示。

图 5-7　嵌套 if...else 循环

【代码解析】程序第 04 行利用 if 语句先判断变量$gender，根据结果再执行 if...else 循环，第 03、13 行对其变量$age 判断，输出结果。

警告：对于编程新手，在使用这种嵌套 if...else 循环时，请小心使用。因为太多层的循环容易使设计的逻辑出问题，或者少打了大括号等，都会导致程序出现莫名其妙的问题。

5.2 switch 语句的使用

在 5.1 节中已经讲述了用嵌套的 if...else 循环来实现多分支的选择，但是对于只有一种层面上的多分支，用这种结构实现显得十分冗长。PHP 中提供的 switch 语句也能进行多分支选择，并且使程序变得更简明。

5.2.1 switch 语句

switch 语句是多分支选择性语句，PHP 中 switch 语句的作用是根据不同的条件判断执行不同的语句。其结构如下所示。

```
switch(表达式)
{
case 1
   执行语句体 1
    break;
case 2
   执行语句体 2
break;
……
default;
执行语句体 n
break;
}
```

switch 后面括号内的表达式，可以是整型表达式或字符型表达式，也可以是枚举型数据。表达式的值与 case 语句后面的值逐个进行匹配，如果匹配得上，就执行该 case 语句后面的执行语句体，然后通过 break 语句跳出该循环。如果最终没有 case 语句能与表达式的值相匹配，则执行 default 后面的执行语句体。

警告：每个case的常量表达式的值必须互不相同，否则就会出现错误。

【范例 5-5】0 代表“she is a girl”，1 代表“he is a boy”，2 代表“it is a cat”。将值 1 赋予变量$i。利用 switch 语句，case 后不含 break 语句。判断输出对应情况，其程序如示例代码 5-5 所示。

示例代码 5-5

```
01  <?php                                    //PHP 开始标记
02          $i = 1;                          //变量赋值
03      switch($i){                          //取得变量值
04      case 0:
05          echo "she is a girl<br>";        //输出“she is a girl”
06      case 1:
07          echo "he is a boy<br>";          //输出“he is a boy”
08      case 2:
09          echo "it is a cat<br>";          //输出“it is a cat”
10      }
11  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-5.php，查看运行结果，即可看到其结果如图 5-8 所示。

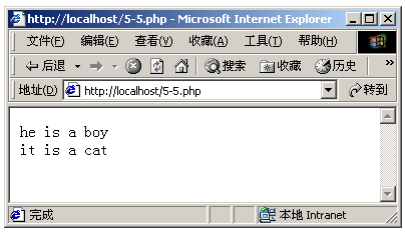

图 5-8　缺少 break 的执行结果

【代码解析】上述代码第 02 行给变量$i 赋值为 1，进入 switch 语句。第 04 行判断 case 的第一种情况，不符合，进入第 06 行判断 case 1，为真，则进入执行 case 1 语句体。由于没有遇到 break，不再判断，接着执行以后的情况，输出了“it is a cat”语句。

注意：初学者容易犯的错误是漏掉分支语句后的 break 语句。如果 case 分支后面没有 break 语句，那么即使匹配了此分支，程序仍将进入下一个 case 分支，直到遇到 break 语句才跳出该循环。

5.2.2 switch 语句的其他形式

switch 语句中的 case 语句是按顺序执行判断，当没有任何一种情况与表达式的值相等时，就执行 default 分支，因此 default 的应该放置到最后，其后面可以不加 break 语句。另外，case 语句也可以为空，当程序执行到此时，将自动转移到下一个 case 语句。

【范例 5-6】将字符串“php 基础教程”赋给变量 name，利用 switch 语句判断环变量 name 的值，如果匹配不上，则输出“对了！呵呵”。其程序如示例代码 5-6 所示。

示例代码 5-6

```
<?php                                          //PHP 开始标记
    $name="php 基础教程 ";                      //变量赋值
    switch($name){                             //取得变量的值
        case"php":
            echo"不准确哦";                     //输出“不准确哦”
            break;                             //跳出 switch 循环
        case"教程":
            echo"太笼统了";                     //输出“太笼统了”
            break;                             //跳出 switch 循环
        case"基础教程":
            echo"快对了";                       //输出“快对了”
            break;                             //跳出 switch 循环
        default:
            echo"对了!呵呵";                    //输出“对了!呵呵”
            break;                             //跳出 switch 循环
    }
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-6.php，查看运行结果，即可看到其结果如图 5-9 所示。

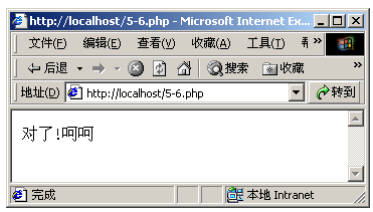

图 5-9 switch 语句执行结果

【代码解析】上述示例代码比较简单，如果在 case "php"分支后添加分支 case 0，后面为空，则将输出“太笼统了”，请读者自己试验。

5.3 while 语句的使用

当程序需要不停地执行某语句，直到达到某一条件为止时，这种情况需要采用循环程序机构。PHP 中常用 while 循环来实现。while 循环是 PHP 中最简单的循环类型，常见的有 while…、do…while…两种结构形式。下面分别介绍这两种结构循环语句。

5.3.1 while 循环

while 用来实现“当型”循环，其基本结构如下：

```
while(判断语句)
{
  执行语句体
}
```

这是一种普遍的格式，判断语句一般用逻辑运算符号作为判断条件。当判断语句为真时，执行语体被执行后再检查表达式的值，如果为真，执行语句体再次被执行，当判断语句的值为假时，退出该循环。

【范例 5-7】利用 while 循环，输出小于等于 6 的各个数的和，其程序如示例代码 5-7 所示。

示例代码 5-7

```
01  <?php                                  //PHP 开始标记
02      $int=1;                            //变量赋值
03      $sum=0;                            //变量赋值
04      while($int<=6){                    //取得变量值
05
06          $sum=$sum+$int;                //输出变量值
07          echo $sum;
08          echo "<br>";                   //空格
09          $int ++;                       //变量自加
10      }
11  ?>                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-7.php，查看运行结果，即可看到其结果如图 5-10 所示。

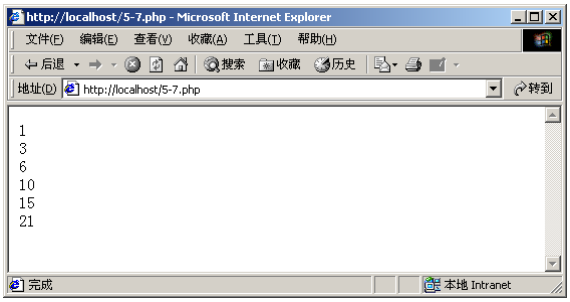

图 5-10　while 循环执行结果

【代码解析】这是一个循环输出 11 以内的整数的例子。第 03 行当 while 语句的判断语句为真时，则一直执行 while 下的语句。程序每执行一次循环体，$int 的值就加 1，直到$int 的值等于 11，才终止循环。“\n”的作用是空格，不能省略。也可改进此程序，使其变得更简洁。

```
while(++$int<10)
echo "$int\n";
```

此程序输出结果同图 5-10 所示。

注意：任何循环都必须出现一次循环为假的情况，否则循环将无限进行下去，从而成为死循环。

5.3.2 do…while 循环

do…while 循环非常类似于 while 循环，是 while 循环的变体。只是其在每次循环结束时检查表达式是否为真，而不是在循环开始时。do…while 又称“直到型”循环，其基本结构如下：

```
do{
语句体
}while(表达式)
```

其特点是程序先执行语句体，再进行判断表达式。它是这样执行的：先执行一次指定的内嵌的语句体，然后判断表达式。如果为真，返回执行语句体，若为假，结束 do…while 循环。

【范例 5-8】利用 do…while 循环，输出 1～10 之间各数的阶乘，其程序如示例代码 5-8 所示。

示例代码 5-8

```
01  <?php                                  //PHP 开始标记
02      $i=1;                              //变量赋值
03      $sum=1;                            //变量赋值
04      do{                                //do 语句
05          echo $sum=$sum*$i;             //输出
06          echo "\n";                    //空格
07          $i++;                          //变量自加
08      }while($i<=10)                     //判断
09  ?>                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-8.php，查看其运行结果，即可看到如图 5-11 所示的结果。

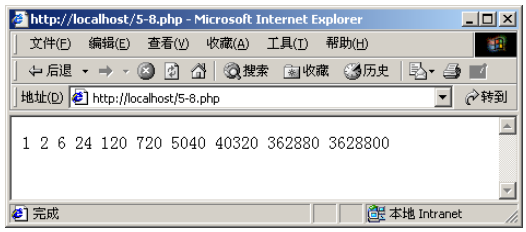

图 5-11　do…while 循环

【代码解析】程序第 02、03 行定义了两个变量，第 04 行开始进入 do…while 循环，直到第 08 行的判断语句为假，则跳出 do…while 循环。

5.3.3 while 和 do…while 循环的区别

while 循环和 do…while 循环的主要区别是第一次循环肯定要执行。在一般情况下，用 while 语句和 do…while 语句处理同一问题时，若二者的循环体部分相同，它们的运行结果一般也相同，但在 while 后面的表达式一开始为假时，两种循环的结果是不同的。

【范例 5-9】将变量 year 的值设为 2008，判断变量 year 同 2008 的大小关系，利用 while 循环和 do…while 循环输出迎奥运活动，比较两者的区别。其程序如示例代码 5-9 所示。

示例代码 5-9

```
01  <?php                                  //PHP 开始标记
02      $year=2008;                        //变量赋值
```

```
03          //没有进入 while 循环
04      while($year>2008){                    //判断
05          echo "我们进不去啊";               //程序执行不到这一行
06      }
07          //至少进入一次 do…while 循环
08      do{
09          echo "喜迎 08 奥运会";            //程序将输出“喜迎 08 奥运会”
10      }while($year>2008)                    //判断
11  ?>                                        //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-9.php，查看运行结果，即可看到其结果如图 5-12 所示。

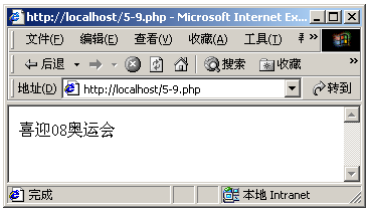

图 5-12　对比 while、do…while 循环

【代码解析】代码第 02 行给变量$year 赋值，第 03 行判断 while 语句中判断语句，因为为假，则不执行语句体。第 08 行执行 do 语句后再判断 while 语句中的判断语句，如果为假，则跳出循环。

提示：通过上述示例能看出 do…while 循环和 while 循环两者的区别，在使用时要小心，有时不能替代。

5.4　for 语句的使用

PHP 中 for 循环是最复杂、功能也最强大的循环，任何 while 循环都可以用 for 循环代替。其写法多种多样，下面分别介绍几种常见的写法。

5.4.1　一般形式

语言中的 for 循环语句使用最为灵活，不仅可以用于循环次数已经确定的情况，而且可以用于循环次数不确定，而只给出循环结束条件的情况。其结构形式如下：

```
for(表达式 1，表达式 2，表达式 3)
{
  执行语句体
}
```

其执行过程如下所述。

① 先求表达式 1。

② 求表达式 2，若其值为真，则执行 for 语句中的执行语句体。若为假（值为 0），执行步骤（5），结束循环。

③ 求解表达式 3。

④ 转回上面第②步骤继续执行。

⑤ 结束循环，执行 for 以后的语句。

其执行流程图如图 5-13 所示。

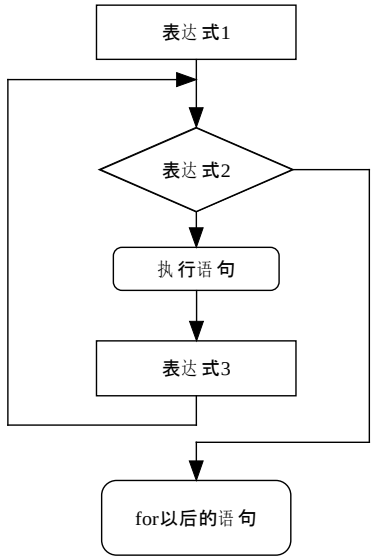

图 5-13　for 语句的执行流程图

【范例 5-10】利用一个 for 循环语句，算出距周一有几天。其程序如示例代码 5-10 所示。

示例代码 5-10

```
01  <?                                                    //PHP 开始标记
02      print("<B>距离星期一还有几天？</B>\n");
03      print("<OL>\n");
04      for($currentDate = date("U");                     //定义$currentDate 时间格式
05          date("l", $currentDate) != "Monday";          //判断是不是当前系统时间是Monday
06          $currentDate += (60 * 60 * 24))               //当前时间加上 1 天
07      {
08          print("<LI>" . date("l", $currentDate) . "\n");
                                                          //打印时间
09      }
10      print("</OL>\n");
11  ?>                                                    //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-10.php，查看运行结果，即可看到其结果如图 5-14 所示。

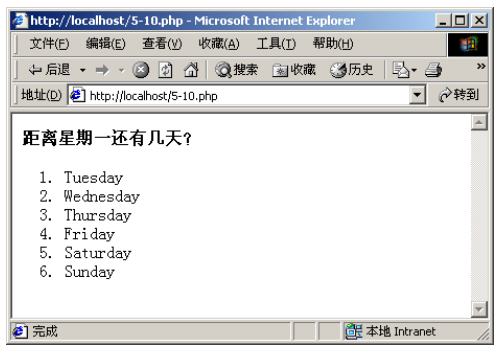

图 5-14　for 语句

【代码解析】程序第 02 行打印粗体字体，第 04 行利用$dae（U）定义时间格式，第 05 行判断系统时间，第 06 行计算距周一的时间。

for 语句完全可以代替 while 语句，其中示例代码 5-7 的执行结果还可以通过 for 循环实现。其代码如下：

```
01  <?php                                  //PHP 开始标记
02      $i = 1;                            //变量赋值
03      for (;;) {                         //省略表达式的 for 语句
04      if ($i > 10) {                     //判断
05          break;                         //跳出循环
06      }
07      echo $i. "\n";                     //输出变量值
08      $i++;                              //变量自加
09      }
10  ?>                                     //PHP 结束标记
```

第 02 行给变量赋值，03～09 是 for 语句，中间嵌套 if 语句，起判断是否跳出 for 循环的作用，其执行结果如图 5-9 所示。

提示：for 语句中三个表达式均可省略。

5.4.2 多重循环

使用多重循环来控制多个变量也是在 for 语句中经常被忽略的一个特性，因为这种执行方法比较容易出错。在一般的任务中用到的是双重循环，三重以上的循环一般意义不大，这里通过实例讲解 for 语句的多重循环。

【范例 5-11】利用 for 语句的多重循环，输出一个 9*9 乘法表，以表格实行输出，其程序如示例代码 5-11 所示。

示例代码 5-11

```
01  <?php                                                    //PHP 开始标记
02      print("<table border=1>");                           //表格开始
03      for($row=1; $row <= 9; $row ++)                      //判断执行语句
04      {
05          print("<tr>\n");                                 //开始行
06          for($column=1; $column <= $row; $column ++)      //for 判断执行语句
07      {
08          print("<td>");                                   //开始列
09          print($row * $column);                           //表格元素乘积
10          print("</td>");                                  //结束列
11      }
12          print("</tr>");                                  //行结束
13      }
14      print("</table>");                                   //表格结束
15  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-11.php，查看其运行结果，即可看到如图 5-15 所示的结果。

【代码解析】程序第 02 行定义了表格线宽。第 03 行执行 for 语句循环，第 06 行嵌套一个 for 语句，开始另一个循环，两个语句循环判断执行。

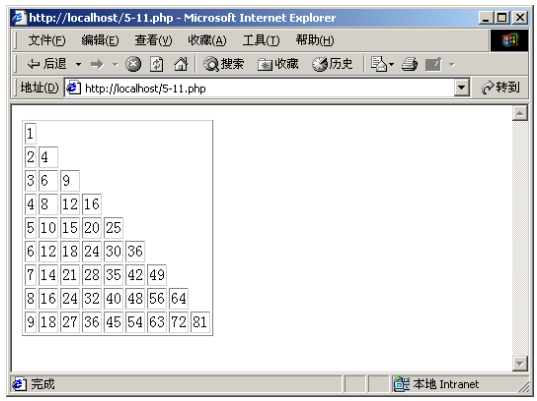

图 5-15　多重循环

5.5　break/continue 语句的使用

前面已经提过 break 的用法，其实 continue 和 break 都是在循环体中控制程序跳转的语句。前面例题中循环的结束是通过判断循环控制条件为假而正常退出的。然而在某些场合，只要满足一定的条件就应当提前结束循环的执行，或只结束本次循环，转入下次循环。

5.5.1　break 语句的用法

break 具有结束、中断整个循环的作用，可以被用到 for、do…while 等循环中，起无条件结束循环的作用。实际上 break 语句还可以用来从循环体内跳出循环，即提前结束循环，接着执行其他的语句。

【范例 5-12】计算半径为小于 5 的整数，面积在 50 以内的圆的面积。其程序如示例代码 5-12 所示。

示例代码 5-12

```
01  <?php                                          // PHP 开始标记
02          $pi=3.14;                              //变量赋值
03          for($r=1;$r<=5;$r++){                  //判断执行
04          $area=$pi*$r*$r;                       //计算面积
05          if($area>50)break;                     //判断
06              echo $area."<br>";                 //输出
07      }
08  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-12.php，查看运行结果，即可看到其结果如图 5-16 所示。

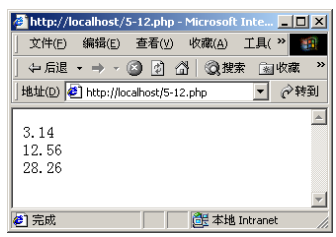

图 5-16　break 的用法

【代码解析】上述示例代码是 for 语句中嵌套一个 if 语句，if 语句判断变量$area 是否大于 50，当大于 50 时，终止 for 循环。

5.5.2 continue 语句的用法

continue 语句起结束本次循环，执行下次循环的作用，其一般形式为：

```
continue;
```

continue 语句的作用是结束本次循环，即跳过循环体中后面未执行的语句，接着执行下一次循环。

【范例 5-13】利用 continue 语句，输出 1～10 之间不能被 3 整除的数。其程序如示例代码 5-13 所示。

示例代码 5-13

```
01  <?php                                    // PHP 开始标记
02      for($n=1;$n<=10;$n++){               //判断执行
03          if($n%3==0)                      //判断
04          continue;                        //跳过以后的语句
05      echo $n."<br>";                      //输出变量值
06      }
07  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-13.php，查看运行结果，即可看到其结果如图 5-17 所示。

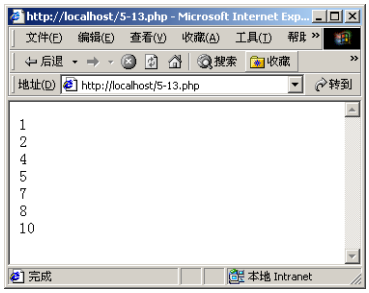

图 5-17　continue 语句执行结果

【代码解析】程序 03～04 行是一个 if 语句，当条件为真，执行 continue 语句，使之返回 for 循环。若为假，执行 if 后的语句。

提示：读者可以将 04 行的 continue 换为 break 语句，查看其输出结果变化。

5.5.3 continue 和 break 语句的区别

continue 语句的作用是结束本次循环，接着去判断是否执行下次循环，而 break 语句则终止整个循环，不再进行判断。两个语句在 while 循环中使用方法的如下：

break 语句：

```
while(判断语句1)
{
    执行语句体
  if(判断语句2)
    break;
```

```
   执行语句体
 }
```

continue 语句：

```
while(判断语句1)
{
    执行语句体
  if(判断语句2)
    continue;
  执行语句体
 }
```

程序的执行流程图如图 5-18 和图 5-19 所示。

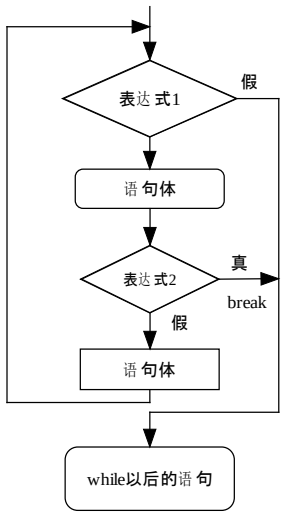

图 5-18　break 执行流程图

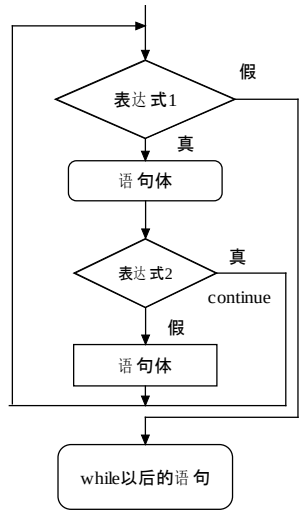

图 5-19　continue 执行流程图

5.6　综合练习

1．利用 for 循环输出 99 乘法表。

提示：9*9 表是两个变量相乘获得的，要利用循环嵌套，在两个数相等时结束一次循环。其代码如示例代码 5-14 所示。

示例代码 5-14

```
<?php                                          // PHP 开始标记
    for($i=1;$i<=9;$i++){                      /for 循环判断语句
        echo  "<table>";                       //定义表格开始
        echo "<tr>";                           //定义列
    for($k=1;$k<=$i;$k++){                     // for 循环判断语句
        echo "<td>";                           //定义行开始
        echo $i*$k;                            //输出数值
        echo "</td>";                          //定义行结束
    }
        echo "</tr>";                          //定义列结束
        echo "</table>";                       //定义表格结束
        }
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-14.php，查看运行结果，即可看到其结果如图 5-20 所示。

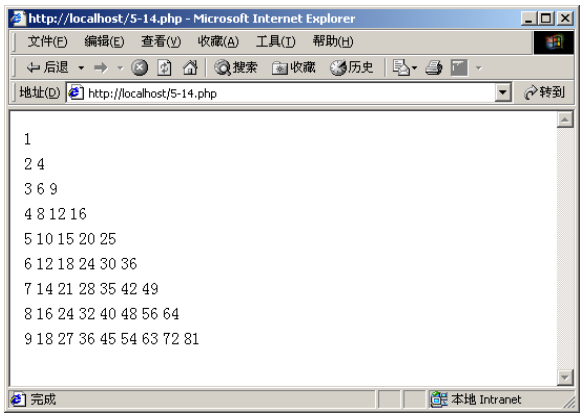

图 5-20　99 乘法表

2．在一周的时间内，输出当天星期几，周末输出今天休息。

提示：时间变量可以通过 for 循环实现，然后通过 switch 语句判断执行不同的输出。其代码如示例代码 5-15 所示。

示例代码 5-15

```
<?php                                                    //PHP 开始标记
    for($d=1;$d<=7;$d++){                                //for 循环判断语句
        echo "\$d= $d <br>";                             //输出变量值
        switch ($d){                                     //判断变量$d 的值
        case 1:                                          ///判断变量
        echo "今天周一";                                 //输出“今天周一”
    break;                                               //跳出循环
    case 2:                                              //判断变量
        echo "今天周二";                                 //输出“今天周二”
        break;                                           //跳出循环
    case 3:                                              //判断变量
        echo "今天周三";                                 //输出“今天周三”
        break;                                           //跳出循环
    case 4:                                              //判断变量
        echo "今天周四";                                 //输出“今天周四”
        break;                                           //跳出循环
    case 5:                                              //判断变量
        echo "今天周五";                                 //输出“今天周五”
        break;                                           //跳出循环
 default :                                               //无条件情况
    echo "今天休息" ;                                    //输出“今天休息”
    }
    echo "<hr width=300 align=left><br>";                //定义 hr
    }
?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/5-15.php，查看运行结果，即可看到其结果如图 5-21 所示。

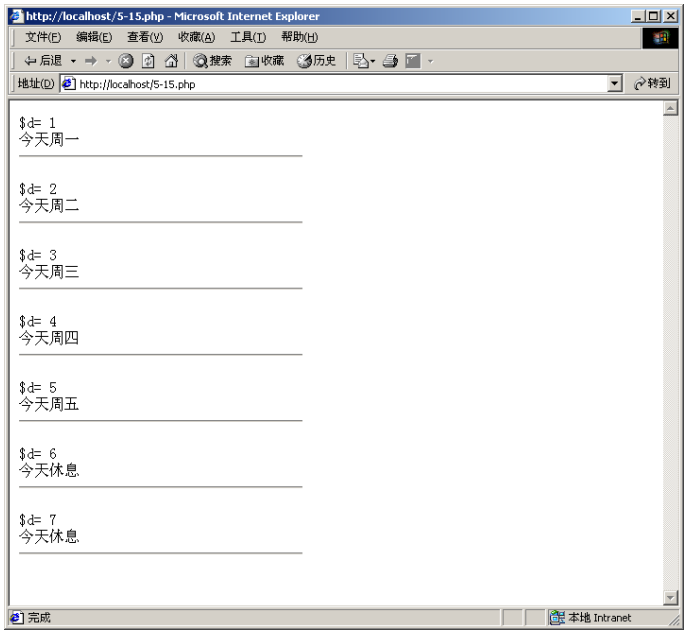

图 5-21　一周活动

5.7　小结

本章主要介绍了几种常见的语句结构，如 if 语句、for 语句等。通过本章的学习，相信读者能够编写出具有一定功能的语句，为今后的编程打下很好的基础。本章所提供的例子都较简单，今后会逐渐加大程序的广度和深度。

如果读者对本章内容还有疑问，可参考《精通 PHP5 应用开发》（秦涛，曾文玉．北京：人民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民．北京：中国水利水电出版社，2003）。

5.8　习题

一、填空题

1．PHP 语言中比较常用的流程控制语句主要有_____、_____、_____等语句。

2．if 判断语句如果其值为_____，则相应语句被执行。

3．switch 语句后面括弧内的表达式可以是_____、_____、_____。

4．若 case 语句中的常量表达式的值都没有与表达式的值匹配，则执行_____后面的语句。

5．用 while 语句和 do…while 语句处理同一问题时，若二者的循环体部分相同，它们的运行结果一般也相同，但在_____时，两种循环的结果是不同的。

6．_____语句只结束本次循环，而不是终止整个循环的执行。

7．case 常量表达式只是起语句标号作用，并不是在该处进行_____。

二、选择题

1．从循环体内跳出循环，即提前结束循环的语句是（　　）。

A．break 语句　　B．continue 语句

C．if 语句　　D．switch 语句

2．选择下面程序的运行结果（　　）。

```
01  <?php
02   $ip = 76;
03   if ($ip == 76)
04   $i = 1;
05   switch($i){
06  case 0:
07            echo "根据 IP 判断，您来自中国。<br>";
08  case 1:
09            echo "根据 IP 判断，您来自美国。<br>";
10  case 2:
11            echo "根据 IP 判断，您来自日本。<br>";
12       }
13  ?>
```

A．根据 IP 判断，您来自美国。
　　根据 IP 判断，您来自日本。

B．根据 IP 判断，您来自美国。

C．根据 IP 判断，您来自日本。

D．没有任何输出

3．选择下面程序的运行结果（　　）。

```
01  <?php
02       $i=1;
03       do{
04            echo "您在该网站购买了".$i."件商品<br>";
05            $i++;
06       }while($i%10==0);
07       echo 浏览了.$i."件商品";
08  ?>
```

A．您在该网站购买了 1 件商品
　　浏览了 1 件商品

B．您在该网站购买了 2 件商品
　　浏览了 2 件商品

C．您在该网站购买了 1 件商品
　　浏览了 2 件商品

D．您在该网站购买了 2 件商品
　　浏览了 1 件商品

4．选择下面程序的运行结果（　　）。

```
1 <?php
2        for($n=1;$n<=100;$n++)
3        {    if($n%10!=0)
4             continue;
5        echo "您是第".$n."位给我留言的朋友<br>";
6        }
7 ?>
```

A．发生编译错误

B．您是第 10 位给我留言的朋友

C．您是第 1 位给我留言的朋友
　　您是第 2 位给我留言的朋友

您是第 3 位给我留言的朋友
您是第 4 位给我留言的朋友
您是第 5 位给我留言的朋友
您是第 6 位给我留言的朋友
您是第 7 位给我留言的朋友
您是第 8 位给我留言的朋友
您是第 9 位给我留言的朋友

D．没有任何输出

三、简答题

1．简述 while 和 do…while 循环的区别。

2．简述 continue 和 break 语句的区别。

四、编程题

1．利用 for 循环语句，编写今天论坛发了多少帖子。

2．用 do…while 循环输出今天注册的最新 10 名网站会员的用户名。

第 6 章　常 用 函 数

函数是按照某种需要编写出来的一段程序，这段程序就相当于一个模块，需要者可以直接拿来使用，不需要重新编写。其就像一台电视机，不需要理解里面的结构，需要时直接使用即可。这种将具有一定功能的程序函数化的方法，能使程序变得简洁。下面开始学习函数化的方法和一些常用函数。

学习本章，可以获得以下知识点：

- 能够创建一定功能的函数；
- 了解参数的传递和参数的作用范围；
- 掌握函数的调用方法；
- 掌握一些常用函数的用法。

6.1　自定义函数

PHP 提供了功能强大的函数，但这远远满足不了需要，程序员可以根据需要自己创建函数。本节就开始学习创建函数的方法。

6.1.1　自定义函数格式

除了使用 PHP 函数库中的函数之外，还可以定义自己的函数。定义函数的方式是关键字 function 后面加上函数名，其结构如下：

```
function function_name(parameters)
{
//function
```

```
}
```

其代码含义如下所示。

- function 为声明函数的关键字。
- funtion_name 为函数的名称。函数名必须唯一，不分大小写。
- 在 function 和函数名之间要有空格；在函数名和左边的小括号“(”之间以及右边的小括号“(”和左边的大括号“{”之间可以有 0 个、1 个或很多空格、制表符，也可以创建新行。
- perameters 为函数的参数，在调用函数时，参数值可以相互传递。也可以不设参数，不设参数的函数称为无参函数。
- 如果要定义包括多个参数的函数，则可以在 perameters 后面添加其他参数，用逗号将各个参数隔开，括号及其包括内的参数名称被称为参数列表。

下面是一个简单的函数实例，输出“北京奥运会”。

```
01  <?                                          //PHP 开始标记
02      function foo(){                         //声明函数 foo
03          echo"北京奥运会";                    //输出“北京奥运会”
04          }
05  ?>                                          //PHP 结束标记
```

代码第 02 行用关键字 function 声明了一个名为 foo()的函数，大括号内的为执行语句，具体如何调用这个函数将在 6.2 节介绍。

提示：function 函数体可以包括一个单独的语句，也可以包含任意数量的执行语句。

6.1.2 调用用户定义的函数

定义完函数之后，就可以在其他地方调用这个函数了。调用函数的方式是将函数名后面加上括号，其一般形式为：

```
function_name(parameters)
```

如果调用无参函数，则 parameters 可以为空，但不能省略括号。如果参数表包含多个参数，则各参数用逗号隔开。实参与形参按顺序对应传递数据，实参和形参的个数应相等，类型应一致。

【范例 6-1】创建一个不含参数输出“北京奥运会”的函数，并且使用此函数。其程序如示例代码 6-1 所示。

示例代码 6-1

```
01  <?                                          //PHP 开始标记
02      function foo(){                         //声明函数
03          echo "北京奥运会";                   //输出“北京奥运会”
04          }
05      foo()                                   //调用函数 foo
06   ?>                                         //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-1.php，查看其运行结果，即可看到如图 6-1 所示的结果。

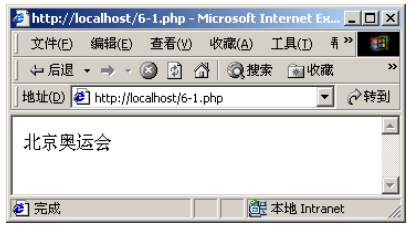

图 6-1　运行结果

【代码解析】程序第 02～04 行定义了一个 foo()函数，第 05 行调用函数 foo()输出“北京奥运会”。

6.1.3　按值传递参数

上面是无参函数的调用，调用有参数函数就有参数值的传递问题。参数传递的方式有按值传递和引用传递两种，PHP 默认为按值传递。外部信息可通过参数传递到函数之中。下面通过实例讲解按值传递的方法。

【范例 6-2】商品的单价为 10 元，有 3 件商品，创建一个函数，用其确定商品总的价格，其程序如示例代码 6-2 所示。

示例代码 6-2

```
01  <?php                                          //PHP 开始标记
02          function selles($price,$num){          //声明函数
03              $total=$price*$num;                //计算结果
04              echo "total cost: $total";       //输出"total coast::30"
05          }
06      selles(10,3);                              //调用函数
07  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-2.php，查看其运行结果，即可看到如图 6-2 所示的结果。

【代码解析】程序第 02～05 行定义了一个含有两个参数的 selles()函数，两个参数分别为$price 和$num。第 06 行调用该函数，并将参数的值传递给相应的变量。执行该函数，输出“total coast:30”。

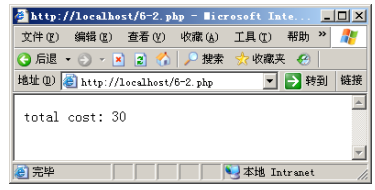

图 6-2　运行结果

参数可能是浮点型，但由于 PHP 的松散性特点，完全可以传递任何类型的变量。另外，并非只能向函数传递静态值，也可以按动态方式传递变量。

【范例 6-3】将示例代码 6-2 修改为按动态方式传递变量，如示例代码 6-3 所示。

示例代码 6-3

```
01  <?php                                          //PHP 开始标记
02      function salas($price,$num){               //声明函数 salas
03          $total=$price*$num;                    //计算结果
04          echo "total cost: $total";           //输出结果
05      }
06      $price=8.8;                                //给$price 变量赋值
07      $num=3;                                    //给$num 变量赋值
08      salas($price,$num);                        //调用函数
09  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-3.php，查看其运行结果，即可看到如图 6-3 所示的结果。

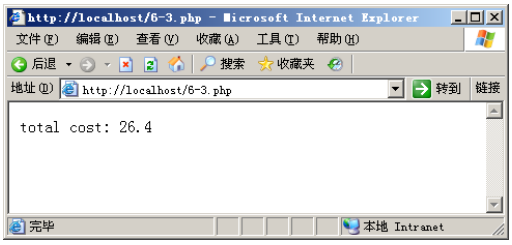

图 6-3　运行结果

【代码解析】上述代码传递浮点型参数。第 06～07 行给变量赋值，第 08 行调用函数，传递参数。执行该函数，输出“total cost:26.4”。

提示：按值传递意味着函数范围内对这些值的任何改变，在函数外都会被忽略。如果希望在函数外也能反映出这些改变，则可以按引用传递参数。引用参数是在参数前面加上“&”符号。

6.1.4 引用传递参数

变量在内存中储存并分配给其地址，应用时通过这个地址能很容易地找到。这就像旅社的一个一个房间，住店时给每人一个房间号，这就是他们的地址，别人可通过这个地址找到他们。如图 6-4 解释了按值传递和引用传递之间的区别。

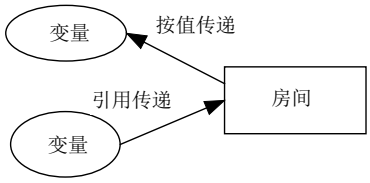

图 6-4 传递参数

按值传递只是传递参数的信息，而参数的地址没办法知道。这就像老板把房间内的人的情况告诉其他人，但不告诉房间的地址。引用传递是传递参数的地址，这就像老板将地址告诉其他人，其他人直接访问该房间。下面通过实例来说明两者的区别。

【范例 6-4】分别用按值传递参数和引用传递两种方式执行$a=$a+1，输出 a 的值比较两者的区别。其程序如示例代码 6-4 所示。

示例代码 6-4

```
01  <?php                                  //PHP 开始标记
02      $a=1;                              //变量赋值
03      function add ($a){                 //声明函数 add
04          $a=$a+1;                       //变量加 1
05          echo $a."<br>";                //输出变量 a 的值
06      }
07      add($a);                           //调用函数
08      echo $a;                           //输出变量
09  ?>                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-4.php，查看其运行结果，即可看到如图 6-5 所示的结果。

【代码解析】第 02 行给变量赋值，第 03～06 行定义函数，执行$a=$a+1 使变量的值改变后输出。第 07 行调用执行函数，第 08 行再次输出变量$a 的值。

由图 6-5 可以看出，函数执行后变量$a 的值并未发生变化。如果将代码 6-4 的第 03 行改为：

```
function add (&$a)
```

则运行结果如图 6-6 所示。

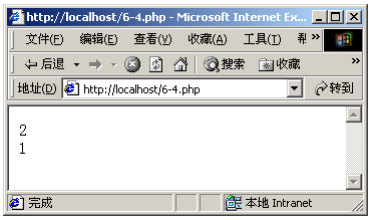

图 6-5 参数传递

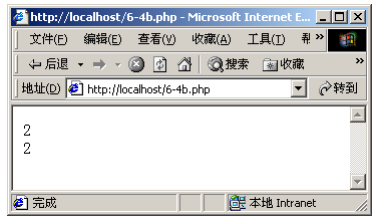

图 6-6 运行结果

警告：如果函数内的变量发生了改变，则存储在内存中的变量值就发生了变化。这一点和静态变量与动态变量的区别有点类似。

6.1.5 定义默认参数

默认参数就是在定义函数时，该参数已经被赋值。如果调用一个包括默认参数的函数，但是没有提供该参数的值，那么该参数将使用指定的默认值。

【范例 6-5】将价格变量 purchase 的值设为 123.45，变量 rate 的默认参数值设为 0.0725。然后将参数 rate 的值改为 0.08 进行参数调用，演示其调用规律。其程序如示例代码 6-5 所示。

示例代码 6-5

```
<?php                                          //PHP 开始标记
    function salestax ($amount, $rate=0.0725)  //定义函数 salestax
    {
        echo "<br>amount=$amount";             //输出变量 amount 的值
        echo "<br>rate=$rate";                 //输出变量 rate 的值
        return $amount*$rate;                  //返回变量值
    }
    $purchase=123.45;                          //定义变量 purchase
    echo "<br>purchase=$purchase";             //输出变量 purchase 的值
    $tax=salestax($purchase,0.08);             //调用函数 salestax
    echo "<br>tax=$tax";                       //输出变量 tax 的值
    $purchase=123.45;                          //定义变量 purchase
    echo "<br>purchase=$purchase";             //输出变量 purchase 的值
    $tax=salestax($purchase);                  //调用函数
    echo "<br>tax=$tax";                       //输出变量 tax 的值
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-5.php，查看其运行结果，即可看到如图 6-7 所示的结果。

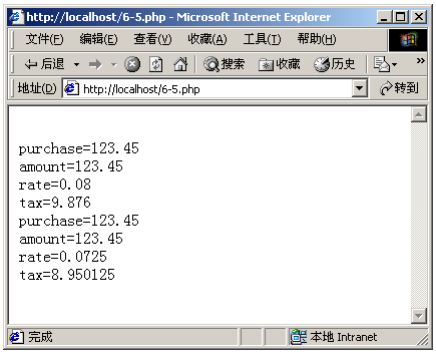

图 6-7 默认参数

【代码解析】salestax 函数定义了两个参数：一个是名为$amount 的必需参数，另一个是名为$rate 的默认参数。如果调用这个函数并只提供一个参数的值，那么这个值将作为$amount 的值，并使用值 0.0725 作为$rate 的值。这样，在第一次调用该函数时，$rate 的值为 0.08，这是作为该函数的第二个参数指定的。不过在第二次调用该函数时，因为只指定了一个参数值，所以$rate 的值仍为 0.0725。

当调用的参数不被赋值时，会产生错误。下面通过实例查看参数不被赋值调用函数时的情况。

【范例 6-6】定义一个含有两个参数的 student1 函数，参数变量 num 的值默认值为 01，再定义函数 student2，参数变量 name 的默认值为 Li，其程序如示例代码 6-6 所示。

示例代码 6-6

```
01  <?php                                          //PHP 开始标记
02      function student1($name, $num=01){         //声明函数 student1
03          echo "amount=$name<br>";                //输出变量 amount 的值
04          echo "num=$num<br>";                    //输出变量 num 的值
05      }
06      function student2($name=Li,$num){          //声明函数 student2
07          echo "amount=$name<br>";                //输出变量 amount 的值
08          echo "num=$num<br>";                    //输出变量 num 的值
09      }
10      student1(L);                               //调用函数 student1
11      student2(02);                              //调用函数 student2
12      student2(liu,03)                           //调用函数 student2
13  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-6.php，查看其运行结果，即可看到如图 6-8 所示的结果。

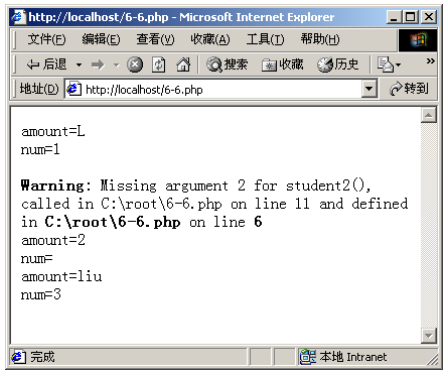

图 6-8　执行结果

【代码解析】程序第 02～09 行定义了两个含有默认参数的函数，其中第二个函数将默认参数$name 放在了非默认参数$num 的前面。第 10～12 行是对两个函数的调用，第 11 行的调用产生了错误，因为$num 没有被赋值，第 12 行调用默认参数的值失去了意义。

注意：当使用默认参数时，任何默认参数必须放在非默认参数的右侧。否则，会造成错误。

6.1.6　函数返回值

编写程序时，有时不想将所得结果直接输出到浏览器上，而将结果传递给其他变量，这种情况下，可以利用 return 语句。return 语句可以将数据传递给调用者，而不是立即传递给用户。而且 return 语句可以返回任何类型的数据，其中包括数组和对象。

【范例 6-7】计算一个长为 4、宽为 3 的长方形面积，通过变量输出其面积。其程序如示例代码 6-7 所示。

示例代码 6-7

```
01  <?php                                          //PHP 开始标记
02      function calculate_area($height,$width){   //声明函数 calculate_area
03          $area=$height*$width;                  //计算结果
```

```
04            return $area;                          //返回计算结果
05        }
06        $total=calculate_area(4,3);                //返回值赋给变量
07        echo $total                                //输出变量 total 的值
08    ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-7.php，查看其运行结果，即可看到如图 6-9 所示的结果。

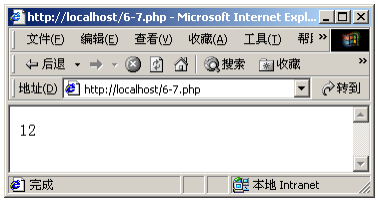

图 6-9　函数返回值

【代码解析】程序第 04 行将计算结果返回给函数，第 06 行调用函数，并将返回值赋给变量$total。

上述代码还可以省去变量$total，而直接返回计算结果。其程序可改写为：

```
function calculate_area($height,$width){
   return  $height*$width;
  }
```

上述代码结果与示例代码 6-7 相同。

6.1.7　返回多个值

return 语句不但可以一次返回一个值，而且可以一次返回多个值，下面通过实例来讲解使用 return 语句返回多个值。

【范例 6-8】定义一个名为 results 的函数，定义一个数组$fruit，数组元素为 apple、banana、orange，并用 return 返回其数组元素的值。其程序如示例代码 6-8 所示。

示例代码 6-8

```
01  <?php
02      function results(){                          //声明函数
03      $fruit=array("apple",                        //声明数组
04      "banana",                                    //数组元素
05      "orange");                                   //数组元素
06      return $fruit;                               //返回
07      }
08          //调用函数
09      $a = results();                              //调用函数
10      print_r($a);                                 //输出数组
11  ?>
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-8.php，查看其运行结果，即可看到如图 6-10 所示的结果。

图 6-10　运行结果

【代码解析】程序第 02～07 行定义了一个函数，第 03 行定义了一个名为 fruit 的数组，第 09 行调用函数 results，第 12 行输出返回结果。

注意：return 语句可以返回任何类型的变量，这是使自定义函数返回多个值的关键。

6.2 函数的高级调用

PHP 中，函数之间都是平行独立的。函数内部不但可以是一般的执行语句，也可以是函数，而且可以实现调用自身函数，这些都是函数的高级调用。下面分别介绍这些调用方式。

6.2.1 嵌套调用

嵌套调用是指在一个函数中还调用了其他函数，函数的定义是独立的，各函数均处于平行的关系。这种关系就像一个大的容器里可以放置很多小容器，这些容器彼此独立，各自容纳自己的东西，和其他容器没有关系。

【范例 6-9】在函数 areas 中声明另一个函数 convert_pound，计算每平方米的布价为 2.1 元，买宽为 2、长为 3.6 的布料所需的钱，其程序如示例代码 6-9 所示。

示例代码 6-9

```
01  <?php                                                    //PHP 开始标记
02      function areas($weight,$length){                     //声明函数 areas
03          function convert_pound($heigh, $price=2.1){      //声明函数
04              return $heigh*$price;                        //返回计算结果
05      }
06          $area=$weight*$length;                           //计算面积
07          echo " Total cost in ". convert_pound($area);    //输出计算结果
08          }
09      areas(2,3.6);                                        //调用函数 areas
10  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-9.php，查看其运行结果，即可看到如图 6-11 所示的结果。

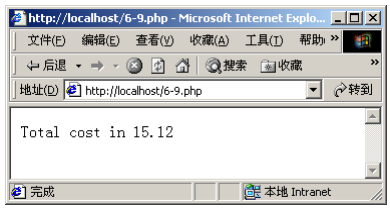

图 6-11 运行结果

【代码解析】程序第 03～05 行定义了嵌套在 areas 函数下的函数，第 06 行对定义在 areas 函数内的函数进行了调用。

注意：由于结构化程序的特点，一个大的问题往往会被逐步分解为不同层次的许多函数。因此，对函数的嵌套调用将不可避免，并大量存在。

6.2.2 递归调用

数学中有一些需要采用递推形式的算法，例如要求数 *n* 的阶乘。在求解 *n* 的阶乘中使用了(*n*-1)的阶乘，即要算出 *n*!，必须先要知道(*n*-1)!。而要知道(*n*-1)!，又必须知道(*n*-2)!，依此类推，

直至 1!=1。只有求得 1!=1，再从此为基础，返回来计算 2!，3!……(*n*-1)!，*n*!，才能得到最终欲求得的 *n*!值。

解决此类递推求值问题所使用的函数程序必须要能调用自身，此种能直接或间接调用自身的函数在 PHP 语言中是允许的，称之为递归函数。对其进行调用，称之为递归调用。

【范例 6-10】使用递归计算 5 的阶层，其程序如示例代码 6-10 所示。

示例代码 6-10

```
<?php                                          //PHP 开始标记
    function recurser($n)                      //声明函数 recurser
    {
        if($n==0)                              //判断变量是否为 0
        return 1;                              //返回值 1
        else
        return $n* recurser($n-1);             //返回计算结果
    }
    $text=recurser(5);                         //调用函数 recurser
    echo $text;                                //输出变量
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-10.php，查看其运行结果，即可看到如图 6-12 所示的结果。

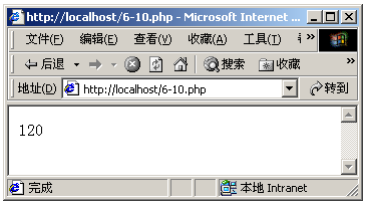

图 6-12　递归调用

【代码解析】程序第 04～07 行是一个 if…else 循环，当 $n 不为 0 时，就执行 else 语句，每次将结果返回给函数 recurser()，使$n 减小 1，再接着执行 recurser()函数，直到$n 的值为 0，返回 1，结束递归。

提示：递归函数的结构十分简练，构造递归函数的关键是找到适当的递归算法和终结条件，因为递推的过程不能无限制地进行下去，必须要有一个结束此过程的条件。

6.3　函数变量

函数对变量的操作非常普遍，形式又多种多样，例如变量的赋值、参数传递、变量函数等。而处理变量问题又会出现变量的作用域，变量的存储方式等问题。本节将针对上述问题来讲解这些变量的处理方式。

6.3.1　变量函数

PHP 支持变量函数的概念，这意味着如果一个变量名后有圆括号，PHP 将寻找与变量的值相同的函数，并且执行找到的函数，这个变量可以被实现函数的回调。

【范例 6-11】定义函数名为 rtext，输出“rtext()”，再定义函数 setname，输出“my name is（name）”。通过函数变量输出这两个函数，并将名字 Jun 输出，其程序如示例代码 6-11 所示。

示例代码 6-11

```
<?php                                          //PHP 开始标记
    function testvar() {                       //声明函数 testvar
        echo "testvar()函数<br>";              //输出 testvar()函数
```

```
04      }
05          function setname($string){                    //声明函数 setname
06          echo "my name is $string" ;                   //输出变量值
07      }
08      $rtext= 'testvar';                                //定义变量
09      $rtext();                                         //变量函数
10      $rtext='setname';                                 //定义变量
11      $rtext("Jun");                                    //调用函数
12  ?>                                                    //PHP 结束标记
```

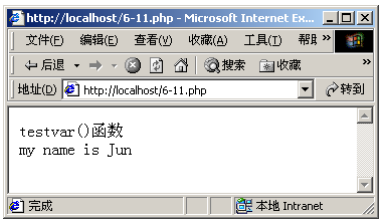

图 6-13　变量函数

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-11.php，查看其运行结果，即可看到如图 6-13 所示的结果。

【代码解析】程序第 02～04 行定义了 testvar()函数。第 08 行将字符串“testvar”赋给变量$rtext。第 09 行变量名后加上圆括号，程序将执行 testvar()函数。同理，采用同样的方式可以调用 setname()函数。

注意：变量函数不能用于语言结构，例如 echo()、print()、unset()、isset()、empty()、include()、require()等语句。

6.3.2　局部变量

每个变量都有其有效范围，一旦离开，变量将发生变化或失去意义，其称为变量的作用域。这就像无线上网，有一定的服务范围，当超出了这个范围就收不到信号。在 PHP 中，按变量作用范围将其分为局部变量和全局变量。

局部变量是指在函数内定义的变量只在本函数中有效，函数以外就不能使用这些变量。同理，函数外定义的变量，函数内也不能使用。例如下面是对局部变量的调用：

```
01  <?php                                          //PHP 开始标记
02      $int=1;                                    //定义变量 int
03      function fun1(){                           //声明函数 fun1
04          $int2=2;                               //定义变量 int2
05          echo  "$int<br>";                  //$int1 为空
06          }
07      fun1();                                    //调用函数
08      echo "$int2<br>";                          //int2 为空
09  ?>                                             //PHP 结束标记
```

以上代码输出为空，因为变量 int1，int2 都为局部变量，变量$int1 是在主程序中定义的，却在函数中调用。而变量$int2 则相反，所以程序什么都不输出。

6.3.3　全局变量

和局部变量相对的是全局变量，能实现跨域调用。在主程序中定义的变量不但在主程序中有效，在函数内也能调用。同样在函数中定义的变量也能被函数以外的程序调用，即全局变量对整个程序都是有效的。定义全局变量的方法是在变量名前冠以“global”关键字。

【范例 6-12】在函数 fun1 外定义变量 int1 的值为 1，在函数内调用 int1 后，设为全局变量。再次调用，将变量 int2 设为全局变量，并将其赋予变量，函数外输出变量 int2 的值。其程序如示例代码 6-12 所示。

示例代码 6-12

```
01  <?php                                          //PHP 开始标记
02      $int1=1;                                   //声明变量
03      function fun1(){                           //定义函数
04          echo "$int1<br>";                      //为空
05          global $int1;                          //建立主函数的变量为全局变量
06          global $int2;                          //建立全局变量
07          $int2=2;                               //注意不能写成 global $int2=2
08          echo "$int1<br>";                      //输出 1
09      }
10      fun1();                                    //调用函数
11      echo "$int2<br>";                          //输出变量的值
12  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-12.php，查看其运行结果，即可看到如图 6-14 所示的结果。

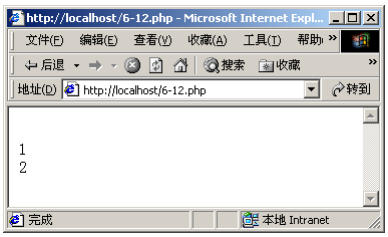

图 6-14　运行结果

【代码解析】代码第 05、06 行声明了两个全局变量，第一个 echo 语句不会输出数值，只进行换行。因为此时$int 还是局部变量，不能在函数中调用，被作为空处理。而建立全局变量后就能在以后的程序中任意调用了。

在函数内调用$int 的值还可以通过 PHP 中定义的$GLOBALS 数组来实现，通过$GLOBALS[“变量名称”]将需要的变量值取出。在用户自定的函数或程序中，可以用“$GLOBALS”数组取出需要的变量。

【范例 6-13】定义两个变量 int1、int2，将其值分别设为 1、2。用$GLOBALS 数组实现调用函数以外的两个变量 int1、int2，其程序如示例代码 6-13 所示。

示例代码 6-13

```
01  <?php                                                    //PHP 开始标记
02      $int1=1;                                             //变量赋值
03      $int2=2;                                             //变量赋值
04      function fun1(){                                     //声明函数 fun1
05          $GLOBALS["int1"]= $GLOBALS["int1"]+$GLOBALS["int2"];
                                                             //调用函数外变量
06      }
07      fun1();                                              //调用函数 fun1
08      echo $int1                                           //输出变量 int1 的值
09  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-13.php，查看其运行结果，即可看到如图 6-15 所示的结果。

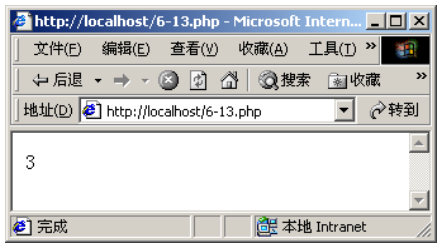

图 6-15　运行结果

【代码解析】上述代码第 05 行用$GLOBALS 数组在函数 fun1 中调用$int1、$int2 的值，将运算结果赋给变量$int1。

注意：全局变量和局部变量是从变量的作用域角度来分的。同时，变量也可以从时间角度来分，可以分为静态存储变量和动态存储变量。

6.3.4　动态变量

动态变量即值不断变化的变量，动态变量的执行特点是当函数执行完毕后，其变量的存储空间将自动被释放。其就如拉煤的火车，运到目的地就卸货，还能回来再拉。

【范例 6-14】定义函数 test，将动态变量 num 的值设为 0，输出变量值后其值加 1，调用函数两次，查看变量 num 值的变化，其程序如示例代码 6-14 所示。

示例代码 6-14

```
01  <?php                                    //PHP 开始标记
02      function test()                      //声明函数 test
03      {
04          $num = 0;                        //定义变量 num
05          echo $num;                       //输出变量 num 的值
06          echo "<br>";
07          $num++;                          //变量加 1
08          }
09      test();                              //调用函数 test
10      test()                               //调用函数 test
11  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-14.php，查看其运行结果，即可看到如图 6-16 所示的结果。

图 6-16　运行结果

【代码解析】程序每次调用时都会将$num 的值设为 0，并输出“0”，将变量加 1 的$num++没有作用，因为一旦退出本函数，则内存空间被释放。

6.3.5 静态变量

动态变量当函数执行完毕后，变量的存储空间自动被释放。如果想利用本次的结果进行后续的计算，就需要将其设置为静态变量。静态变量是在函数执行完毕后，仍能保留其存储空间的变量。静态变量的类型说明符是 static。

【范例 6-15】如果将代码 6-14 中的变量改写为定义静态变量，结果就不一样了，其程序如示例代码 6-15 所示。

示例代码 6-15

```
01  <?php                                  //PHP 开始标记
02      function Test()                    //声明函数
03      {
04      static $num = 0;                   //声明静态变量
05      echo $num;                         //输出变量 num 的值
06      echo "<br>";
07      $num++;                            //变量加 1
08      }
09      Test();                            //调用函数
10      Test()                             //调用函数
11  ?>                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-15.php，查看其运行结果，即可看到如图 6-17 所示的结果。

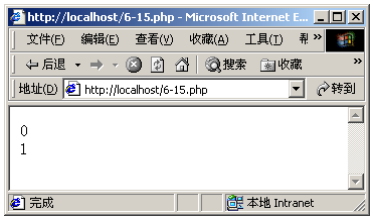

图 6-17 运行结果

【代码解析】上述代码第 04 行将$num 定义为静态变量，这样每次调用 Test()函数时就不会执行$num=0 了，而是每次调用 Test()函数时都会输出$num 的值并加 1。如果是第 4 次调用将会输出 3，然后将变量$num 的值加 1。

6.4 文件包含

编写程序中，经常把一些常用的东西写成简单的文件保存起来，然后在需要的地方调用该文件。PHP 可以通过一些函数重用代码来调用这些文件。常用到的函数主要有 require()语句和 include()语句，下面分别介绍这两个函数。

6.4.1 require 包含文件

函数 require()的功能是将包含文件的内容取代自身的位置，包含的文件应是一个事先编好的文件。如果要将文件包含在不同的目录下，则必须写清文件存放的路径。函数 require()不支持 return 提供的返回值功能。

【范例 6-16】编写一个包含 add()和 sub()函数的文件 require.php，分别计算 2008+2000 和 2008–2000 的值，并通过其他文件调用该文件，并输出结果。其程序如示例代码 6-16 所示。

示例代码 6-16

```
01  <?php                                  //PHP 开始标记
02      require "require.php";             //包含文件
03      echo add();                        //输出调用函数结果
04      echo "<br>";                       //换行
```

```
05      echo sub();                                    //输出调用函数结果
06  ?>                                                 //PHP 结束标记
```

调用的 require.php 文件代码如下：

```
01  <?php                                              //PHP 开始标记
02      function add(){                                //声明函数
03          return 2008+2000;                          //返回函数值
04      }
05
06      function sub(){                                //声明函数
07          return 2008-2000;                          //返回函数值
08      }
09  ?>                                                 //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-16.php，查看其运行结果，即可看到如图 6-18 所示的结果。

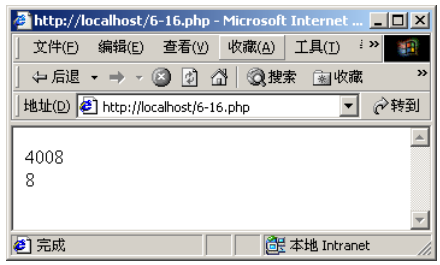

图 6-18　require 包含文件

【代码解析】程序第 02 行调用 require.php 文件，第 03～05 行输出调用的 add()、sub()函数结果。

注意：require()语句实际上只是把目标文件调入当前的脚本文件中，而不是经由 HTTP 或其他的网络协议传输。因此，任何在脚本范围内的变量在执行目标文件时会自动生效。

6.4.2　include 包含文件

include()命令也能将一个外部文件的内容引入到程序中，其允许有返回值。下面通过示例讲解 include()的用法。

【范例 6-17】分别将 2008、2 赋予变量$a、$b，通过调用 include.php 文件计算两个变量之和。其程序如示例代码 6-17 所示。

示例代码 6-17

```
01  <?php                                              //PHP 开始标记
02      $a=2008;                                       //变量赋值
03      $b=2;                                          //变量赋值
04      $c=include("include.php");                     //包含文件
05      echo "以上程序的执行结果是：<br>";              //输出
06      echo "两年以后是 $c 年";                       //输出变量值
07  ?>                                                 //PHP 结束标记
```

调用的 include.php 代码如下：

```
01  <?php                                          //PHP 开始标记
02      echo "include 的用法<br>";                  //输出
03      return $a+$b;                              //返回函数值
04  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-17.php，查看其运行结果，即可看到如图 6-19 所示的结果。

【代码解析】代码第 02、03 行定义两个变量，第 04 行将 include 包含文件的计算结果赋给变量 c。第 05、06 行输出结果。

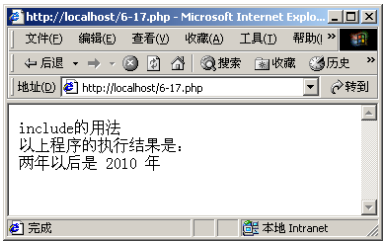

图 6-19　include 包含文件

6.4.3　require 和 include 的区别

require 和 include 的最大区别在于处理引入外部文件的错误信息方式不同，require()遇到错误时会产生错误而停止执行程序。而 include()会产生警告后忽略错误继续执行程序。

【范例 6-18】输出“require 和 include 的区别在于：”，然后通过 require 调用一个不存在的 NOTEXIT.php 文件，再输出“你是否能知道后面的情况”。其程序如示例代码 6-18 所示。

示例代码 6-18

```
01  <?php                                          //PHP 开始标记
02      echo "require 和 include 的区别在于:<br>";  //输出
03      require ("NOTEXIT.php");                   //包含文件
04      echo "你是否能知道后面的情况";               //输出
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-18.php，查看其运行结果，即可看到如图 6-20 所示的结果。

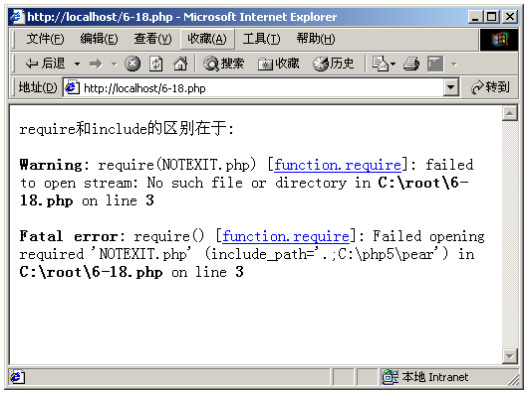

图 6-20　require 编译错误

【代码解析】程序第 03 行利用 require 函数调用了一个不存在的文件 NOTEXIT.php，程序出错，报告错误并停止执行。如果将实例代码 6-18 中的第 03 行换成如下代码：

```
include ("NOTEXIT.php");
```

并将文件保存为：6-18b.php，打开 IE 浏览器，在地址栏输入 http://localhost/6-18b.php，查看其运行结果，即可看到如图 6-21 所示的结果。

提示：通过上面的结果可以看出，include()产生警告后忽略错误，继续执行后面的代码，则输出“你是否能知道后面的情况”。

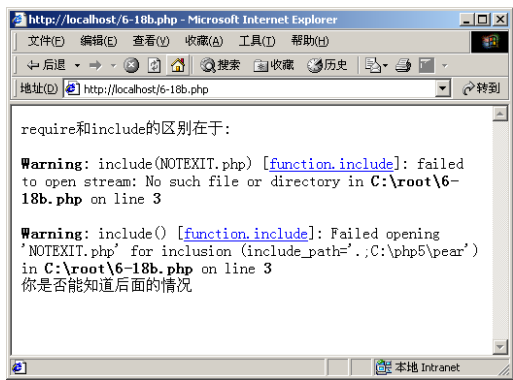

图 6-21　include 编译错误

6.4.4　单次调用文件

require()和 include()在调用文件时没有特别的限制。而 require_once()和 include_once()函数会检查给定代码是否已经调入到了文档内。如果已经调用，则给定代码就不会被再次调入文档。

【范例 6-19】编写 include_once 文件，将关系式$sum=$sum+$i 存放在文件中，利用 for 循环分别用 include_once 函数和 include 函数调用该文件。其程序如示例代码 6-19 所示。

示例代码 6-19

```
01  <?php                                          //PHP 开始标记
02      for ($i=1;$i<=6;$i++){                     //for 循环语句
03          include_once "once.php";               //包含文件
04      }
05      echo $sum;                                 //输出变量值
06      echo "<br>";                               //换行
07      for ($i=1;$i<=6;$i++){                     //for 循环
08          include "once.php";                    //包含文件
09      }
10      echo $sum;                                 //输出变量
11  ?>                                             //PHP 结束标记
```

调用的 once.php 文件代码如下：

```
01  <?php                                          //PHP 开始标记
02      $sum=$sum+$i;                              //表达式
03  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-19.php，查看其运行结果，即可看到如图 6-22 所示的结果。

【代码解析】代码第 02 行利用 for 循环调用 once.php 文件。第 05 行输出调用结果，由于使用的 include_once 函数，则 once.php 文件只调用了一次。第 07 行利用 for 循环通过 include 调用 once.php 文件。第 10 行输出调用结果，include 调用 6 次 once.php 文件，则输出结果为 22。

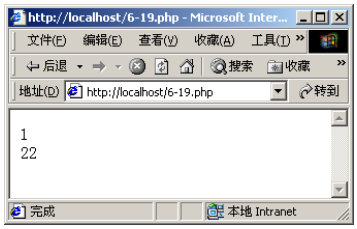

图 6-22　单次调用

6.5　数学函数

编程中少不了要进行数据的处理，如取绝对值、取整、取余弦等，PHP 提供了一些专门的函数进行数据的处理。下面简要介绍几个常用的数据处理函数。

6.5.1　进位函数

PHP 提供了两个进位函数，round()对浮点数进行四舍五入和 ceil()进一取整，这两个函数的用法如下所示。

round()其结构形式为：

```
float round ( float val [, int precision] )
```

参数 val 是要处理的参数，参数 precision 是指定的精度。函数将返回 val 根据指定精度 precision 进行四舍五入的结果。precision 也可以是负数或 0。

ceil()结构形式如下：

```
float ceil ( float value )
```

返回不小于 value 的下一个整数，value 如果有小数部分则进一位。ceil()返回的类型仍然是 float，因为 float 值的范围通常比 integer 要大。

【范例 6-20】利用函数 round()处理数值 100.1、100.5、100.2、10.6534、1241757、5.045，分别将其变为 100、101、100、10.65、1242000、5.05；利用 ceil()处理数值 4.3、9.909，分别将其变为 5、10，其程序如示例代码 6-20 所示。

示例代码 6-20

```
01  <?php                                          //PHP 开始标记
02      echo round(100.1)."<br>";                  //3
03      echo round(100.5)."<br>";                  //4
04      echo round(100.2, 0)."<br>";               //4
05      echo round(10.6534, 2)."<br>";             //1.96
06      echo round(1241757, -3)."<br>";            //1242000
07      echo round(5.045, 2)."<br>";               //5.05
08      echo ceil(4.3)."<br>";                     //5
09      echo ceil(9.909);                          //10
10  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-20.php，查看其运行结果，即可看到如图 6-23 所示的结果。

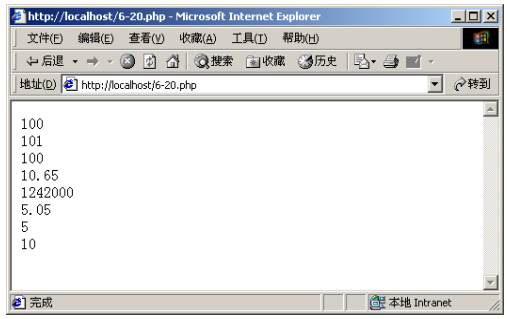

图 6-23　进位函数

【代码解析】代码 02～07 行是利用函数 round()对数据进行四舍五入处理，第 08 行和 09 行是利用函数 ceil()进行进一取整处理。

提示：进位的方式很多，应根据需要选择合适的函数。

6.5.2　进制转化

在计算中常遇到进制的转化问题，例如十进制数转化为二进制数，二进制数转化为十六进制数等，PHP 提供了一组函数可以直接进行转化。函数 decbin()将十进制数转换为二进制数，其语法结构为：

```
string decbin ( int number )
```

函数返回一个字符串，给定的 number 参数用二进制数表示，所能转换的最大数值为十进制数 4294967295，其结果为 32 个 1 的字符串。函数 dechex()将十进制数转换为十六进制数，其语法结构为：

```
string dechex ( int number )
```

函数返回一个字符串，给定的 number 参数用十六进制数表示，所能转换的最大数值为十进制数 4294967295，其结果为“ffffffff”。函数 decoct()将十进制数转换为八进制数，其语法结构为：

```
string decoct ( int number )
```

函数返回一个字符串，参数 number 用八进制数表示，所能转换的最大数值为十进制数 4294967295，其结果为“37777777777”。

【范例 6-21】分别将数 10、46、12、26、15、234 转化为十六进制数、十六进制数、二进制数、二进制数、八进制数、八进制数，其程序如示例代码 6-21 所示。

示例代码 6-21

```
01  <?php                                          //PHP 开始标记
02      echo dechex(10) . "   ";       //转化为十六进制数
03      echo dechex(46)."<br>";                    //转化为十六进制数
04      echo decbin(12) . "   ";       //转化为二进制数
05      echo decbin(26)."<br>";                    //转化为二进制数
06      echo decoct(15) . "   ";       //转化为八进制数
07      echo decoct(234);                          //转化为八进制数
08  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-21.php，查看其运行结果，即可看到如图 6-24 所示的结果。

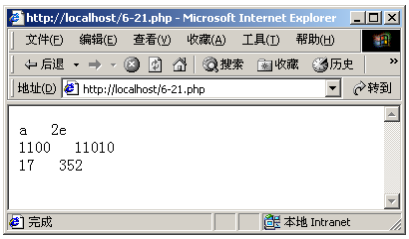

图 6-24　进制转化

【代码解析】上述代码是十进制数同其他进制数之间的转化。其中 是空格标识符，输入一个 就代表一个空格。

6.6　时间函数

在浏览网页时，经常会看到各种样式的时间显示在浏览器上，PHP 提供了大量的时间函数来处理这些时间，本节将介绍几个常用的时间函数。

6.6.1　日期函数 date()

在 PHP 中，提供了 data()函数来获得当前的日期和时间，其结构形式如下：

```
string  date(srting format,[int timestamp]);
```

date()函数返回 format 指定的格式表示时间和日期的字符串，如表 6-1 列出了部分 format 参数及其所代表的意义。参数 timestamp 是可选的，如果没有给出 timestamp 参数，则使用当前时间。

表 6-1　format参数及其意义

参　数	含　义	说　明
a	上午或下午	am 或 pm
A	上午或下午	AM 或 PM
B	互联网时间	000～999
d	月份中的第几天	01～31
D	替换成星期几的英文简称	Mon～Sun
F	用英文表示月份	January～December
g	小时	1～12
G	小时	0～23
h	小时	01～12
H	小时	00～23
i	分钟	00～59
I	是否为夏令时	是为 1，否为 0
j	月份中的第几天	1～31
l	星期几	Sunday～Saturday
L	是否为闰年	闰年为 1，不是闰年为 0
m	用数字表示月份，按两位输出	01～12
M	月份缩写的英文单词	Jan～Dec
n	月份按阿拉伯数字格式输出	1～12
s	秒数	00～59

续表

参　数	含　义	说　明
S	每月天数英文后缀	1st、2nd
t	指定月份的天数	28～31
T	本机设置的时区	如 EST
w	一周的第几天(从 0～6)	0～6
Y	替换成 4 位的年号	如 2008
y	替换成 2 位的年号	如 08
z	一年中第几天	0～366
Z	以秒表示的时区偏差	-43200～43200

【范例 6-22】利用 date()函数输出各种形式的日期。其程序如示例代码 6-19 所示。

示例代码 6-22

```
01  <?php                                                          //PHP 开始标记
02      echo date("F j, Y, g:i a"). "     ";//输出日期
03      echo date("m.d.y"). "     ";          //输出
04      echo  date("j, n, Y"). "     ";       //输出
05      echo  date("Ymd"). "     ";                 //输出
06      echo  date('h-i-s, j-m-y, it is w Day z '). "     ";
                                                                       //输出
07      echo  date('\i\t \i\s \t\h\e jS \d\a\y.'). "     ";
                                                                       //输出
08      echo  date("D M j G:i:s T Y"). "     ";   //输出
09      echo  date('H:m:s \m \i\s\ \m\o\n\t\h'). "     ";
                                                                       //输出
10      echo  date("H:i:s"). "     ";             //输出
11      echo date('l dS \of F Y h:i:s A');                            //输出
12      echo date(DATE_ATOM, mktime(0, 0, 0, 7, 1, 2000));            //输出
13  ?>                                                             //PHP 结束
                                                                       标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-22.php，即可看到如图 6-25 所示的结果。

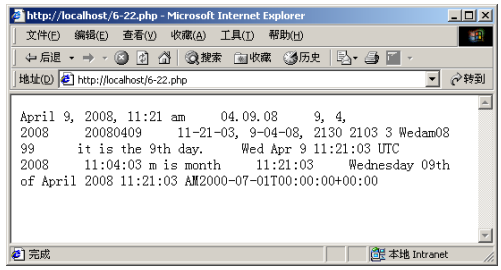

图 6-25　data 函数

【代码解析】代码第 02～11 行是普通的输出，其参数含义都能在表 6-1 中找到。第 12 行是函数的嵌套，mktime 也是时间函数，其用法在以后的章节中将讲到。

6.6.2　获得日期信息

getdate()函数用于取得日期信息，其结构形式如下：

```
array getdate ( [int timestamp] )
```

返回一个 timestamp 指定包含日期信息的数组，其返回的数组元素及其意义如表 6-2 所示。如果没有给出时间戳，则认为是当前本地时间。

表 6-2　返回值

键　名	意　义	范　围
seconds	秒	0～59
minutes	分钟	0～59
hours	小时	0～23
mday	月份中第几天	1～31
wday	星期中第几天	0～6
mon	月份	1～12
year	年份	例如：1999 或 2003
yday	一年中第几天	0～365
weekday	星期	Sunday～Saturday
month	月份	January～December
0	自从 1970 开始至今的秒数。	返回值和系统相关，典型值为-2 147 483 648～2 147 483 647

【范例 6-23】计算从 1970 开始，366 天后的时间参数，其程序如示例代码 6-23 所示。

示例代码 6-23

```
01  <?php                                          //PHP 开始标记
02      $today=getdate(60*60*24*366);              //定义变量
03      print_r($today);                           //输出数组
04  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-23.php，查看其运行结果，即可看到如图 6-26 所示的结果。

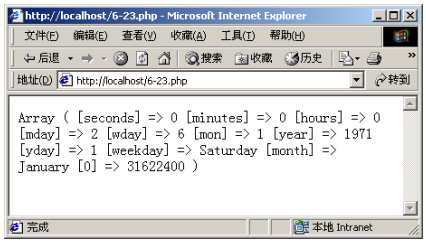

图 6-26　getdate 函数

【代码解析】代码第 02 行将 getdate()函数获得的数组赋给变量$today，第 03 行输出变量的值。

提示：可以直接利用函数 getdate()获得当前的时间参数。

6.6.3　其他时间函数

除了上面介绍的两个函数，PHP 支持的时间函数还有很多，这里简单介绍几个常用的时间函数。gmdate()函数，其函数的结构形式为：

```
string gmdate ( string format [, int timestamp] )
```

同 date()函数完全一样，只是返回的时间是格林威治标准时间（GMT）。mktime()函数，其

函数的结构形式为：

```
int mktime ( [int hour [, int minute [, int second [, int month [, int day [, int year [, int is_dst]]]]]]] )
```

mktime 取得一个日期的 UNIX 时间戳，根据给出的参数返回 UNIX 时间戳。时间戳是一个长整数，包含了从 UNIX 纪元（January 1 1970 00:00:00 GMT）到给定时间的天数。

【范例 6-24】利用 mktime 函数获得该年二月份的天数。其程序如示例代码 6-24 所示。

示例代码 6-24

```
<?php                                                          //PHP 开始标记
    $lastday = mktime(0, 0, 0, 3, 0, 2007);                    //定义时间变量
    echo strftime("Last day in Feb 2007 is: %d", $lastday);    //输出时间
    echo "<br>";                                               //输出换行
    $lastday = mktime(0, 0, 0, 4, -31, 2008);                  //定义时间变量
    echo strftime("Last day in Feb 2008 is: %d", $lastday);    //输出时间
?>                                                             //PHP 结束标记
```

图 6-27　mktime 函数

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-24.php，查看其运行结果，即可看到如图 6-27 所示的结果。

【代码解析】任何给定月份的最后一天都可以被表示为下个月的第 0 天，而不是-1 天，因此程序第 02 行表示的是二月，而不是三月，函数返回 28，2007 年的二月是 28 天，第 05 行同样表示的是 2 月，程序返回 29。

6.7　综合练习

1．定义 order 函数，计算 6 的阶乘。再定义 call 函数，使用变量函数调用 call 函数，输出“通过变量调用函数”。

提示：该习题通过在函数内调用其他函数来熟悉嵌套的调用，还利用返回值实现自身的调用，以求得值的阶乘。

示例代码 6-25

```
<?                                                             //PHP 开始标记
    //使用函数递归实现阶乘功能
    function order($arg)                                       //声明函数
    {
        if ($arg>1)                                            //判断条件
    return $arg * order($arg -1);                              //返回函数值
        return 1;                                              //返回值
    }
    function call()                                            //声明函数
        {
        echo " 通过变量调用函数<br> " ;                        //输出
    }
    $n=6;                                                      //变量赋值
    $temp = order($n);                                         //变量赋值
    echo "$n 的阶乘是$temp<br> " ;                             //输出
    $arg = "call";                                             //变量赋值
    $arg( ) ;                                                  //调用函数
?>                                                             //PHP 结束标记
```

【程序结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-25.php，查看其运行结果，即可看到如图 6-28 所示的结果。

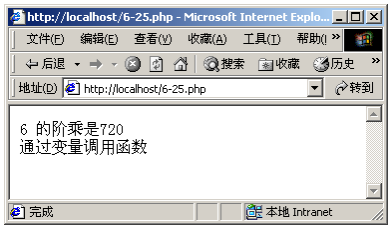

图 6-28　函数套用

2．打印出（-10，10）之间，以 2 为间隔段输出 Atan（x）和 Atan（|x|）的值。

提示：利用 table 标签制成表格形式，利用 for 循环可实现函数值的输出。

示例代码 6-26

```
<?                                                          //PHP 开始标记
    echo "<h4>-10 到 10 的 Atan 函数的值<h4>";              //输出标题字
    print("<table border=\"1\">\n");                        //打印表格
    print("<tr><th>x</th><th>Atan(x)</th><th>Atan(|x|)</th></tr>\n");
                                                            //输出行
    for($x = -10; $x <= 10; $x += 2)                        //for 循环判断语句
    {
        print("<tr>");                                      //开始行
        print("<td>$x</td>");                               //输出 x 值
        print("<td>" . Atan($x) . "</td>");                 //输出 Atan 值
        print("<td>" . Atan(Abs($x)) . "</td>");            //输出 Atan 值
        print("</tr>");                                     //结束行标记
}
    print("</table>");                                      //表格结束
?>                                                          //PHP 结束标记
```

【程序结果】打开 IE 浏览器，在地址栏输入 http://localhost/6-26.php，查看其运行结果，即可看到如图 6-29 所示的结果。

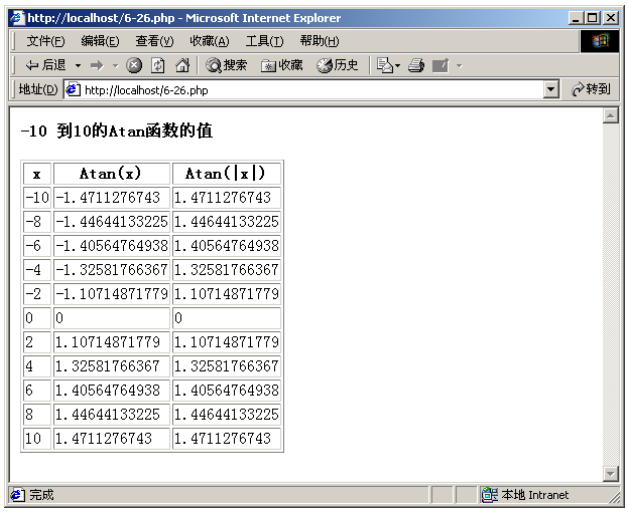

图 6-29　Atan（x）和 Atan（|x|）的值

6.8 本章小结

本章主要介绍了如何创建函数和函数间参数的传递过程，简要介绍了一些数学函数、时间函数等常用函数。本章的内容不多，在前几章的基础之上学习起来很容易。值得注意的是，如果将 PHP 5 中的自定义函数掌握好，对以后的类和对象的学习将会更加得心应手。并能在实际开发中，大大提高程序的简练度，达到事半功倍的效果。

如果读者对本章内容还有疑问，可参考《精通 PHP5 应用开发》（秦涛，曾文玉．北京：人民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民．北京：中国水利水电出版社，2003）。

6.9 习题

一、填空题

1．PHP 自定义函数的关键字是_____。

2．函数的调用一般带有参数的传递问题，参数传递可分为_____和_____两种，PHP 默认的传递方式为_____。

3．_____可以向调用者返回数据，其结果不是立即传递给用户，而是将值传给调用者。

4．在函数内部定义的变量只在本函数内有效，也就是说只有在本函数内才能使用它们，在函数以外是不能使用这些变量的，这类变量称为_____。

5．在主程序中定义的变量，不但在主程序中有效，函数内也能调用；同样在函数中定义的变量也能被函数以外的程序调用，这类变量称为_____。

6．当函数执行完毕后，变量的存储空间将自动被释放出去，这类变量称为_____；在函数执行完毕后，仍能保留其存储空间的变量称为_____。

7．调用函数自身的方法称为_____。

8．require 和 include 的最大区别在于处理引入的外部文件的错误信息方式不同，函数_____遇到错误时会产生错误而停止执行程序，而函数_____会在产生警告后忽略错误继续执行。

9．函数_____和_____会检查给定代码是否早已经插入到了程序内，如果已经插入，则给定代码就不会被再次调用。

10．PHP 提供了两个进位函数：函数_____对浮点数进行四舍五入，函数_____对浮点数进 1 取整。

二、选择题

1．return 语句可以返回（　　）类型的数据。

A．整型　　B．符点型

C．数组　　D．以上都有

2．包含文件，不支持 return 提供的返回值功能的函数是（　　）。

A．require()　　B．include()

C．header()　　D．include_once()

3．选择下面程序的运行结果（　　）。

```
01      $a=2008;
02      function add (&$a){
03          $a=$a+1;
04          echo $a."<br>";
05  }
06      add($a);
07      echo $a;
```

A．2008　　B．2009　　C．2009　　D．编译有误

2008　　2008　　2009

4．选择下面程序的运行结果（　　）。

```
01  $int=1;
02  function num(){
03      $int=$int+1;
04      echo  "$int<br>";
05      }
06  num();
```

A．程序无输出　　B．1　　C．2　　D．以上都不对

5．选择下面程序的运行结果（　　）。

```
01  function sum()
02  {
03  static $num = 0;
04  $num++;
05        echo $num.", ";
06  }
07  sum();
08  sum()
```

A．0，1，　　B．1，1，　　C．1，2，　　D．程序有误

三、简答题

1．简述按值传递和引用传递的区别。

2．简述动态变量和静态变量的区别。

3．简述局部变量和全局变量的区别。

四、编程题

1．编写计算当天消费总额的函数（价格*商品数），通过参数传递计算价格为 5 元，购买 20 个这样的商品所消费的金额，并在页面下方显示当前日期。

2．将一块长方形布料的尺寸（height=50，width=20）存放到文件夹 area.php 文件内，通过函数嵌套计算买这块每平方价格为 2.35 元布料所需的钱数。

第 7 章　数　　组

前面已经讲过整型、字符型等基本数据类型，数组类型也是一种数据类型。不过，数组可以将以前学过的整型、字符型数据存入到数组中去，所以说其是一种较复杂的数据类型，但这种数据类型可以更方便地操作数据。下面就开始学习对数组的操作。

学习本章，可以获得以下知识点：

- 掌握数组的定义、创建、调用；
- 数组的输出；
- 数组的遍历；
- 数组的计算；
- 能对数组进行任意的排序。

7.1 定义数组

数组实际上是一个数据集合，相当于是一个数据容器，把一个个数据按一定的规则存入到容器中。其就像一个旅馆，旅馆内有好多房间，房间按一定的规则编号，供旅客居住。

7.1.1 数组的构成

数组的构成形式如下所示。

```
$数组名[指针]=值
```

- 数组名：是一个数组区别于另一个数组的方式。数组名就像旅店的名字，每个旅店都有自己的名字。
- 键（key）：中括号内为键名，也称索引或标识符。键代表某值在数组中存放的位置，可以通过查询键来获取相应的值。键名就是旅店中各个房间的门牌号，门牌号可以按不同的方式命名。
- 值（value）：等号右边为数组中存放的元素的值。值就是屋内存放的东西了，当想找到某物时，可通过门牌号来查找。

例如：

```
$states[1]=hello
```

数组名为 states，键为 1、值为 hello 的数组。

提示：创建数组有两种方式，一种是给数组变量赋值，另一种是调用函数创建。

7.1.2 使用赋值创建数组

PHP 是一种松散类型的语言，在使用数组前，不需要先声明数组。在创建数组时，不需要指定其大小，而是直接给一个数组变量赋值。

【范例 7-1】将字符串 ASP、PHP、JSP 赋给名为 languages 的数组，然后分别输出这三个元素，其程序如示例代码 7-1 所示。

示例代码 7-1

```
<?php                                    // PHP 开始标记
    $languages[]="ASP";                  //数组元素
    $languages[]="PHP";                  //数组元素
    $languages[]="JSP";                  //数组元素
    echo $languages[0]."<br>",           //输出数组元素
    echo $languages[1]."<br>",           //输出数组元素
    echo $languages[2];                  //输出数组元素
?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-1.php，查看其运行结果，即可看到如图 7-1 所示的结果。

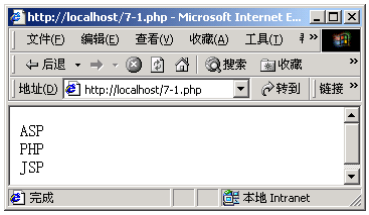

图 7-1　定义数组

【代码解析】代码第 02～04 行相当于定义一个名为$languages 的数组，变量名后面的方括号向 PHP 指出变量$language 是一个数组变量。PHP 将数值存储在以$language[0]开始的连续编号的单元中。

下面是这三个赋值语句产生的数组内容：

```
0=>Per1
1=>PHP
2=>JSP
```

符号=>指出一个标识符（键 key）与一个值相关联。在这个例子中，标识符 0 与“ASP”相关联，键 1 与值“PHP”相关联，键 2 与值“JSP”相关联。

另外，可以编写一个指定标识符值的赋值语句，这样就可以将一个值与特定的数组元素相关联。如下面的例子：

```
$language[0]="ASP";
$language[1]="PHP";
$language[2]="JSP";
```

注意，最后的赋值不包括索引值。PHP 将值“JSP”与数组的下一个连续元素相关联：

```
0=>Per1
1=>PHP
2=>JSP
```

数组元素不需要与连续的标识符相关联，下面的例子说明了这个事实。

```
$language[10]="ASP";
$language[20]="PHP";
$language[]="JSP";
```

上述示例赋值语句产生下面的数组：

```
10=>Per1
20=>PHP
21=>JSP
```

键可以是数值键或关联键，数值键与值没有真正的联系，一般只代表其存放的位置；关联键与值有一定关系。

注意：与前面一样，值“JSP”与这个数组的下一个连续元素相关联。

7.1.3 使用 array()函数创建数组

函数 array()可以用来新建一个数组，其基本结构形式如下：

```
array array(item1 [,item2…[,itemn]])
```

item 即表示数组中的元素，也可以为空。array()函数创建数组时自动给元素分配标识符，标识符从 0 依次增加。

【范例 7-2】使用 array 定义数组，将字符串 apple、banana、orange 作为数组$fruit 的三个元素，并输出元素值，其程序如示例代码 7-2 所示。

示例代码 7-2

```
01      <?php                                          //PHP 开始标记
02          $fruit=array("apple",                      //定义数组
03                   "banana",                         //数组元素
04                   "orange");                        //数组元素
05          echo $fruit[0]."<br>";                     //输出数组元素
06          echo $fruit[1]."<br>";                     //输出数组元素
07          echo $fruit[2]."<br>"                      //输出数组元素
08      ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-2.php，查看其运行结果，即可看到如图 7-2 所示的结果。

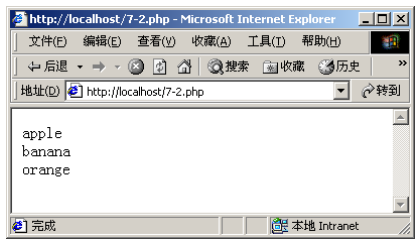

图 7-2　array()函数

【代码解析】代码第 02 行使用 array 函数定义了一个数组，数组内含有三个元素，第 03～05 行分别对这三个元素进行了输出。

7.1.4 键名分配

键名在创建过程能自动分配，但也可以直接指定。其就像店主可以给客人分配房间，客人也可以自己直接指定房间，下面通过示例讲解直接给元素分配键名的方法。

【范例 7-3】用 array()函数定义数组$arr，元素值为（新浪、网易、腾讯、雅虎），对应的键名为（1、2、3、10），其程序如示例代码 7-3 所示。

示例代码 7-3

```
01  <?php                                              //PHP 开始标记
02          $arr = array(1 => "新浪",                  //定义数组
03              2 =>"网易",                            //数组元素
04              3 => "腾讯",                           //数组元素
05              10=> "雅虎"                            //数组元素
06          );
07          echo $arr[1];                              //输出“新浪”
```

```
08          echo "<br>";
09          echo $arr[2];                          //输出“网易”
10          echo "<br>";
11          echo $arr[3];                          //输出“腾讯”
12          echo "<br>";
13          echo $arr[10];                         //输出“雅虎”
14  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-3.php，查看其运行结果，即可看到如图 7-3 所示的结果。

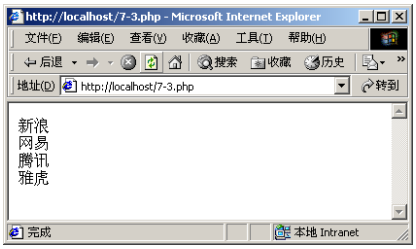

图 7-3　数组

【代码解析】数组里面的数据实际上是按一定顺序排列的，每个数据都有一个 key 对应，这个 key（键名）由自己决定。如果没有给出 key，系统会按序列分配一个键名（key），这里键名“10”直接指定给元素值“雅虎”。

7.1.5　用字符串作键名

PHP 不但可以使用整数作为标识符，也可以使用字符串作为标识符。使用字符串作为标识符的数组称为字符串索引（string-indexed）数组，例如：

```
$language['PHP']="Hi";
$language['JSP']="Med";
$language['ASP']="Low";
```

这些赋值将关联以下标识符和值。

```
PHP=>Hi
JSP=>Med
ASP=>Low
```

PHP 中规定，键名只能是两种类型中的一种：整数（integer）或字符串（string），下面通过实例来讲解使用字符串作为键名的赋值和调用方法。

【范例 7-4】创建数组$arr，将元素新浪、网易、腾讯、雅虎分别赋予键名 a、b、c、0，其程序如示例代码 7-4 所示。

示例代码 7-4

```
01  <?php                                          //PHP 开始标记
02      $arr = array("a" => "新浪",                 //定义数组
03               "b"=>"网易",                       //数组元素
04               "c" => "腾讯",                     //数组元素
05                "雅虎");                          //数组元素
06      echo $arr['a']."<br>";                     //输出“新浪”
07      echo $arr['b']."<br>";                     //输出“网易”
08      echo $arr['c']."<br>";                     //输出“腾讯”
09      echo $arr[0]."<br>";                       //输出“雅虎”
```

```
10  ?>                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-4.php，查看其运行结果，即可看到如图 7-4 所示的结果。

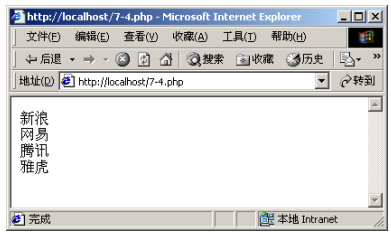

图 7-4　字符串键名

【代码解析】代码第 02～05 行定义了数组，含有 4 个元素，前 3 个键名为字符串。第 06～09 行输出数组元素。

注意：定义字符串要用引号，所以访问数组数据时中括号内的键名一定要用引号。

7.1.6　键名的新建 / 修改

要增加数组数据或修改数据，可以利用与赋值法创建数组相同的方式实现数据的增加或修改。只是在修改键名时，方括号内的键名要与被修改的键名相同，这样就能替换掉原来的键名。

【范例 7-5】创建数组$arr，将元素新浪、网易、腾讯、雅虎，分别赋予键名 a、b、c、0。然后将键名为 a 的元素值改为“PHP 中文社区”，向数组中添加元素值为新浪、百度，键名分别为 e 、 1 ，其程序如示例代码 7-5 所示。

示例代码 7-5

```
01  <?php                                       //PHP 开始标记
02      $arr = array("a" => "新浪",              //定义数组
03                "b"=>"网易",                  //数组元素
04                "c" => "腾讯", "雅虎"         //数组元素
05          );
06          $arr[a] = "PHP 中文社区";            //更改数组元素值
07          $arr['e'] = "新浪";                  //添加数组元素值“新浪”
08          $arr[] = "百度";                     //添加数组元素值“百度”
09      echo $arr['a'] ."<br>"";                 //输出“PHP 中文社区”
10      echo $arr['b'] ."<br>";                  //输出“网易”
11      echo $arr['c'] ."<br>";                  //输出“腾讯”
12      echo $arr['e'] ."<br>";                  //输出“新浪”
13      echo $arr[0] ."<br>";                    //输出“雅虎”
14      echo $arr[1] ."<br>";                    //输出“百度”
15  ?>                                           //PHP 结束标记
```

图 7-5　键名修改

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-5.php，查看其运行结果，即可看到如图 7-5 所示的结果。

【代码解析】代码第 02～05 行定义一个名为$arr 的数组，第 06 行再次定义同一个数组，且键名与上面的元素相同。第 07～08 行定义不同键名的同一数组，第 09～14 行输出数

组元素。

7.1.7 创建多维数组

多维与一维的区别在于多维数组有两个或多个下标，其用法与一维数组大致相同，只是多维数组的操作更为复杂，但功能强大。以二维数组为例，二维数组由一维数组构成，其就像大房子里套有小房子，表示方法为$a[0][0]，表示数组名为 a 的二维数组的第一个元素。下面通过实例讲解创建多维数组的方法。

【范例 7-6】将 32，23，12，54，65，29 6 个数据用二维数组表示。假如命名为$array2，每一个数据分别为：$array2[0][0]=32；$array2[0][1]=23；$array2[0][2]=12；$array2[1][0]=54；$array2[1][1]=65；$array2[1][2]=29，其程序如示例代码 7-6 所示。

示例代码 7-6

```
01  <?php                                                    //PHP 开始标记
02      $array2[0][0]=32;                                    //数组元素
03      $array2[0][1]=23;                                    //数组元素
04      $array2[0][2]=12;                                    //数组元素
05      $array2[1][0]=54;                                    //数组元素
06      $array2[1][1]=65;                                    //数组元素
07      $array2[1][2]=29;                                    //数组元素
08      for ($i=0;$i<=1;$i++){                               //遍历数组
09          for ($j=0;$j<=2;$j++){                           //数组元素
10          echo "\$array2[$i][$j]=".$array2[$i][$j]."   ";
                                                             //数组元素
11     }
12     echo "<br>";                                         //输出换行
13  }
14  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-6.php，查看其运行结果，即可看到如图 7-6 所示的结果。

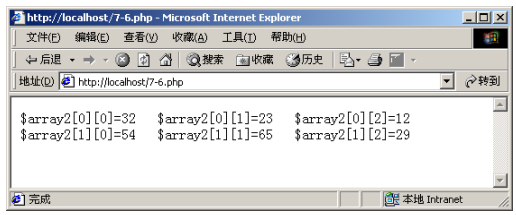

图 7-6 多维数组

【代码解析】代码第 01～07 行定义一个二维数组，第 08～11 行利用嵌套的 for 循环输出二维数组的元素值。

7.2 数组输出与测试

输出数组是指将数组元素输出到浏览器上，输出数组常用的函数有 var_dump()、print_r()等，测试数组是指判断变量是否为数组，下面分别详细讲解数组的输出和测试。

7.2.1 打印变量

var_dump()打印变量的相关信息，其结构形式如下：

```
void var_dump ( mixed $expression [, mixed $expression [, $... ]] )
```

函数 var_dump()数组递归展开数组元素，显示的类型、键名、元素值等。

【范例 7-7】定义一个数组$a，其元素为 1、2 和另一个数组 arry（a，b，c），利用函数 var_dump()输出各元素，其程序如示例代码 7-7 所示。

示例代码 7-7

```
01  <?php                                    //PHP 开始标记
02      $a=array( 1,                         //定义数组
03                2,                         //数组元素
04                array("a",                 //嵌套数组
05                      "b",                 //数组元素
06                      "c")                 //数组元素
07                );
08      var_dump($a);                        //打印数组
09  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-7.php，查看其运行结果，即可看到如图 7-7 所示的结果。

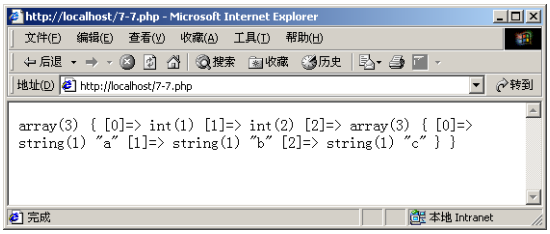

图 7-7　var_dump()函数

【代码解析】代码第 02～07 行定义一个数组，第 08 行输出数组各元素值和对应键名。

注意：为了防止程序直接将结果输出到浏览器中，可以使用输出控制函数来捕获此函数的输出，并把它们保存到一个类型变量中。

7.2.2　输出数组

输出数组常用的另一个函数为 print_r()，该函数的作用是直接输出数组的键名及其对应的值，下面通过实例查看其输出形式。

【范例 7-8】定义一个数组$foo，其元素为：bob、fred、jussi、jouni、egon 和 marliese，利用函数 print_r()输出各元素，其程序如示例代码 7-8 所示。

示例代码 7-8

```
01  <?                                       //PHP 开始标记
02      $foo = array("bob",                  //定义数组
03                   "fred",                 //数组元素
04                   "jussi",                //数组元素
05                   "jouni",                //数组元素
06                   "egon",                 //数组元素
07                   "marliese" );           //数组元素
08      print_r($foo)                        //输出数组
09  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-8.php，查看其运行结果，即可看到如图 7-8 所示的结果。

【代码解析】代码第 02～07 行定义一个$foo 数组，第 08 行输出数组各键名和其对应元素值。

7.2.3 测试数组

函数 is_array()用来判断变量是不是数组，其结构形式为：

```
bool is_array ( mixed $var )
```

检测$var 是否是数组，如果是，则返回 True；否则返回 False。

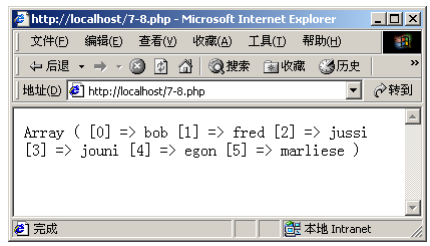

图 7-8 print_r()函数

【范例 7-9】将字符串“I'm an array”，赋给数组$arr，用 is_array()函数判断变量是否为数组，如果是，输出元素值；如果不是，则输出“不是数组”。其程序如示例代码 7-9 所示。

示例代码 7-9

```
01  <?php                                          //PHP 开始标记
02      $arr[] = "I'm an array.";                  //定义数组
03      if(is_array($arr))                         //判断数组
04          {
05              foreach($arr as $val)              //遍历数组
06              {
07          echo $val;                             //输出变量
08              }
09          }
10  else
11  echo "不是数组";                               //输出“不是数组”
12  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-9.php，查看其运行结果，即可看到如图 7-9 所示的结果。

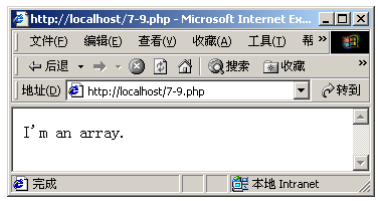

图 7-9 测试数组

【代码解析】代码第 02 行定义数组，第 03 行判断变量是否为数组，第 05 行遍历数组，将数组的值赋给变量$val。

7.3 遍历数组

遍历数组通常需要遍历数组并获得各个键或值（或者同时取得键和值），PHP 中许多函数能完成两项任务，不仅能获取当前指针位置的键或值，还能将指针移向下一个适当的位置。本节将分别介绍这些函数。

7.3.1 foreach 遍历数组

foreach 用来直接遍历数组，其遍历数组的结构形式为：

```
foreach (array_expression as $value) statement
```

array_expression 为要遍历的数组，as 是关键词。每次遍历，当前单元的值被赋给$value，并且数组内部的指针向前移一步。statement 是后续操作，如果有多个语句，可以采用大括号包含。

【范例 7-10】建立一个数组$ur，元素分别是：www.sina.com、www.yahoo.com、www.163.com、www.qq.com、www.baidu.com，对应的键名分别是，新浪、雅虎、网易、腾讯、百度，用 foreach 遍历数组元素，其程序如示例代码 7-10 所示。

示例代码 7-10

```
01  <?php                                                    //PHP 开始标记
02      $url = array('新浪'=>'www.sina.com',                  //定义数组
03                   '雅虎'=>'www.yahoo.com',                //数组元素
04                   '网易'=>'www.163.com',                  //数组元素
05                   '腾讯'=>'www.qq.com',                   //数组元素
06                   '百度'=>'www.baidu.com');               //数组元素
07          foreach (  $url as $link ) {                     //遍历数组
08      echo $link.'<br>';                                   //输出元素值
09      }
10  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-10.php，查看其运行结果，即可看到如图 7-10 所示的结果。

图 7-10 foreach 遍历数组

【代码解析】代码第 02～06 行定义一个数组。第 07 行利用 foreach 遍历数组，循环读取数组$url 中的每个单元，并且每次都把该单元的值赋给变量$link，并且指针向前移一步。第 08 行输出遍历所得的$link 值。

注意：foreach 仅能用于数组，当试图将其用于其他数据类型或者一个未初始化的变量时会产生错误。

7.3.2 foreach 含键名的遍历

在示例代码 7-10 中遍历的$url 数组只取得了数组值，而没取得键名，如果要取得数组键名，可以通过以下形式：

```
foreach (array_expression as $key => $value) statement
```

这种格式遍历数组与第一种一样能够得到数组中的各个元素值，而且还能将当前单元的键名在每次循环中赋给变量$key。

【范例 7-11】定义$url 数组，其元素为（www.sina.com、www.yahoo.com、www.163.com、www.qq.com、www.baidu.com），其对应键名为（新浪、雅虎、网易、腾讯、百度），通过 foreach 函数遍历得到各元素的值和键名，其程序如示例代码 7-11 所示。

示例代码 7-11

```
01  <?php                                                    //PHP 开始标记
02      $url = array('新浪'=>'www.sina.com',                  //定义数组
03                   '雅虎'=>'www.yahoo.com',                //数组元素
04                   '网易'=>'www.163.com',                  //数组元素
05                   '腾讯'=>'www.qq.com',                   //数组元素
06                   '百度'=>'www.baidu.com'                 //数组元素
07                   );
08      foreach ($url as $site => $link ) {                   //遍历数组
09          echo $site,$link ;                                //输出变量
10          echo "<a href=http://$link>$site</a><br>";       //输出变量
11      }
12  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-11.php，查看其运行结果，即可看到如图 7-11 所示的结果。

图 7-11　执行结果

【代码解析】代码第 02～07 行定义一个数组，第 08 行利用 foreach 遍历数组，并将每次循环中将键名赋给变量$site。

7.3.3　each()函数遍历数组

each()函数遍历数组比 foreach 稍复杂，其结构形式为：

```
array each(array input_array)
```

each()函数返回具有 4 个单元的数组，包括当前指针位置的键名和元素值，键名及其对应含义如表 7-1 所示。

表 7-1　键名及其对应含义

键　　名	返 回 值
0	当前标识符-值对的键部分
l	当前标识符-值对的值部分
"key"	当前标识符-值对的键部分
"value"	当前标识符-值对的值部分

如果内部指针越过了数组的末端，则 each()返回 false。

注意：可以使用标识符值 0 或“key"来访问这个标识符-值对的标识符。同样，可以使用标识符值 1 或"value"来访问这个标识符-值对的值。

【范例 7-12】定义$languages 数组，数组元素为（ASP、PHP、JSP），对应键名为（10、20、21），利用 each()函数遍历数组，输出前两个数组元素的值和对应的键名，其程序如示例代码 7-12 所示。

示例代码 7-12

```
<?php                                          //PHP 开始标记
    $languages=array(10=>"ASP",                //定义数组
                     20=>"php",                //数组元素
                     21=>"JSP");               //数组元素
    $leach=each($languages);                   //遍历数组
    echo $leach[key].'<br>';                   //输出键名
    echo $leach[value].'<br>';                 //输出元素值
    $leach=each($languages);                   //遍历数组
    echo $leach[0].'<br>';                     //输出数组元素
    echo $leach[1].'<br>'                      //输出数组元素
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-12.php，查看其运行结果，即可看到如图 7-12 所示的结果。

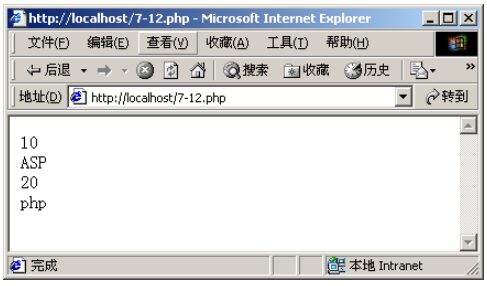

图 7-12　执行结果

【代码解析】程序第 02～04 行定义数组，第 05 行用 each 函数遍历数组，第 06 行输出键名。第 07 行输出元素值，第 08 行遍历第二个元素，第 09 行输出键名，第 10 行输出元素值。

7.3.4　list()函数遍历数组

函数 list()允许在一个单独的赋值语句中向多个变量赋值，其一般格式如下所示：

```
viod list($var1,var2,….$varn);
```

函数 list 把数组中的值一次赋给一组变量，从$varl 到$varn 的每一个指定变量都将从数组 array_value 中接收到值。list()函数使访问 each()返回的键和值变得非常容易，因此 list()函数经常与 each()一起使用。

【范例 7-13】定义$languages 数组，数组元素为（ASP、PHP、JSP），对应键名为（10、20、21），利用 list()函数遍历数组，其程序如示例代码 7-13 所示。

示例代码 7-13

```
01  <?php                                          //PHP 开始标记
02      $languages=array(10=>"ASP",                //定义数组
03                       20=>"PHP",                //数组元素
04                       21=>"JSP");               //数组元素
05          list($key,$value)=each($languages);    //遍历函数
06          echo "key=$key, value=$value";         //输出“key=10，value= ASP”
07          $next=next($languages);                //指针后移
08          echo "<br>next=$next";                 //输出“next=JSP”
09  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-13.php，查看其运行结果，即可看到如图 7-13 所示的结果。

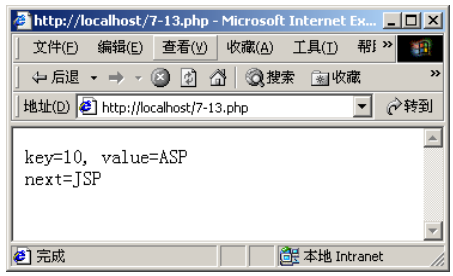

图 7-13　执行结果

【代码解析】代码第 02～04 行定义数组，第 05 行利用 list 遍历数组得到键名和元素值。第 06 行输出遍历键名和元素值，第 07 行将指针移到下一个元素，第 08 行输出所得结果。

提示：list()函数与 array()函数相反，list()函数将一个数组划分为一系列标量值，而 array()函数根据一系列标量值构造一个数组。

7.3.5　for 语句遍历

for 语句的用法前面已经讲过，可以利用其特性实现对数组的遍历输出，下面通过例子讲解 for 语句遍历数组的方法。

【范例 7-14】利用 for 语句遍历含数组元素为北京、天津、上海、深圳的数组$Cities，实现数组的输出，其程序如示例代码 7-14 所示。

示例代码 7-14

```
01  <?                                             //PHP 开始标记
02      $Cities[] = "<B>北京</B>";                 //等同于$Cities[0] = "北京"
03      $Cities[] = "<B>天津</B>";                 //等同于$Cities[1] = "天津"
04      $Cities[] = "<B>上海</B>";                 //等同于$Cities[2] = "上海"
05      $Cities[] = "<B>深圳</B>";                 //等同于$Cities[3] = "深圳"
06      /*
07      ** 统计元素个数
08      */
09      $indexLimit = count($Cities);              //把数组中元素的个数赋给$indexLimit
10      /*
11      ** 打印所有数组
12      */
```

```
13      for($index=0; $index < $indexLimit; $index++)          //循环遍历数组
14      {
15          print("第 $index 个城市是 $Cities[$index]。 <BR>");    //输出数组元素
16      }
17  ?>                                                         //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-14.php，查看其运行结果，即可看到如图 7-14 所示的结果。

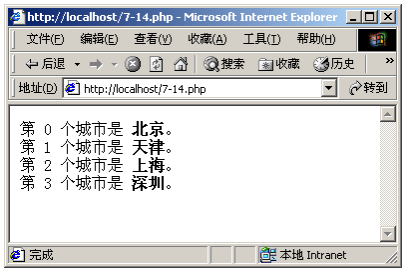

图 7-14　for 语句遍历数组

【代码解析】代码第 02～05 行定义一个含有 4 个元素的数组，第 09 行统计元素的个数，第 13～16 行利用 for 循环输出数组元素。

7.3.6　返回键和键值

key()函数与 current()函数相似，current()返回与当前元素相关联的值，而 key()返回与当前元素相关联的键名，其具体用法通过下列实例讲解。

【范例 7-15】定义$languages 数组，数组元素为（ASP、PHP、Python），对应键名为（10、20、21），利用 key()函数输出当前元素对应的键名，其程序如示例代码 7-15 所示。

示例代码 7-15

```
01  <?                                                  //PHP 开始标记
02      $languages=array(10=>"ASP",                     //声明数组
03                       20=>"PHP",                     //数组元素
04                       21=>"Python");                 //数组元素
05      $current=current($languages);                   //取得数组的元素值
06      $key=key($languages);                           //返回键名
07      echo "current()returned $current";              //输出“current()returned:ASP”
08      echo "<br>key()returned $key";                  //输出“key()returned 10”
09  ?>                                                  //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-15.php，查看其运行结果，即可看到如图 7-15 所示的结果。

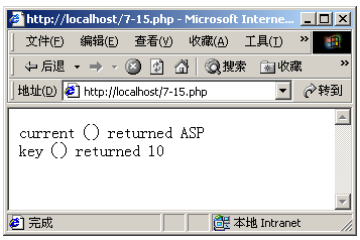

图 7-15　key()函数

【代码解析】代码第 02～04 行定义数组，第 05 行利用 current()函数取得当前数组元素的值，第 06 行利用 key()函数取得当前元素的键名，第 07、08 行分别输出所得结果。

7.3.7 查找数组元素值

array_search()函数查找数组的元素值，其结构如下：

```
mixed array_search(mixed $needle,array $haystack[,boolean $strict])
```

array_search()函数返回的是键名，而不是布尔值。参数$needle 是要查找的值，参数$haystack 是指定的查找对象。找不到时返回 false，找到的元素如果正好是第一个元素，则返回 0。而 PHP 会自动转化为 false，所以需要使用“===”判断返回的值。

【范例 7-16】定义数组$a，其元素值为（a、b、c、d、e、f），并查找数组中是否含有变量$a。如果含有，则返回其位置；如果不含，则输出“在数组 a 中未发现字符'a'”，其程序如示例代码 7-16 所示。

示例代码 7-16

```
01  <?php                                          //PHP 开始标记
02      $a = array("a","b","c","d","e","f");       //定义数组
03      $index = array_search("a",$a);             //查找元素 a
04      if($index === false)                       //判断查找结果
05          echo "在数组 a 中未发现字符'a'";        //输出“在数组 a 中未发现字符'a'”
06      else                                       //不等于 false 的情况
07          echo "Index = $index";                //输出键名
08  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-16.php，查看其运行结果，即可看到如图 7-16 所示的结果。

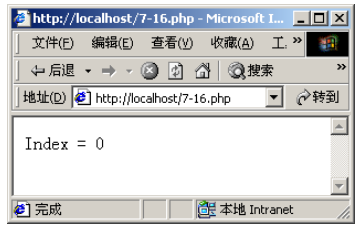

图 7-16 array_search()函数

【代码解析】代码第 03 行查找数组 a 中是否含有变量$a，第 04 行判断返回值，第 05 行和第 07 行输出结果。

7.4 数组计算

数组计算主要是指数组的比较、元素个数的计算、数组的合并等。计算除了利用运算符外，PHP 还提供一些函数，例如利用 count()函数来计算数组中元素的个数。下面分别介绍几种常用的数组计算方法。

7.4.1 计算元素个数

函数 count()用来计算数组中的单元数目或对象中的属性个数，其结果形式如下：

```
int count ( mixed $var [, int $mode ] )
```

函数返回 var 中的单元数目，参数 var 通常是一个数组，任何其他类型都只有一个单元。$mode 是可选的参数，可选的参数自 PHP 4.2.0 起可用。count()对没有初始化的变量返回 0，但对于空的数组也会返回 0。

【范例 7-17】定义数组$a 和$b，元素值分别为（1、3、5）（7、9、11），其对应的键名分别为（0、1、2）（0、5、10），利用 count()函数计算数组元素数目，其程序如示例代码 7-17 所示。

示例代码 7-17

```
01  <?php                                    //PHP 开始标记
02          $a[0] = 1;                       //数组元素
03          $a[1] = 3;                       //数组元素
04          $a[2] = 5;                       //数组元素
05      $result1 = count($a);                //$result == 3
06          $b[0] = 7;                       //数组元素
07          $b[5] = 9;                       //数组元素
08          $b[10] = 11;                     //数组元素
09      $result2 = count($b);                // $result == 3;
10      $result3 = count(null);              // $result == 0
11      $result4 = count(false);             // $result == 1
12      echo "$result1<br>";                 //输出变量 result1 的值
13      echo "$result2<br>";                 //输出变量 result2 的值
14      echo "$result3<br>";                 //输出变量 result3 的值
15      echo "$result4";                     //输出变量 result4 的值
16  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-17.php，查看其运行结果，即可看到如图 7-17 所示的结果。

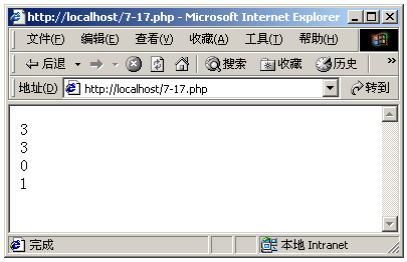

图 7-17　count()函数

【代码解析】代码第 02～03 行定义了一个数组$a。第 04 行计算数组元素个数，第 05～08 行定义了一个数组$b。第 09 行计算数组 b 的元素个数，第 10 行计算 NULL 元素个数。第 11 行计算 false 元素个数，第 12～15 行输出计算所得的值。

注意：count()识别不了无限递归。

7.4.2　count()函数的递归

count()函数结构形式中可选参数 mode 的默认值是 0，如果可选的 mode 参数设为 COUNT_RECURSIVE（或 1），count()函数将递归地对数组计数，这对计算多维数组的所有单元尤其有用。

【范例 7-18】定义数组$food，数组内含有两个数组元素 fruits 和 veggie，其数组元素值分别为（orange、banana、apple）（carrot、collard、pea），利用 count()函数递归计算数组元素数目，其程序如示例代码 7-18 所示。

示例代码 7-18

```
01  <?php                                                          //PHP 开始标记
02      $food = array('fruits' => array('orange',                  //定义数组 food
03                                      'banana',                  //数组元素
04                                         'apple'),               //数组元素
05                'veggie' => array('carrot', 'collard','pea')); //recursive count
06          echo count($food, COUNT_RECURSIVE);                    //输出 8
07      echo "<br>";                                               //换行
08      echo count($food);                                         //输出 2
09  ?>                                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-18.php，查看其运行结果，即可看到如图 7-18 所示的结果。

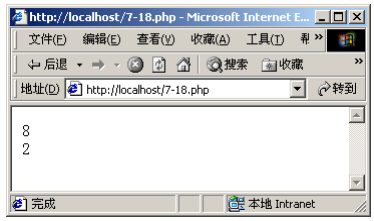

图 7-18　count()函数

【代码解析】代码第 02～05 行定义了一个数组，第 06 行利用递归计算数组$food 获得的元素个数，第 08 行是不利用递归计算获得的元素个数。

7.4.3　数组运算符

前面讲过的运算符在数组中也同样能应用，下面列出较常用的数组运算符，如表 7-2 所示。

表 7-2　数组运算符

例　子	名　称	结　果
$a + $b	合并	$a 和$b 的合并
$a == $b	相等	如果$a 和$b 具有相同的键 / 值对，则为 true
$a === $b	全等	如果$a 和$b 具有相同的键 / 值对并且顺序和类型都相同，则为 true
$a != $b	不等	如果$a 不等于$b，则为 true
$a <> $b	不等	如果$a 不等于$b，则为 true
$a !== $b	不全等	如果$a 不全等于$b，则为 true

提示：+ 运算符把右边的数组附加到左边的数组后面，但是重复的键名不会被覆盖。

【范例 7-19】定义数组$a 和$b，其数组元素分别为（apple、banana）（pear、strawberry、cherry），对应的键名分别为（a、b）（a、b、c），利用数组运算，进行数组的合并，其程序如示例代码 7-19 所示。

示例代码 7-19

```
01      <?php                                          //PHP 开始标记
02          $a = array("a"=> "apple",                  //定义数组
03                  "b"=>"banana");                    //数组元素
04          $b=array("a"=>"pear",                      //定义数组
05                  "b"=>"strawberry",                 //数组元素
06                  "c"=>"cherry");                    //数组元素
07          $c=$a+ $b;                                 //合并数组$a 和$b
08          echo "Union of \$a and \$b: <br>";         //输出合并结果
09          var_dump($c);                              //打印数组
10          $c = $b + $a;                              //合并数组$a 和$b
11      echo  "Union of \$b and \$a:<br>";             //输出合并结果
12      var_dump($c);                                  //打印数组
13  ?>                                                 //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-19.php，查看其运行结果，即可看到如图 7-19 所示的结果。

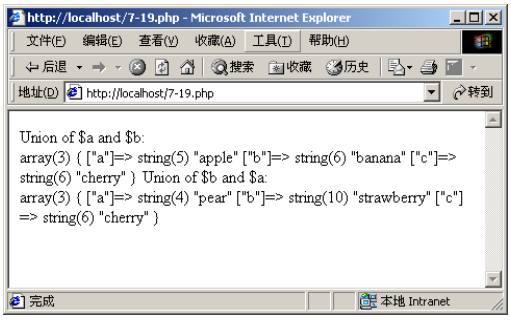

图 7-19　数组运算符

【代码解析】代码第 02～06 行定义了两个数组，第 07 行合并数组$a 和$b 的值，第 11 行合并$b 和$a 的值，通过输出可以看出其差别。

7.4.4　相同元素数组的比较

数组中的单元如果具有相同的键名和值，则无论用哪种运算符比较时都相等，但当元素相同，键名不同时，利用运算符“==”或“===”得到的结果也将不同。

【范例 7-20】数组$a 和数组$b 中的元素分别为（apple，banana）（banana，apple），并分配相同的键名，利用数组运算符比较数组的值，其程序如示例代码 7-20 所示。

示例代码 7-20

```
01  <?php                                          //PHP 开始标记
02      $a=array("apple","banana");                //定义数组
03      $b=array(1=>"banana",                      //定义数组
04              "0"=>"apple");                     //数组元素
05      var_dump($a==$b);                          //bool(true)
06      var_dump($a===$b);                         //bool(false)
07  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-20.php，查看其运行结果，即可看到如图 7-20 所示的结果。

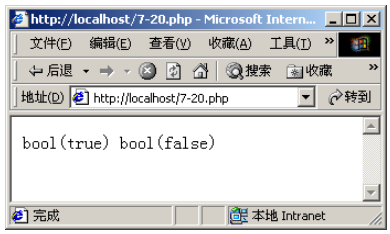

图 7-20　相同元素数组的比较

【代码解析】代码第 02～04 行定义了两个数组，第 05 行比较数组$a 和$b 的元素值和键名，第 06 行比较数组$a 和$b 的元素的值、键名、顺序类型等。

警告：针对数组的比较，最好使用表达式“===”，否则，有可能返回不理想的结果。

7.5　数组排序

数组排序实质上是对数组元素的排序，就是按照元素值或键名的大小进行由小到大或由大到小的排列。PHP 中提供了很多对数组进行排序的函数，下面分别介绍这些函数。

7.5.1　按标识符排序

按标识符排序即是按标识符 ASCⅡ码值的大小，从左至右依次比较其大小，然后按照一定的规则放置元素。在 PHP 中提供了以下几种对数组元素比较的函数：

ksort()：按照数组标识符顺序排列
krsort()：按照数组标识符逆序排列
uksort()：使用用户自定义的比较函数对数组标识符进行排列

【范例 7-21】定义元素值为（lemon、orange、banana、apple）键名为（d、a、b、c）的$fruits 数组，按标识符排列数组元素顺序，其程序如示例代码 7-21 所示。

示例代码 7-21

```
01  <?                                                    //PHP 开始标记
02      $fruits = array("d"=>"lemon",                     //定义数组
03                      "a"=>"orange",                    //数组元素
04                      "b"=>"banana",                    //数组元素
05                      "c"=>"apple");                    //数组元素
06          krsort($fruits);                              //按数组标识符排序
07          foreach ($fruits as $key => $val)             //遍历数组
08      {
09          echo "$key = $val<br>";                       //输出数组元素
10       }
11  ?>                                                    //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-21.php，查看其运行结果，即可看到如图 7-21 所示的结果。

【代码解析】代码第 02～05 行定义了一个数组，第 06 行按照标识符逆序排列，第 07 行遍历将键名赋给变量$key，第 09 行输出变量。

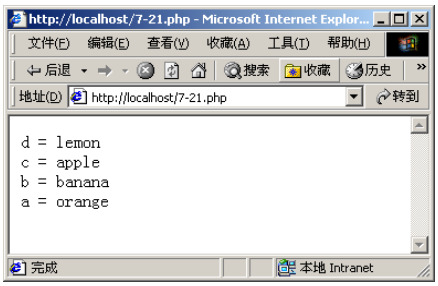

图 7-21　按标识符排序

7.5.2　按元素值排序

PHP 还提供了一组函数，对数组的元素值进行排序。

- asort()：按照由小到大的顺序对数组排序。
- arsort()：按照由大到小的顺序对数组逆序排序。
- uasort()：使用用户自定义的比较函数对数组中的值进行排序。

三个函数按照不同方式对数组元素值进行排序，且都保留其原始键名。

【范例 7-22】定义$languages 数组，数组元素为（ASP、PHP、JSP），对应键名为（10、20、21），利用函数 asort 和 rsort 进行元素排序，比较两者的差异，其程序如示例代码 7-22 所示。

示例代码 7-22

```
01  <?php                                          //PHP 开始标记
02      $languages=array(10=>"ASP",                //定义数组$languages
03               20=>"PHP",                        //数组元素
04               21=>"Python");                    //数组元素
05          asort($languages);                     //正序排列，保留其键名
06          print_r($languages);                   //输出新数组
07          rsort($languages);                     //逆序排列，忽略其键名
08          print_r($languages);                   //
09  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-22.php，查看其运行结果，即可看到如图 7-22 所示的结果。

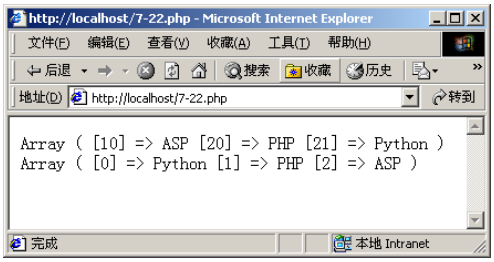

图 7-22　按元素值排序

【代码解析】代码第 02～04 行定义了一个数组，第 05 行对数组$languages 进行正序排列，第 07 行对数组进行逆序排列。

函数 asort()首先比较元素值，保留其键名，然后按从小到大的顺序输出，函数 rsort()比较元素值后，按其从大到小的顺序重新索引输出。

注意：排序是区分大小写的。因为大写字母 H 排在小写字母 e 之前，所以在这个排序输出中，PHP 出现在 Python 之前。

7.5.3 删除原有键名排序

对数组的元素值进行排序，PHP 提供了另外三个函数，其名字和功能如下所示。

- sort()：按照由小到大的顺序对数组排序。
- rsort()：按照由大到小的顺序对数组逆序排序。
- usort()：使用用户自定义的比较函数对数组中的值进行排序。

这三个函数不仅对数组重新排序，还删除$array 数组中原有的键名，为数组中的单元赋予新的键名，当本函数运行结束时，数组单元将被重新排列。

【范例 7-23】定义$fruits 数组，其元素为（lemon、orange、banana、apple），其对应键名为（d、a、b、c）。对数组元素进行由小到大排序，然后输出数组元素，其程序如示例代码 7-23 所示。

示例代码 7-23

```
01  <?                                              //PHP 开始标记
02      $fruits = array("d" => "lemon",             //定义数组
03                      "a" => "orange",            //数组元素
04                      "b" => "banana",            //数组元素
05                      "c" => "apple");            //数组元素
06      sort($fruits);                              //数组排序
07      foreach ($fruits as $key => $val)           //遍历数组
08          {
09          echo "fruits[$key] = $val";           //输出数组
10          }
11  ?>                                              //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-23.php，查看其运行结果，即可看到如图 7-23 所示的结果。

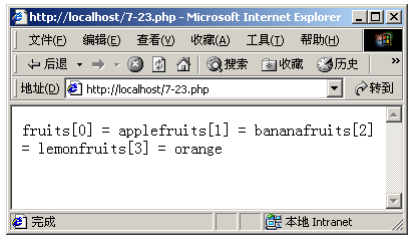

图 7-23　sort()函数

【代码解析】代码第 02～05 行定义了一个数组，第 06 行利用 sort 函数排列数组元素。第 07 行遍历数组，第 09 行输出数组。

7.6　综合练习

1．用 array()函数建立一个包含 5 个数组元素的浮点型一维数组，并分别为其赋值 1，1.2，-2.3，5.4E+3，90.2，用 array()函数建立一个包含 4×2 个数组元素的字符串型二维数组，其元

素为：

北京，上海，天津，重庆

广州，杭州，南京，成都

提示：

一维数组的定义与赋值假定数组名称为$numb，则：$numb=array（1，1.2，-2.3，5.4E+3，90.2）。

二维数组的定义与赋值假定数组名称为 city，则：$city=array（array（北京，上海，天津，重庆），array（广州，杭州，南京，成都）），其程序如示例代码如 7-24 所示。

示例代码 7-24

```
<?php                                                    //PHP 开始标记
    $numb=array(1,1.2,-2.3,5.4E+3,90.2);                 //定义数组$numb
    $city=array(array("北京","上海","天津","重庆"),array("广州","杭州","南京","
成都"));                                                 //定义数组$city
    for ($i=0;$i<=4;$i++){                               //遍历数组
      echo "\$numb[$i]=".$numb[$i]."<br>";             //输出
    }
    echo "<br><br>";
    for ($i=0;$i<=1;$i++){                               //循环 i 值
      for ($j=0;$j<=3;$j++){                             //循环 j 值
      echo $city[$i][$j]."   ";           //输出数组元素
      }
   echo "<br>";                                          //换行
    }
?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-24.php，查看其运行结果，即可看到如图 7-24 所示的结果。

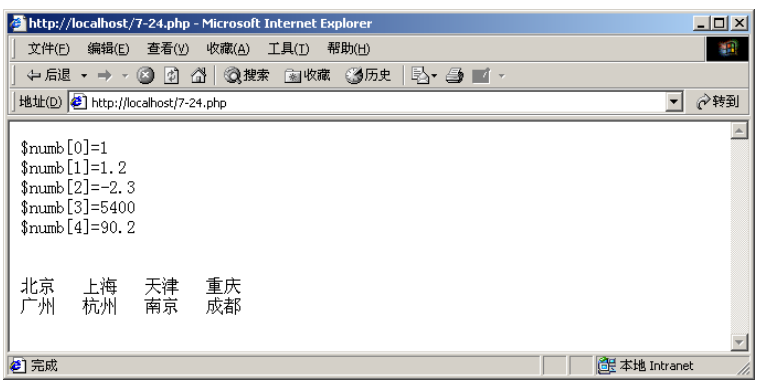

图 7-24　创建多元数组

2．将数字 1，1.2，-2.3，90.25 放进一个数组中，输出其元素值，并计算数组各元素的和，然后将数组元素分别按升序和降序排列。

提示：

- 利用函数 array()创建数组。
- 利用 count()计算数组元素个数。
- 分别利用函数 asort()和 rsort()进行排列。

其程序如示例代码 7-25 所示。

示例代码 7-25

```
01  <?php                                                          //PHP 开始标记
02      $numb=array(1,1.2,-2.3,90.25);                             //初始化数组
03      echo "数组\$numb 的元素个数：".count($numb)."<br>";          //计算数组$numb 的元素个数
04      echo "数组元素值：<br>";  print_r($numb); echo "<br>";       //输出元素值
05      echo "数组的各元素之和：<br>".array_sum($numb)."<br>";       //计算数组$numb 的各元素的和
06      asort($numb);                                              //对数组$numb 进行升序排列
07      echo "排序结果:<br>"; print_r($numb); echo "<br>";          //输出元素值
08      rsort($numb);                                              //对数组$numb 进行降序排列
09      echo "排序结果:<br>"; print_r($numb); echo "<br>";          //输出元素值
10  ?>                                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/7-25.php，查看其运行结果，即可看到如图 7-25 所示的结果。

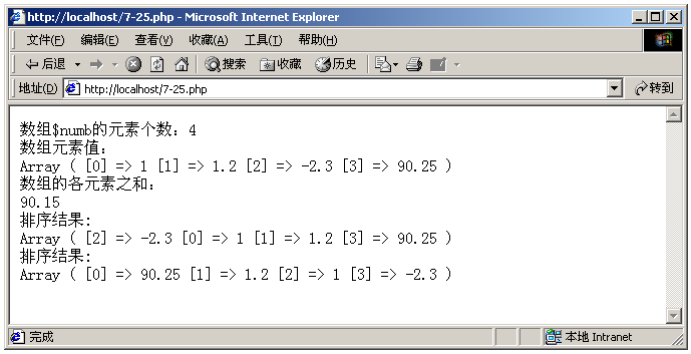

图 7-25　元素排序

7.7　小结

数组的处理函数有着强大、灵活、高效的特点。本章着重介绍了数组的创建、遍历、排序等常用的数组处理操作。掌握好本章内容能给很多问题带来方便。如果读者对本章内容还有疑问，可参考《精通 PHP 5 应用开发》（秦涛，曾文玉．北京：人民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民．北京：中国水利水电出版社，2003）。

7.8　习题

一、填空题

1．数组由_____、_____、_____三部分构成。

2．能作为数组键名的数据类型为_____、_____。

3．函数_____能够判断变量是不是数组。

4．利用函数 print_r()输出数组，该函数能够直接输出数组的_____及其_____。

5．函数_____返回与当前元素相关联的值，而函数_____返回与当前元素相关联的键名。

6．如果数组的元素相同，键名不同，利用_____运算符作判断，返回 false。

7．进行数组排序可以按数组的_____或_____进行排列，排列方式可以是_____，也可以是_____。

8．sort 函数不仅是重新排序，删除$array 数组中原有的_____，而且为数组中的单元赋予新的_____，当本函数运行结束时，数组单元将被_____重新排列。

二、选择题

1．能够计算数组元素个数的函数为（　　）。

A．array　　B．list　　C．count　　D．sort

2．对数组的元素值进行大到小进行排序的是（　　）。

A．sort()　　B．rsort()　　C．usort()　　D．assort()

3．程序返回的数组元素个数为（　　）。

```
01  $animal=arry("horse",
02               "monkey",
03               "lion");
04  $num=eacho($animal)
```

A．1　　B．2　　C．3　　D．4

4．选择下面程序的运行结果为（　　）。

```
01  $numb= array(array(10,15,30),
02           array(10,15,30),
03           array(10,15,30)
04           );
05                echo count($numb, 1);
```

A．3　　B．6　　C．9　　D．12

5．选择下面程序的运行结果为（　　）。

```
$a = array("a","b","c","d");
$index = array_search("a",$a);
if($index == false)
echo "在数组 a 中未发现字符'a'";
else
echo "Index = $index";
```

A．在数组 a 中未发现字符'a'　　B．0

C．1　　D．2

三、简答题

1．简述创建数组的方法。

2．简述 each 遍历数组返回的 4 个单元的数组的意义。

3．列出按元素值排序的 6 个函数。

四、编程题

1．把下列信息存放到一个二维表，然后遍历输出。

张三	李四	王五	赵六
86	90	82	85

2．将学生成绩（87，67，92，65，84，72，78）放进一个数组中，显示其元素个数，然后按降序排列。

第 8 章　字 符 串

通过前面的介绍读者已经对字符串的概念有了初步了解。本章开始对字符串进行具体的操作，如删除特殊字符、大小写转换、比较大小等，下面将详细介绍这些操作。

学习本章，可以获得以下知识点：

- 掌握去除空格和其他特殊字符的方法；
- 能够进行字符串的大小写转换；
- 比较字符串的大小关系；
- 查找与匹配；
- 能够对字符串进行加密。

8.1　去除特殊字符

在某些情况下，字符串中不允许出现空格或者其他一些特殊字符，这时候可以利用 PHP 提供的几个函数来去除字符串中的空格或其他特殊字符。如 trim()函数、ltrim()函数等，下面分别进行介绍。

8.1.1　去除两端空格

trim()函数可以去除字符串开始及结束位置的空格和指定的任意特殊字符，其结构形式如下：

```
string trim (string $str [,string $charlist])
```

trim()函数有两个参数：第一个参数$str 是被处理的字符串，第二个参数$charlist 是要删除的特殊字符。如果第二个参数为空，则视为默认情况。如果想通过该函数过滤掉特殊的字符，可以指定第二个参数，函数最后返回的是一个经过处理的字符串。

【范例 8-1】利用 trim()函数去除字符串“\t\t p h p　\t”中的特殊字符“\t”，然后再去除开始和结束位置的空格，程序如示例代码 8-1 所示。

示例代码 8-1

```
01  <?php                                          //PHP 开始标记
02      $text="\t\t p h p  \t";                    //设置变量$text
03      echo $text;                                //输出变量$text
04      echo "a";                                  //输出字符串 a
05      echo "<br>";                               //输出换行
06      echo trim($text,"\t");                     //去除特殊字符‘\t’
07      echo "a";                                  //输出字符串 a
08      echo "<br>";                               //输出换行
09      echo trim($text);                          //去除左端和右端的空格
10      echo "a";                                  //输出字符串 a
11  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-1.php，查看其运行结果，即可看到运行结果如图 8-1 所示。

【代码解析】代码第 02 行给变量赋值，第 03 行输出变量值，第 06 行去除特殊字符“\t”。

第 09 行去除字符串左端和右端的空格，其作用可以通过源文件（如图 8-2 所示）进行查看。

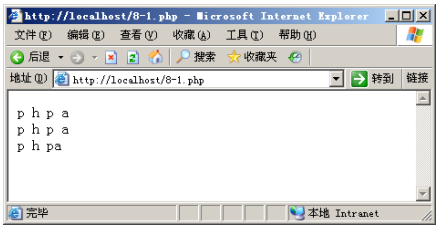

图 8-1　trim()函数

图 8-2　源文件

提示：如果要去除其他特殊字符，如“\n、\b”等，直接将其写在$charlist 的位置即可。

8.1.2　去除左端空格

ltrim()函数用于去除两端的空格，但如果只去除左端的空格呢？PHP 提供了 ltrim()函数用于去除字符串左端的空格和指定的任意特殊字符，其语法结构为：

```
string ltrim ( string $str [, string $charlist] )
```

ltrim()函数有两个参数：第一个参数$str 是被操作的字符串，第二个参数$charlist 是指定的要去除的特殊字符。如果第二个参数为空，则去除的字符同 trim()相同。

【范例 8-2】利用 ltrim()函数去除字符串“　...It is very easy!...”左端指定的特殊字符“...”，其程序如示例代码 8-2 所示。

示例代码 8-2

```
01  <?php                                          //PHP 开始标记
02      $text="  ...It is very easy!...";          //变量赋值
03                                                 //删除左边的空格
04      $left=ltrim($text);                        //去除空格
05      echo $left;                                //输出变量$left
06      echo "<br>";                               //输出换行
07      $lleft=ltrim($text, " . ");                //去除“.”
08      echo $lleft;                               //输出变量$lleft
09  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-2.php，查看其运行结果，即可看到运行结果如图 8-3 所示。

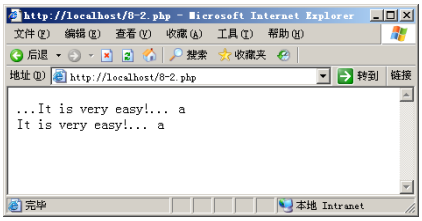

图 8-3　ltrim()函数

【代码解析】代码第 02 行给变量赋值，第 04 行去除字符串左端空格，第 07 行去除左端“.”，第 08 行输出操作结果。

8.1.3 去除右端空格

rtrim()函数的作用与 ltrim()函数相反，rtrim()函数用于去除字符串右端的空格和指定的任意特殊字符，其用法与 ltrim()函数相同。

【范例 8-3】利用 rtrim()函数删除字符串“These are a few words ... ”右端的特殊字符，其程序如示例代码 8-3 所示。

示例代码 8-3

```
01  <?php                                                  //PHP 开始标记
02      $text = "These are a few words ...  ";             //给变量$text 赋值
03      var_dump($text);                                   //打印$text 数组
04      print "<br>";                                      //输出换行
05      $trimmed = rtrim($text);                           //去除右端空格
06      var_dump($trimmed);                                //打印数组
07  ?>                                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-3.php，查看其运行结果，即可看到运行结果如图 8-4 所示。

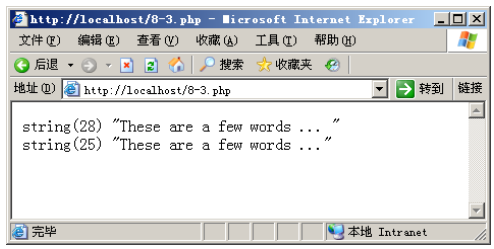

图 8-4 rtrim()函数

【代码解析】代码第 02 行给变量$text 赋值，第 03 行计算字符个数并输出字符串，第 05 行去除右端空格，第 06 行再次计算字符个数并输出字符串。

提示：去除左端和右端特殊字符与除去两端特殊字符的使用方法相同。

8.2 字符串的大小写转换

字符串的大小写转换包括全部转换为大写，或全部转换为小写，还有只将字符串首字母转换为大写等多种转换方式。下面分别讲解这些转换方式。

8.2.1 转换为小写

函数 strtolower()将传入的字符串全部转换为小写，并返回转换后的字符串，该函数的格式如下：

```
String strtolower(string $str)
```

参数$str 为要转换的字符串，将字符串中的所有字符转换为小写。

【范例 8-4】利用 strtolower()函数将字符串“I went To BEI JING.”中的字符全部转换为小写，程序如示例代码 8-4 所示。

示例代码 8-4

```
01  <?php                                          //PHP 开始标记
02      $str= "I went To BEI JING.";               //变量赋值
03      $lstr= strtolower($str);                   //全部转化为小写
04      echo $lstr;                                //输出变量值
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-4.php，查看其运行结果，即可看到运行结果如图 8-5 所示。

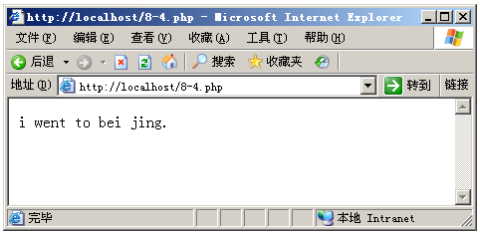

图 8-5　strtolower()函数

【代码解析】代码第 02 行将字符串“I went To BEI JING.”赋给变量$str，第 03 行将字符串中的字母全部转换为小写。

8.2.2　转换为大写

函数 strtoupper()的作用是将字符串中的所有字符全部转换成大写，并返回转换后的字符串，其结构形式如下：

```
String strtoupper (string $str)
```

参数$str 为要换的字符串，将字符串中的所有字符转换为大写。

【范例 8-5】利用 strtoupper()函数将字符串“i want to GO shoping.”中的字符全部转换为大写，其程序如示例代码 8-5 所示。

示例代码 8-5

```
01  <?php                                          //PHP 开始标记
02      $str="i want to GO shoping.";              //变量赋值
03      $ustr= strtoupper($str);                   //全部转换为大写
04      echo $ustr;                                //输出变量
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-5.php，查看其运行结果，即可看到运行结果如图 8-6 所示。

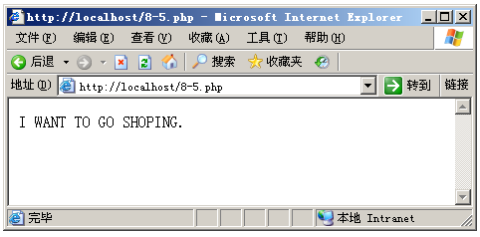

图 8-6　strtoupper()函数

【代码解析】代码第 02 行给变量$str 赋予字符串，第 03 行将变量中的字符全部转换为大写，第 04 行输出转换后的结果。

提示：不能对函数 strtoupper()和 strtolower()进行嵌套使用，因为显示的结果只能有一种，不会既是大写又是小写。

8.2.3 首字符大写转换

函数 ucfirst()的作用是将字符串的第一个字符转换为大写，并返回转换后的字符串，其结构形式如下：

```
string ucfirst (string $str)
```

参数$str 为要转换的字符串，将该字符串中的第一个字符转换为大写。

【范例 8-6】利用 ucfirst()函数将字符串“hello world!”中的首字符转换成大写，其程序如示例代码 8-6 所示。

示例代码 8-6

```
01  <?php                                          //PHP 开始标记
02      //原来首字符小写
03      $foo='hello world!';                       //声明变量
04      $foo=ucfirst($foo);                        //将首字符转化为大写
05      echo $foo. "<br>";                         //输出变量 foo
06      //首字符大写，则不改变
07      $fo='Hello PHP!';                          //声明变量
08      $fo=ucfirst($fo);                          //将首字符转化为大写
09      echo $fo. "<br>";                          //数组变量 fo
10  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-6.php，查看其运行结果，即可看到运行结果如图 8-7 所示。

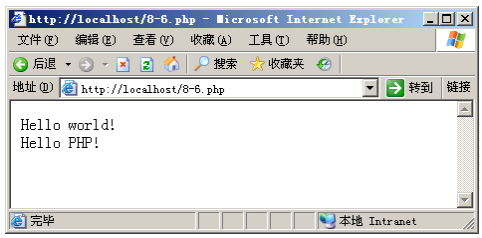

图 8-7 ucfirst()函数

【代码解析】代码第 03 行和第 07 行分别给变量$foo、$fo 赋予字符串，第 04 行和第 08 行将字符串中的第一个字符改成大写，第 05 行和第 09 行输出字符串。

8.2.4 单词首字符大写转换

函数ucwords()的作用是将字符串中的每个单词的首字符转换为大写，其结构形式与ucfirst()函数相同。下面通过示例演示函数 ucwords()的用法。

【范例 8-7】利用函数 ucwords()将字符串“WELCOME to BEIJING”的每个单词的首字符转换为大写，其余字符为小写的字符串，其程序如示例代码 8-7 所示。

示例代码 8-7

```
01  <?php                                          //PHP 开始标记
02      $text='WELCOME to BEIJING';                //变量赋值
03      $utext=ucwords($text);                     //首字符为大写
04      echo $utext . "<br>";                      //输出变量值
```

```
05                                                          //函数的套用
06      $ultext=ucwords(strtolower($text));                 //首字符为大写
07      echo $ultext                                        //输出变量值
08  ?>                                                      //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-7.php，查看其运行结果，即可看到运行结果如图 8-8 所示。

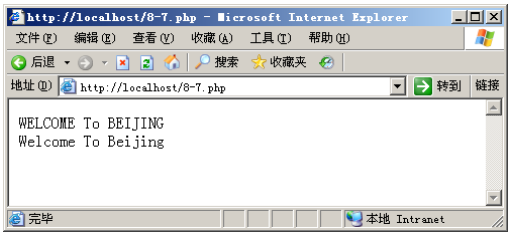

图 8-8　ucwords()函数

【代码解析】代码第 02 行给变量$text 赋值。第 03 行将字符串中每个单词的首字符转换成大写。第 06 行先将字符串中的所有字符转换为小写，然后将每个单词的首字符转换成大写。

8.3　比较字符串

字符串的比较可通过表达式实现，另外 PHP 还提供了一些函数用于字符串的比较，如 strcmp()函数是用于区分大小写的字符串的比较，strcasecmp()函数用于不区分大小写的字符串的比较。下面详细介绍字符串的比较方式。

8.3.1　比较运算符

一般用运算符“!=”或“==”比较两个对象是否相等，之所以说两个对象，是因为比较的双方不一定全部为字符串，也可以为其他数据类型。下面通过示例，比较两个字符串的大小。

【范例 8-8】利用比较运算符比较字符串“joe”和“jerry”的大小，其程序如示例代码 8-8 所示。

示例代码 8-8

```
01  <?php                                                  //PHP 开始标记
02      $a = "joe";                                        //定义变量
03      $b = "jerry";                                      //定义变量
04      if ($a != $b)                                      //判断变量是否相等
05          {
06              echo "不相等";                             //输出“不相等”
07          }
08      else                                               //如果相等的情况
09          {
10              echo "相等";                               //输出“相等”
11          }
12  ?>                                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-8.php，查看其运行结果，即可看到运行结果如图 8-9 所示。

【代码解析】代码第 02、03 行给两变量赋予字符串。第 04 行通过 if 条件句判断两变量是

否相等，并根据结果输出不同的结果。

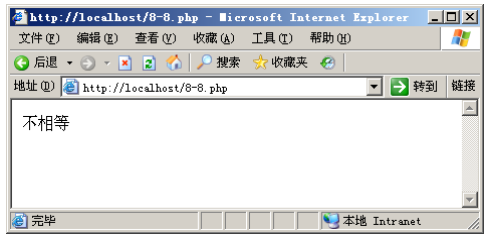

图 8-9　比较运算符

8.3.2　区分大小写字符串的比较

PHP 里还提供这样一组用于字符串比较的函数：strcmp()，strcasecmp()，strncasecmp()，strncmp()，下面对它们分别进行介绍。

函数 strcmp()用于区分大小写（即大小写敏感）的字符串的比较，其结构形式为：

```
int strcmp (string $str1, string $str2)
```

strcmp()函数内的两个字符串按照字节的 ASCII 码值进行比较。如果前者比后者大，则返回大于 0 的整数。如果前者比后者小，则返回小于 0 的整数。如果二者相等，则返回 0。

【范例 8-9】利用函数 strcmp()比较字符串“abcdd”和“aBcde”，“abCde”和“abcde”的大小关系，程序如示例代码 8-9 所示。

示例代码 8-9

```
01  <?php                                          //PHP 开始标记
02      echo strcmp("abcdd", "aBcde");             //返回 1，比较的是 "b"和"B"
03      echo "<br>";                               //换行
04      echo strcmp("abCde", "abcde");             //返回-1，比较的是 "C"和"c"
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-9.php，查看其运行结果，即可看到运行结果如图 8-10 所示。

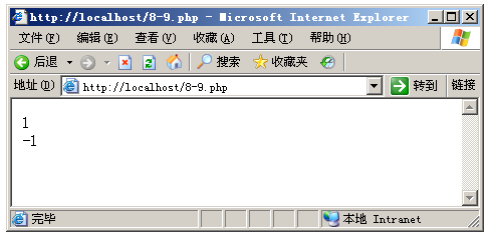

图 8-10　strcmp()函数

【代码解析】代码第 02 行比较字符串 abcdd 和 aBcde 的大小，第一个字符相同，比较第二个字符，由于小写字符的 ASCII 码值大于所有大写字符的 ASCII 码值，所以返回 1。第 04 行比较字符串 abCde 和 abcde 的大小，情况与第 02 行一样，结果返回-1。

8.3.3　不区分大小写字符串的比较

函数 strcasecmp()用于不区分大小写的字符串的比较，其结构形式为：

```
int strcasecmp (string $str1, string $str2)
```

strcasecmp()函数内的两个字符串按照字节的 ASCII 码值进行比较，如果前者比后者大，则返回大于 0 的整数；如果前者比后者小，则返回小于 0 的整数；如果两者相等，则返回 0。其用法与 strcmp()函数相同。

【范例 8-10】利用 strcasecmp()函数比较范例 8-9 中的字符串间的关系，其程序如示例代码 8-10 所示。

示例代码 8-10

```
01  <?php                                          //PHP 开始标记
02      echo  strcasecmp("abcdd", "aBcde");        //返回 -1,比较的是 "b"和"B"
03      echo "<br>";                               //换行
04      echo  strcasecmp("abCde", "abcde");        //返回 0
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-10.php，查看其运行结果，即可看到运行结果如图 8-11 所示。

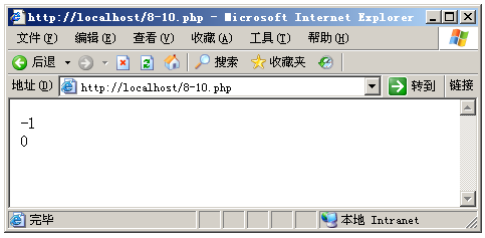

图 8-11　Strcasecmp()函数

【代码解析】代码第 02 行比较字符串“abcdd”和“aBcde”的大小，由于不区分大小写，所以前 4 个字符相同。比较第 5 个字符，d 字符的 ASCII 码值小于 e 字符的 ASCII 码值，所以返回-1。第 04 行比较字符串“abCde”和“abcde”的大小，由于不区分大小写，所以结果返回-1。

8.3.4　选择性的比较字符串

strncmp()函数用于字符串选择性的比较，其结构形式为：

```
int strncmp ( string $str1, string $str2 , int $len)
```

参数$str1 和$str2 为要比较的字符串，从字符串的开头开始比较，第三个参数设置要比较的长度。

【范例 8-11】利用 strncmp()函数比较字符串“abcdd”和“aBcde”，“abcdE”和“abcde”中前三个字符串间的关系，其程序如示例代码 8-11 所示。

示例代码 8-11

```
01  <?php                                          //PHP 开始标记
02      echo strncmp("abcdd", "aBcde", 3);         //返回 1 (>0), 比较了 abc 和 aBc
03      echo "<br>";                               //换行
04      echo strncmp("abcdE", "abcde", 3);         //返回 0，比较了 abc 和 abc
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-11.php，查看其运行结果，即可看到运行结果如图 8-12 所示。

【代码解析】代码第 02 行比较字符串“abcdd”和“aBcde”中前三个字符的关系，因为 b 的 ASCII 码值大于 B 的 ASCII 码值，所以返回 1。第 04 行比较字符串“abcdE”和“abcde”

前三个字符的大小关系，因为字符相同，所以返回 0。

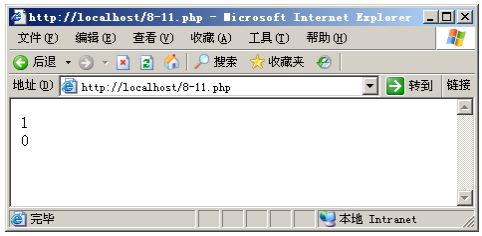

图 8-12　strncmp()函数

8.3.5　不区分大小写选择性的比较

strncasecmp()函数用于不区分大小写的字符串的比较，其结构形式为：

```
int strncasecmp ( string $str1, string $str2, int $len )
```

参数$str1 和$str2 为要比较的字符串，从字符串的开头开始比较，第三个参数设置要比较的长度。

【范例 8-12】利用 strncasecmp()函数比较字符串“abcdd”和“aBcde”，“abcdE”和“abcde”中前三个字符串间的关系，程序如示例代码 8-12 所示。

示例代码 8-12

```
01  <?php                                          //PHP 开始标记
02      echo strncasecmp("abcdd", "aBcde", 3); //返回 0 (>0), 比较了 abc 和 aBc
03      echo "<br>";                               //换行
04      echo strncasecmp("abcdE", "abcde", 3); //返回 0 比较了 abc 和 abc
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-12.php，查看其运行结果，即可看到运行结果如图 8-13 所示。

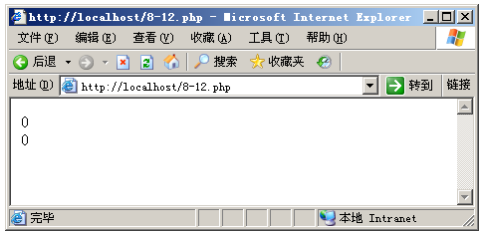

图 8-13　strncasecmp()函数

【代码解析】代码第 02 行比较字符串“abcdd”和“aBcde”中前三个字符的关系，因为不区分大小写，所以字符串相等，返回 0。第 04 行比较字符串“abcdE”和“abcde”前三个字符的大小关系，因为字符相同，所以返回 0。

8.4　查找与匹配

查找一般指查看字符串中是否含有某个或某段字符，PHP 里用于查找或者匹配的函数很多，可以查找字符出现的次数，查找其第一次出现的位置等，下面分别介绍实现这些功能的函数。

8.4.1 查找字符串

strstr()函数和 strrchr()用于查找字符串，两个函数的格式分别是：

```
String  strstr(string $haystack,string $needle)
String  strrchr(string $haystack,string $needle)
```

参数$haystack 为母字符串，即被查找的字符串，参数$needle 为子字符串，即要查找的字符串。

两个函数除了函数名不同，使用方法完全相同，但是其作用略有不同。strstr()函数用来查找子字符串在母字符串中第一次出现的位置，并返回从此位置开始到母字符串结束的部分。strrchr()函数查找子字符串在母字符串中最后一次出现的位置，并返回从此位置开始到母字符串结束的字符串。

注意：如果找不到子字符串，则返回空，因此两个函数可以用来判断一个字符串中是否含有另外一个字符串。

【范例 8-13】查找字符串中“I love iwind too”是否含有“iwind”。查找字符串“php 基础教程里的 ph，缺了字母”中是否含有“p”，并返回最后一个字符 p 后的所有内容。其程序如示例代码 8-13 所示。

示例代码 8-13

```
01  <?                                                  //PHP 开始标记
02      $needle = "iwind";                              //定义变量
03      $str = "I love iwind too";                      //定义变量
04      if (strstr($str, $needle)){                     //判断查找结果
05          echo strstr($str, $needle)."<br>";          //输出查找结果
06          echo "里面有 iwind<br>";                    //输出“里面有 iwind”
07          }
08      else                                            //如果查找结果为 false
09      {
10          echo "里面没有 iwind<br>";                  //输出“里面没有 iwind”
11      }
12      $needle = "p";                                  //定义变量
13      $str1 = "php 基础教程里的 ph，缺了字母";        //定义变量
14      if (strrchr ($str1, $needle)) {                 //判断查找结果
15          echo strrchr ($str1, $needle)."<br>";       //输出查找结果
16          echo "里面有 p";                            //输出“里面有 p”
17          }
18      else                                            //如果查找结果为 false
19      {
20           echo "里面没有 p<br>";                     //输出“里面没有 p”
21      }
22  ?>                                                  //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-13.php，查看其运行结果，即可看到运行结果如图 8-14 所示。

【代码解析】代码第 02、03 行给变量赋值，第 04 行和第 14 行判断字符串变量$str 中是否含有$needle 中的字符串，第 05 行和第 15 行输出返回字符串。

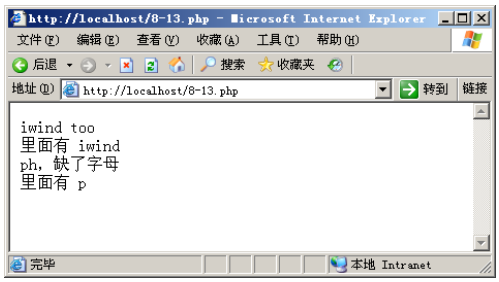

图 8-14　函数 strstr()和 strrchr()

8.4.2　查找字符串出现次数

函数 substr_count()用于查找字符串出现的次数，其结构形式为：

```
int substr_count(string $haystack,string $needle[,int $offset [,int $length]])
```

substr_count()用来统计参数$needle 在另一个参数$haystack 中出现的次数。可选参数为$offset 和$length，分别表示要查找的起点和长度，该函数返回值是一个整数。

【范例 8-14】计算字符串“This is a test”的长度，利用 substr_count()函数查找“is”字符串中出现的次数。再分别从第 4 个字符开始查找“is”的出现次数和第 4 个字符后面的 3 个字符中是否出现“is”。其程序如示例代码 8-14 所示。

示例代码 8-14

```
01  <?php                                                    //PHP 开始标记
02      $text = 'This is a test';                            //定义变量
03      echo strlen($text)."<br>";                           // 14
04      echo substr_count($text, 'is')."<br>";               // 2
05      echo substr_count($text, 'is', 3)."<br>";            //1
06      echo substr_count($text, 'is', 3, 3)."<br>";         //0
07  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-14.php，查看其运行结果，即可看到运行结果如图 8-15 所示。

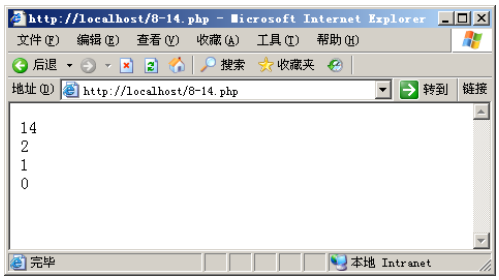

图 8-15　substr_count()函数

【代码解析】代码第 03 行计算字符串长度并输出结果；第 04 行统计字符串“is”在变量$test 中出现的次数，第 05 行从第 4 个字符开始统计“is”出现的次数。第 06 行查找第 3 个字符后面的 3 个字符中是否出现“is”。

注意：offset 的值是从 0 开始的而不是从 1 开始的。

8.4.3 查找最后一次的位置

函数 strrpos()查找字符串的位置，strrpos()函数的格式为：

```
Int strrpos(string $haystack,mixed $needle [,int $offset])
```

strrpos()函数返回字符$needle 在字符串$haystack 中最后一次出现的位置，参数$needle 只能是一个字符，而不能是一个字符串。如果提供一个字符串，函数也只会取字符串中的第一个字符使用，其他字符无效。参数 offset 设置查找字符串的长度，是用来限制查找的范围。

【范例 8-15】利用 strrpos()函数查找字符串“php”中是否含有“p”字符，再查找是否含有“b”字符。程序如示例代码 8-15 所示。

示例代码 8-15

```
01  <?php                                                    //PHP 开始标记
02      $mystring=php;                                       //定义变量
03      $pos = strrpos($mystring, "p");                      //查找字符 p 的位置
04      if ($pos === false) {                                //如果查找结果为 false
05          echo "not found...<br>";                       //输出“not found”
06      }
07      else {                                               //结果为 True 的情况
08          echo "The string was  found in the string '$mystring'<br>";
                                                             //输出
09      }
10      $mystring=php;                                       //定义变量
11      $pos = strrpos($mystring, "b");                      //查找字符 b 的位置
12      if (is_bool($pos) && !$pos) {                        //判断结果
13          echo "not found...";                           //输出“not found”
14      }
15      else {                                               //结果为 True 的情况
16          echo "The string was  found in the string '$mystring'<br>";
                                                             //输出
17      }
18  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-15.php，查看其运行结果，即可看到运行结果如图 8-16 所示。

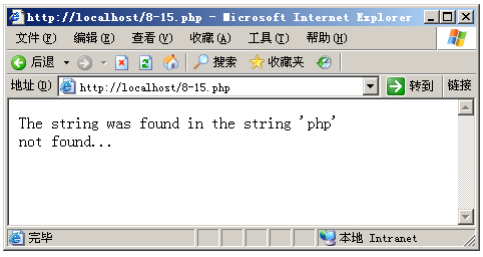

图 8-16 strrpos()函数

【代码解析】代码第 03 行统计字符“p”是否在变量$mystring 中出现，第 04 行判断结果。第 11 行判断字符“b”是否在变量$mystring 中出现，第 12 行判断结果。

8.4.4 查找第一次的位置

函数 strpos()查找字符串的位置，其形式结构为：

```
int strpos(string $haystack,$needle [,int $offset])
```

strpos()函数与 strrpos()函数仅一字之差，但功能相差很大。strpos()函数的 needle 参数允许使用一个字符串，而且返回这个字符串在 haystack 中第一次出现的位置，而不是最后一次。

【范例 8-16】利用 strpos()函数查找字符串“PHP”中是否含有“P”字符。如果找不到，则输出“The string p was not found in the string （字符串）”。如果找到，输出“The string '$findme' was found in the string（字符串）and exists at position（位置）”，并返回该字符出现的位置。其程序如示例代码 8-16 所示。

示例代码 8-16

```
01  <?php                                                          //PHP 开始标记
02      $mystring = 'PHP';                                         //定义变量
03      $findme = 'P';                                             //定义变量
04      $pos = strpos($mystring, $findme);                         //查找字符串位置
05      if ($pos === false) {                                      //判断查找结果
06      echo "The string '$findme' was not found in the string '$mystring'";
                                                                   //输出变量值
07      }
08      else {                                                     //如果结果为真
09          echo "The string '$findme' was found in the string '$mystring'";
                                                                   //输出变量值
10          echo " and exists at position $pos";                   //输出变量值
11      }
12  ?>                                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-16.php，查看其运行结果，即可看到运行结果如图 8-17 所示。

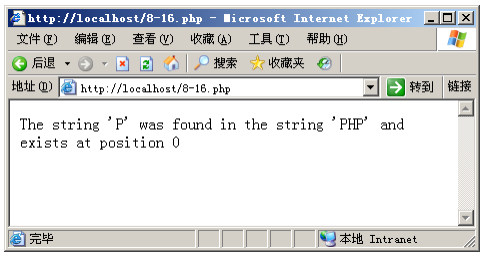

图 8-17　strpos()函数

【代码解析】代码第 04 行查找“P”在变量$mystring 中的出现位置，并将结果赋给变量$pos。第 05 行判断结果，第 10 行输出字符“P”在变量$mystring 中的出现位置。

8.5　替换与分隔

替换就是指将一个字符串的一部分进行改变，使之成为一个新的字符串，以满足新的要求，PHP 里提供专门的函数进行替换，下面分别介绍这些函数。

8.5.1　字符串的替换

str_replace 函数用于字符串的替换，其结构形式为：

```
string  str_replace ( string $str1, string $str2, string $str3 )
```

参数为要被替换的字符串，参数$str2 为新字符串，参数$str3 为原字符串。函数的作用是将原字符串$str3 中字符串$str1，用新字符串$str2 替换。

【范例 8-17】利用 str_replace()函数将字符串“I love ASP，ASP said”中的“ASP”替换为“PHP”，再将字符串“Line 1\nLine 2\rLine 3\r\nLine 4\n”中的字符串“\r\n、\r、\n”替换为“
”，其程序如示例代码 8-17 所示。

示例代码 8-17

```
01  <?php                                                    //PHP 开始标记
02      echo str_replace("ASP", "PHP", "I love ASP, ASP said");
                                                             //将输出 "I love PHP,
                                                             PHP said"
03      echo "<br>";                                         //换行
04      $str = "Line 1\nLine 2\rLine 3\r\nLine 4\n";         //定义变量 str
05      $order = array("\r\n", "\n", "\r");                  //定义变量 order
06      $replace = '<br />';                                 //定义变量 replace
07      $newstr = str_replace($order, $replace, $str);       //字符替换
08      echo $newstr;                                        //输出结果
09  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-17.php，查看其运行结果，即可看到运行结果如图 8-18 所示。

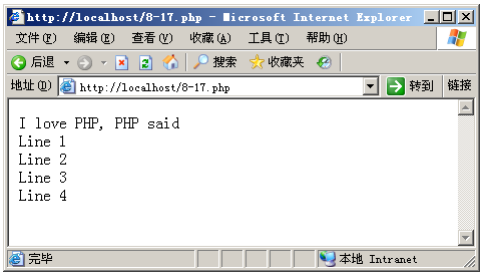

图 8-18　str_replace()函数

【代码解析】代码第 02 行将字符串中 ASP 替换为 PHP，第 03～06 行定义变量 str，第 07 行将字符串变量$str 中的“\r\n、\r、\n”替换为
。

注意：str_replace 还可以实现多对一、多对多的替换，但无法实现一对多的替换。

8.5.2　部分替换

函数 substr_replace()用于替换字符串的一部分，其结构形式如下：

```
string  substr_replace ( string $str1, string $str2, int $str3 [int $n)
```

参数$str1 为原字符串，参数$str2 为替代的字符串，参数$str3 为开始替换的位置，参数$n 为要替换的长度，此参数可选。

注意：开始替换的位置从 0 开始计算，要替换的长度是可选的，要替换的长度应该小于原字符串的长度。

【范例 8-18】利用 substr_replace()函数将字符串“abcdefgh”替换成“abcDEF”和“abcDEFfgh”，对字符串“AAA5BB:/CCC55D/”进行以下处理：前面加上字符串“EEEFF”，再将“CCC55D”替换为“EEEFF”，最后将“CCC55D”删除。程序如示例代码 8-18 所示。

示例代码 8-18

```
01  <?php                                                    //PHP 开始标记
02      echo substr_replace("abcdefghi", "DEF", 3);          //将输出 "abcDEF"
03      echo "<br>";                                         //换行
04      echo substr_replace("abcdefghi", "DEF", 3, 2);       //将输出 "abcDEFfghi"
05      echo "<br>";                                         //输出换行
06      $var = 'AAA5BB:/CCC55D/';                            //定义变量
07      echo "Original: $var<hr />";                       //输出变量 var 的值
08      echo substr_replace($var, 'EEEFF', 0, 0) . "<br >";//替换字符串
09      echo substr_replace($var, 'EEEFF', 8, -1) . "<br>";//替换字符串
10      echo substr_replace($var, 'EEEFF', -7, -1) . "<br>"; //替换字符串
11      echo substr_replace($var, '', 8, -1) . "<br>";      //替换字符串
12  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-18.php，查看其运行结果，即可看到运行结果如图 8-19 所示。

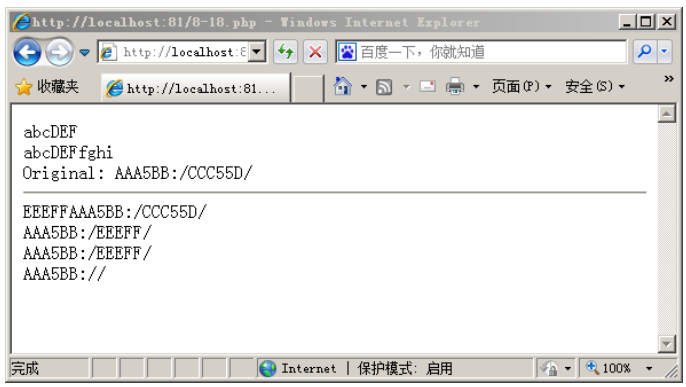

图 8-19　substr_replace()函数

【代码解析】代码第 02 行从第三个位置（即“d”）开始替换，从而把“defghi”都替换成了“DEF”。第 03 行也是从第三个位置（即“d”）开始替换，但只能替换两个长度，即到 e，所以就把“de”替换成了“DEF”。第 08 行将字符串“EEEFF”加到“AAA5BB:/CCC55D/”字符串之前，第 09 行从第 8 个字符开始，一直替换到倒数第二个字符，因此返回“AAA5BB:/EEEFF/”，第 10 行从后面数第 7 个字符开始替换，一直替换到倒数第 2 个字符，因此也返回“AAA5BB:/EEEFF/”，第 11 行删除字符串“CCC55D”。

注意：本函数可安全用于二进制对象。

8.5.3　分隔字符串

分隔字符串可以使用函数 explode()，其结构形式如下：

```
Array explode(string $separator,string $string [,int $limit])
```

该函数有三个参数：参数$separator 设置一个分隔符号，也就是按照什么原则进行分隔。如果参数为空，则函数将返回 false，不被分割。参数$string 指定所操作的字符串，$limit 指定最多将字符串分隔为多少个字符串，该参数是可选的。函数返回一个由分隔成的子串所组成的数组。

【范例 8-19】将字符串“a1 a2 a3 a4 a5 a6”分隔成 6 个字符串，并将其放在一个数组中。

以“/”为分界符，将字符串“localhost/root/123456”分隔成数组，并分别赋给变量 host、user、password。将字符串“a;b;c;d;e;f;”按“;”分隔成三个字符串，并将分成的子串放到数组 back 中。其程序如示例代码 8-19 所示。

示例代码 8-19

```
01  <?php                                                    //PHP 开始标记
02      echo "通过空格分隔字符串：<br>";                      //输出“通过空格分隔字符串：”
03      $pizza="a1 a2 a3 a4 a5 a6";                          //定义变量
04          //通过空格分隔
05      $pieces=explode(" ",$pizza);                         //分隔成数组
06          //输出返回数组的头两个元素
07      echo $pieces[0]."<br>";                              //输出 a1
08      echo $pieces[1]."<br>";                              //输出 a2
09          //将分隔后的元素保存到 list 变量中
10      echo "将分隔后的元素保存到 list 变量中：<br>";        //输出
11      $data="localhost/root/123456";                       //定义变量
12      list($host,$user,$password)=explode("/",$data);      //分隔成数组
13      echo "\$host=".$host. "<br>";                        //输出 foo
14      echo "\$user=".$user. "<br>";                        //输出变量值
15      echo "\$password=".$password. "<br>";
15      echo "限制分隔的字符串数：<br>";                     //输出变量值
16      $limit="a;b;c;d;e;f;";                               //定义变量
17      $back=explode(";",$limit,3);                         //分隔成三个数组
18      print_r($back);                                      //输出数组
19  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-19.php，查看其运行结果，即可看到运行结果如图 8-20 所示。

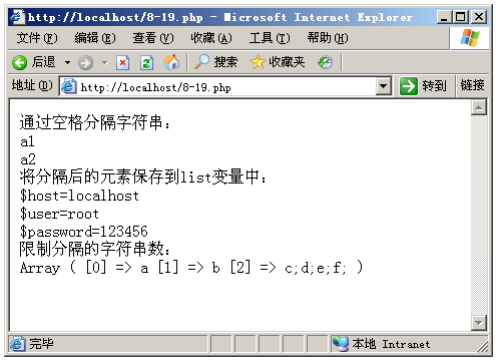

图 8-20　函数 explode()

【代码解析】代码第 05 行以“空格”为原则分隔字符串，并将返回的数组赋给变量$pieces。第 07、08 行输出数组，第 12 行将分隔后的元素保存到 list 变量中，第 17 行最多分隔成三个字符串。

8.6　字符串的其他操作

除上述介绍的字符串的处理，还有一些其他操作经常会用到，例如字符串的加密，格式化输出等。下面分别介绍这些常用的操作。

8.6.1 MD5 算法

MD5 的全称是 Message-Digest algorithm 5（信息-摘要算法），其作用是把一个任意长度的字节串换成一定长度的大字符串。其就像一个魔术箱，放进去一个物品，打开后就转化为其他的物品。在 PHP 中，只需要使用 MD5()函数即可，其格式如下：

```
string MD（string $str）
```

参数$str 即是要加密的字符串，函数一般返回一个更长的字符串，MD5 加密对于一般场合的加密已经足以应付。

【范例 8-20】利用 MD5 函数加密密码“123456”，并输出加密后的字符串，其程序如示例代码 8-20 所示。

示例代码 8-20

```
01  <?php                                          //PHP 开始标记
02      $val = '123456';                           //定义变量
03      echo "原始字符串：$val <br />";             //输出变量
04      $md_val = MD5 ($val);                      //加密字符串
05      echo "MD5 加密后的值为：$md_val";           //输出加密结果
06  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-20.php，查看其运行结果，即可看到运行结果如图 8-21 所示。

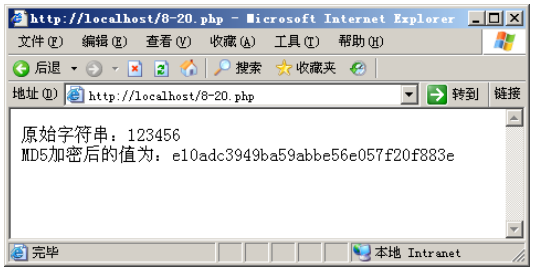

图 8-21 加密

【代码解析】代码第 04 行对字符串变量$val 加密，第 05 行输出加密后的值。

注意：MD5 加密算法是单向加密，即不存在解密算法。通常，使用 MD5 对用户密码加密，或者在采用 GET 方式传递参数时，通过已经使用 MD5 加密的字符串与现在的字符串被 MD5 后的比对实现校验。

8.6.2 创建格式化输出

PHP 包括两个在格式化输出方面非常有用的函数：printf()和 sprintf()。printf()函数为打印输出，sprintf()函数将输出作为一个字符串值返回。每个函数通常都使用两个或更多参数，第一个参数是一个名为格式串（format string）的字符串，其指定输出格式，其余参数指定要输出的值。

格式串包含一系列指令和普通字符，指令是以字符%开始的字符序列，其决定了如何格式化相应的参数。一个简单的指令可以包含%及后面的类型说明符（如 d，其指定将参数作为十进制数处理），普通字符是除%之外的任何字符。

【范例 8-21】将两个数“68.75”、“54.35”相加，利用 sprintf()的格式化输出，将结果输出为“123.100”。其程序如示例代码 8-21 所示。

示例代码 8-21

```
01  <?php                                                    //PHP 开始标记
02      $money1 = 68.75;                                     //变量赋值
03      $money2 = 54.35;                                     //变量赋值
04      $money = $money1 + $money2;                          //变量赋值
05      echo $money."<br>" ;                                 //输出变量值
06      $formatted = sprintf("%01.3f", $money);              //格式化
07      echo $formatted                                      //输出变量值
08  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-21.php，查看其运行结果，即可看到运行结果如图 8-22 所示。

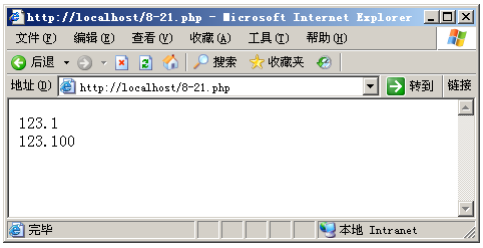

图 8-22　格式化输出

【代码解析】代码第 05 行正常状态下输出变量$money 的值，第 06 行规定输出的格式，第 07 行按规定格式输出。

8.7　综合练习

1．定义 6 组变量$s1、$s2、$s3 的值，熟悉比较、查找、转换等一些常见函数。

第一组：$s1=abcdef　$s2=a　　比较两变量。

第二组：$s1=abcdef　　返回第 3 个字符后的 2 个字符

第三组：$s1=abcdefcdef　$s2=cd　　查找$s2 在$s1 中的出现次数。

第四组：$s1=abcdef　$s2=cd　　查找$s2 在$s1 中的第一次出现位置。

第五组：$s1=php@phei.com.cn　$s2=@　　查找$s2 在$s1 中的第一次出现位置，并返回该位置以后的字符串。

第六组：$s1= Come TO　　将其转化为全大写和全小写。

提示：比较函数有 strcmp()、trim()等，查找函数有 substr_count()、subst()等。大小写转换函数有 strtoupper()和 strtolower()函数等。其程序如示例代码 8-22 所示。

示例代码 8-22

```
01  <?php                                                    //PHP 结束标记
02      $s1="abcdef";$s2="a";$s3=strcmp($s1,$s2);            //定义变量
03      echo  "\$s1=$s1  \$s2=$s2 \$s3=strcmp(\$s1,\$s2) 结果是: $s3";
                                                             //输出变量值
04
05      $s1="abcdef"; $s2=substr($s1,3,2);                   //定义变量
06      echo "<br>\$s1=$s1  \$s2=substr (\$s1,3,2)  结果是:$s2"; //输出变量值
```

```
07
08      $s1="abcdefcdef"; $s2="cd"; $s3=substr_count($s1,$s2); //定义变量
09      echo "<br>\$s1=$s1   \$s2=$s2   \$s3=substr_count(\$s1,\$s2)";
                                                                  //输出变量值
10      echo "结果是:$s3";                                        //输出变量 s3 的值
11
12      $s1="abcdef";  $s2="cd";  $s3=strpos($s1,$s2);          //定义变量
13      echo "<br> \$s1=$s1  \$s2=$s2    \$s3=strpos(\$s1,\$s2)";
                                                                  //输出变量值
14      echo "结果是:$s3";                                        //输出变量 s3 的值
15
16      $s1="php@phei.com.cn"; $s2="@";$s3=strstr($s1,$s2);   //定义变量
17      echo "<br>\$s1=$s1  \$s2=$s2  \$s3=strstr(\$s1,\$s2);";//输出变量值
18      echo "  结果是:$s3";                                      //输出变量 s3 的值
19      $s1="Come TO";                                          //定义变量
20      echo "<br>$s1 用 strtoupper 函数变换为大写:"; echo strtoupper($s1);
                                                                  //将字符转化为大写
21      echo "<br>$s1 用 strtolower 函数变换为大写:"; echo strtolower($s1);
                                                                  //将字符转化为小写
22  ?>                                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-22.php，查看其运行结果，即可看到运行结果如图 8-23 所示。

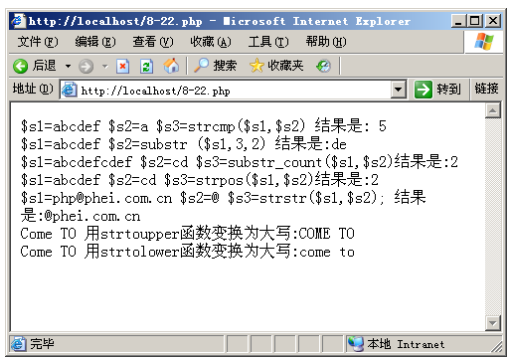

图 8-23　程序运行结果

2．将“apple、banana、orange、pear”按“、”放在数组中的两个元素中，然后去掉 pear，将剩余元素放在一个数组中。

提示：利用 explode 函数分解，并设置分解成的字符串个数，其程序如示例代码 8-23 所示。

示例代码 8-23

```
1   <?php                                                       //PHP 标记开始
2       $str = 'apple、banana、orange、pear';                   //定义变量
3       print_r(explode('、', $str, 2));                        //按|分成两个数组
4       print_r(explode('、', $str, -1));                       //分割成数组
5   ?>                                                          //PHP 标记结尾
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/8-23.php，查看其运行结果，即可看到运行结果如图 8-24 所示。

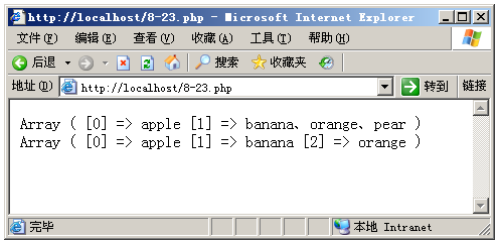
http://localhost/8-23.php - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://localhost/8-23.php
转到 链接
Array ([0] => apple [1] => banana、orange、pear)
Array ([0] => apple [1] => banana [2] => orange)
完毕
本地 Intranet

图 8-24　程序运行结果

8.8 小结

本章主要介绍了字符串的去除特殊字符、字符串的大小写转换、字符串的比较、字符串的查找与替换，以及字符串的加密等一些常用操作。这些操作在编写程序中经常遇到，掌握常用的处理方法，有时能使一些烦琐的问题变得简单。

如果读者对本章内容还有疑问，可参考《精通 PHP5 应用开发》（秦涛，曾文玉．北京：人民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民．北京：中国水利水电出版社，2003）。

8.9 习题

一、填空题

1．函数_____用于去除字符串两端指定的任意特殊字符。函数_____用于去除字符串左端指定的任意特殊字符。函数_____用于去除字符串右端指定的任意特殊字符。

2．函数_____将传入的字符串全部转换为小写，函数_____将传入的字符串全部转换为大写。

3．函数_____的作用是将字符串的第一个字符转化为大写，函数_____的作用是将字符串中的每个单词的首字符转化为大写。

4．函数 strcmp()用于_____的字符串的比较，函数 strcasecmp()用于_____的字符串的比较。

5．代码 strncasecmp("ABCde", "abcdg", 3)的返回结果为_____。

6．函数_____返回查找字符串最后一次出现的位置，函数_____返回查找字符串第一次出现的位置。

7．代码 echo strstr("My name is li lei", i)和 echo strrchr ("My name is li lei", i)的返回结果分别为_____、_____。

8．用于加密字符串的函数为_____。

9．代码 echo $formatted = sprintf("%01.3f", 10)的输出结果为_____。

二、选择题

1．查找子字符串在母字符串中第一次出现的位置，并返回从此位置开始到到母字符串结束的部分的函数是（　　）。

A．strstr()　　B．strrchr()　　C．strcmp()　　D．str_replace()

2．用来查找字符串出现次数的函数为（　　）。

A．strrpos()　　B．strpos()　　C．substr_count()　　D．strncmp()

3．选择下面程序的运行结果（　　）。

```
echo $pos = strrpos(php, "p")."<br>";
echo $po = strpos(php, "p")
```

A．3　　B．2　　C．1　　D．3

1　　0　　1　　3

4．选择下面程序的运行结果（　　）。

```
echo substr_replace("ABC", "DEF", 3, 1)
```

A．ABC　　B．DEF　　C．ABCDEF　　D．以上都不对

5．选择下面程序的运行结果（　　）。

```
echo substr_count("LiLi is a good girl", 'i', 2)
```

A．4　　B．3　　C．2　　D．1

三、简答题

1．简述 trim()函数默认情况下去除的特殊字符。

2．试列出字符串的大小转换的函数。

3．列出关于查找匹配字符串的函数，并说明其功能。

四、编程题

1．创建如下字符串：

```
$str="Welcome to BeiJing."
```

然后进行如下操作：统计字符数、全部转化为大写，将 beijing 改为 China。

2．将人名 LiLei|LiLi |Tom |Jon 放进数组中，并输出结果。

第 9 章　HTML 表单

表单是 Web 页面与用户交互的基本方式，是实现动态网页的一种主要外在形式。通过表单可以进行个人信息注册、网络调查等各种活动。其就像一份试卷，有选择、填空等不同形式的题目，通过填写试题，完成各项活动。表单是用户与页面交互信息的重要手段，下面开始具体介绍表单。

学习本章，可以获得以下知识点：

- 表单输入标记</INPUT>；
- 表单的提交；
- PHP 处理表单的方法；
- 验证表单的方法。

9.1　表单制作

表单是网页上的一个特定区域，这个区域是由一对<form>标记定义的。如图 9-1 所示即是用户的 HTML 表单界面，其程序结构通过查看源文件如图 9-2 所示，其是以<form>标记的表单代码。

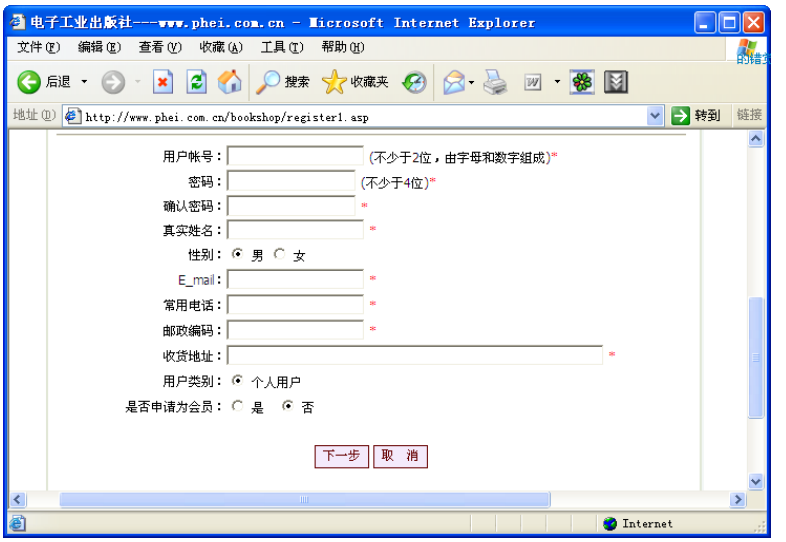

图 9-1　HTML 表单

表单的基本结构形式为：

```
<form  method="method" action="ulr" name="name">
……
</form>
```

属性 name 为表单名称，method 为表单数据的传送方式，一般有 get、post 两种，action 为数据处理的网页程序文件名。

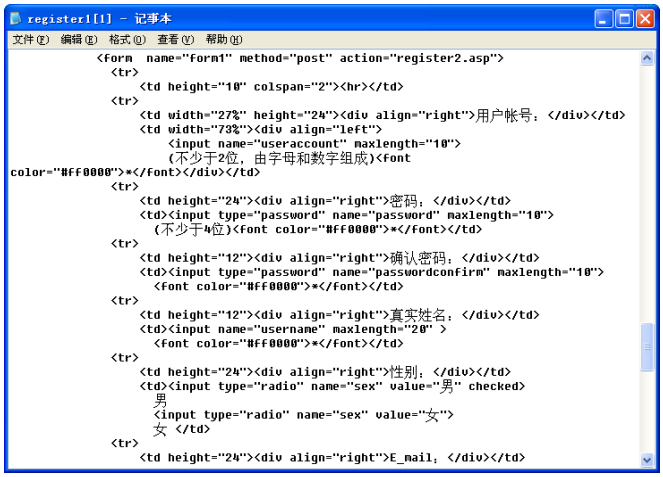

图 9-2　表单源文件

表单以<form>标记开始，以</form>标记结束，在这一对标记之间放入的是表单的对象。单击提交按钮，提交的范围就在这对标记之间的部分，其中包含处理表单的脚本程序位置、提交表单的方法等信息。其就像试卷上标明做完后交到什么地方，怎么去交等信息。在<form>标记中，可以包含如表 9-1 所示的 4 个标记。

表 9-1　表单包含标记

标　记	描　述
<input>	输入标记
<select>	菜单和列表标记
<option>	菜单和列表项目标记
<textarea>	文字域标记

利用表单包含的标记创建一个含有表 9-1 所列的标记的表单，例如：

```
01  <form>                                  <!--表单开始标记-->
02      <input>….. </input>                  <!--输入标记-->
03      <textarea>……</textarea>              <!--文字域标记-->
04      <select>…….                         <!--菜单和列表开始标记-->
05          <option>…….</option>             <!--菜单和列表项目标记-->
06      </select>                           <!--/菜单和列表结束标记-->
07  </form>                                 <!--表单结束标记-->
```

代码是一个完整的表单，以<form>开始，以</form>结束。第 02 行输入标记，第 03 行为文字域标记，第 04～06 行为选择列表标记。

提示：本节程序都保存在 root 目录下的 chengxu 文件夹下。

9.2　输入标记<input>

表单一般要填写信息，填写信息有其特定的输入消息标记。<input>即是一个比较常见的输入标记，常用的有文本域、密码、按钮等都使用这个标记。

9.2.1 文本域 text

文本域 text，其基本结构形式为：

```
<input type=text name = field_name  maxlength = value  size=value
value=field_value>
```

type 的属性值 text 设定为表单的文本域，可输入任何类型的文本、数字和字母，输入的内容单行显示。name 是文本域的名称，maxlengh 是文本域的最大输入字符数，size 是文本域的宽度，value 是文本域的默认值。

【范例 9-1】编写个人简历表单，要求填写“姓名，学历，毕业院校，专业特长，个人爱好和求职意向”等内容。其程序如示例代码 9-1 所示。

示例代码 9-1

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>                                           <!--文件标题标记开始-->
            个人简历                                      <!--文件标题-->
        </title>                                          <!--文件标题标记结束-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        <H1>用个人简历</H1>                               <!--标题字-->
        <form method=get action=mailto:husong@163.com name=invest>
                                                          <!--表单开始标记-->
        姓名：<input  type=text size=20 name=username ><br>
                                                          <!-- input 输入框-->
        学历：<input  type=text size=20 name=grad ><br><!--input 输入框-->
        毕业院校: <input  type=text size=20 name=school ><br>
                                                          <!--input 输入框-->
        专业特长：<input  type=text size=20 name=special ><br>
                                                          <!--input 输入框-->
        个人爱好：<input  type=text size=20 name=taste ><br>
                                                          <!--input 输入框-->
        求职意向：<input  type=text size=20 name=job  ><br>
                                                          <!--input 输入框-->
    </form>                                               <!--表单结尾标记-->
    </body>                                               <!--body 标签结束-->
</html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-1.html，查看其运行结果，即可看到示例代码的运行结果如图 9-3 所示。

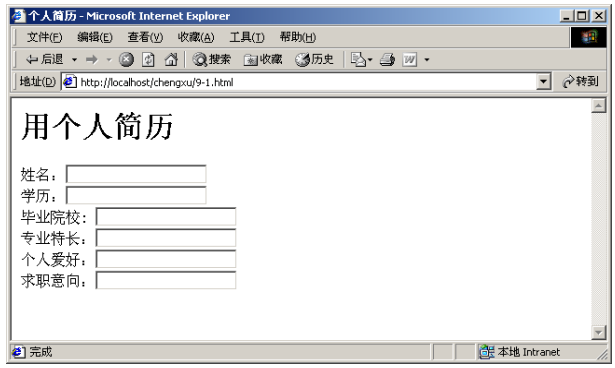

图 9-3　文本域

【代码解析】代码第 09～16 行定义了一个表单。第 10～15 行使用<input>标记，属性值为 text 的文本域，分别输入姓名、学历等。

提示：对<input>标签来说，只有 type 和 name 属性是必需的，其他的可以自由设定。

9.2.2 密码域 password

在表单中，常见到输入到文本域中的文字均以“*”符号显示，这种形式的文本域称为密码域，其基本结构为：

```
<input type=password name=field_name >
```

name 为密码域的名字，这些属性的含义同文本域相同，只是在输入显示形式上有所不同，一般是为了不易被外界发现，为保密而使用。

【范例 9-2】以身份验证为例，填写“姓名，学号，邮箱”等内容，并设置密码，其程序如示例代码 9-2 所示。

示例代码 9-2

```
01  <html>                                                    <!--html 标签开始-->
02      <head>                                                <!--头标记开始-->
03          <title>                                           <!--文件标题标记开始-->
04              身份验证                                      <!--文件标题-->
05          </title>                                          <!--文件标题标记结束-->
06      </head>                                               <!--头文件结束-->
07      <body>                                                <!--body 标签开始-->
08          <H1>身份验证</H1>                                 <!--标题字 -->
09          <form method=get action=mailto:husong@163.com name=invest>
                                                              <!--表单开始标记-->
10          姓名: <input  type=text size=20 name=username ><br>
                                                              <!--input 输入框-->
11          学号: <input  type=text size=20 name= numb  ><br>
                                                              <!--input 输入框-->
12          邮箱: <input  type=text size=20 name=email  ><br>
                                                              <!--input 输入框-->
13          密码: <input  type=password size=21 name=passwd ><br>
                                                              <!--密码域-->
14          密码确认: <input  type=password size=21 name=rpasswd ><br>
                                                              <!--密码域-->
15      </form>                                               <!--表单结尾标记-->
16      </body>                                               <!--body 标签结束-->
17  </html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-2.html，查看其运行结果，然后在密码文本框中输入“123456”，在确认密码文本框中再次输入，即可看到运行结果如图 9-4 所示。

【代码解析】代码第 09～15 行定义一个表单，第 10～12 行为文字域，分别输入姓名、学号等内容，第 13、14 行为密码域，输入密码，然后确认密码。

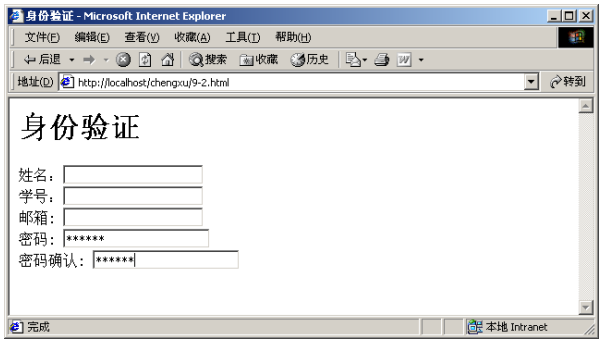

图 9-4　密码域

9.2.3　文件域 file

文件域可以用来浏览查找本地文件路径，其基本结构为：

```
<input type=file name=feild_name>
```

文本域的外观是一个文本框加一个浏览按钮，用户可以直接将上传文件路径填写在文本框中，也可以单击浏览按钮，找到需要上传的文件。

【范例 9-3】建立一个用户调查表，输入“姓名，网站，密码”，并上传个人照片，其程序如示例代码 9-3 所示。

示例代码 9-3

```
01  <html>                                                    <!--html 标签开始-->
02      <head>                                                <!--头标记开始-->
03          <title>                                           <!--文件标题标记开始-->
04              插入文件域                                    <!--文件标题-->
05          </title>                                          <!--文件标题标记结束-->
06      </head>                                               <!--头文件结束-->
07      <body>                                                <!--body 标签开始-->
08          <H1>用户调查表</H1>                               <!--标题字 -->
09          <form method=get action=mailto:husong@163.com name=invest>
                                                              <!--表单开始标记-->
10          姓名：<input  type=text size=20 name=username ><br>
                                                              <!- input 输入框-->
11          网站：<input type=text size=20 name=passwd size=20  value="http://"><br>
12          密码：<input type="password" name="pw" size=20 maelenth=8><br>
13          确认密码：<input type="password" name="rpw" size=20 maelenth =8><br>
14          个人照片：<input type="file" name="FILE">          <!--file 文件域-->
15          </form>                                           <!--表单结尾标记-->
16      </body>                                               <!--body 标签结束-->
17  </html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-3.html，查看其运行结果，即可看到运行结果如图 9-5 所示。

【代码解析】代码第 09～15 行定义一表单，第 10～13 行为文本域，要求输入姓名、网站等信息。第 14 行将 type 属性值设置为 file，可以浏览本地电脑内的文件，然后上传。

提示：如果不指定 input 框的大小，浏览器会生成一个 20 个字符宽的输入框。

图 9-5　文件域

9.2.4　提交与重置按钮

表单一般通过提交来处理信息，要提交信息，就要有提交按钮，HTML 提供提交和重置按钮来实现信息的提交和重置，其基本结构如下：

```
<input type=submit value=发送>
<input type=reset value=重置>
```

submit 是提交按钮，将信息送到指定地点。reset 是重置按钮，重新填写表单信息。value 是显示在按钮上的文字。

【范例 9-4】建立身份验证表单，填写“昵称，年龄，邮箱，密码”等信息，设置“提交”和“重置”按钮以处理信息。其程序如示例代码 9-4 所示。

示例代码 9-4

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>                                           <!--文件标题标记开始-->
            身份验证                                      <!--文件标题-->
        </title>                                          <!--文件标题标记结束-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        <H1>身份验证</H1>                                 <!--标题字 -->
        <form method=get action=mailto:husong@163.com name=invest>
                                                          <!--表单开始标记-->
        昵称: <input  type=text size=20 name=username ><br>
                                                          <!--表单开始标记-->
        年龄: <input  type=text size=20 name=passwd  ><br>
                                                          <!--表单开始标记-->
        邮箱: <input  type=text size=20 name=passwd  ><br>
                                                          <!--表单开始标记-->
        密码: <input  type=password size=20 name=passwd ><br>
                                                          <!--密码域-->
        密码确认: <input  type=password size=20 name=passwd ><br>
                                                          <!--密码域-->
    <p><input type=submit value=发送>                     <!--提交按钮-->
    <input type=reset value=重置>                         <!--重置按钮-->
    </form>                                               <!--表单结尾标记-->
    </body>                                               <!--body 标签结束-->
</html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-4.html，查看其运行结果，即可看到结果如图 9-6 所示。

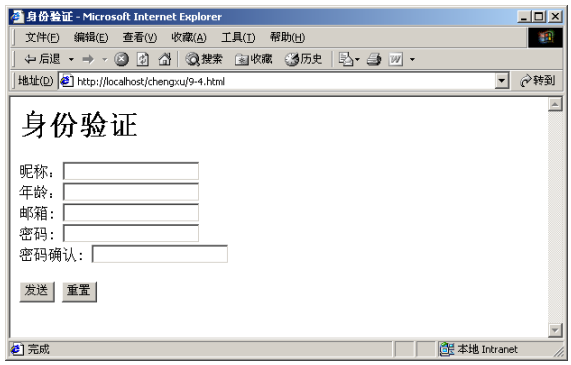

图 9-6　提交重置按钮

【代码解析】代码第 09 行设置开始表单标记。第 10～14 行设置文本域，输入昵称、年龄等信息。第 15 行将属性设置为 submit，设置提交按钮。第 16 行将属性值设为 reset，设置重置按钮。

提示：另外还有普通按钮 button，其按钮不能用来提交和重置表单。

9.2.5　复选框

兴趣爱好往往不止一种，如果要对用户的爱好进行调查，需要用户有多个选择，这种情况称为复选。而复选框 checkbox 可以同时选中多个选项，其结构形式如下。

```
<INPUT Type="checkbox" name="field_name" value="value">
```

checkbox 为复选框，当属性设为 checkbox 时，可以同时选择多个项目。

【范例 9-5】对用户的爱好进行调查，并提交信息。调查内容包括“时尚，旅游，电影，音乐，体育，艺术，文学，上网”等，其程序如示例代码 9-5 所示。

示例代码 9-5

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>                                           <!--文件标题标记开始-->
            用户调查                                      <!--文件标题-->
        </title>                                          <!--文件标题标记结束-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        <H1>兴趣爱好</H1>                                 <!--标题字 -->
        <form action=url method=post>                     <!--表单开始标记-->
        <p><input name=mode  type=checkbox >时尚          <!--复选框-->
        <p><input name=tour  type=checkbox>旅游           <!--复选框-->
        <p><input name=movie  type=checkbox>电影          <!--复选框-->
        <p><input name=music  type=checkbox >音乐         <!--复选框-->
        <p><input name=spot  type=checkbox >体育          <!--复选框-->
        <p><input name=art  type=checkbox >艺术           <!--复选框-->
        <p><input name=literature  type=checkbox >文学    <!--复选框-->
        <p><input name=web  type=checkbox >上网           <!--复选框-->
```

```
18          <p><input name=others  type=checkbox >其他          <!--复选框-->
19          <p><input type=submit value=发送>                   <!--提交按钮-->
20          <input type=reset value=重置>                       <!--重置按钮-->
21      </form>                                                 <!--表单结尾标记-->
22      </body>                                                 <!--body 标签结束-->
23  </html>                                                     <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-5.html，查看其运行结果，即可看到结果如图 9-7 所示。

图 9-7　复选框

【代码解析】代码第 09～21 行定义表单。第 10～18 行定义复选框的选项信息。第 19 行设置提交按钮，用来处理信息。第 20 行设置重置按钮，用来重置填写信息。

9.2.6　单选按钮

如果在许多选项中只允许选择一个项目，此种情况称为单选。单选按钮 radio 只允许用户有一个选择，其基本结构形式为：

```
<input type="radio" name="field_name" value="value">
```

radio 为单选按钮的属性值，当属性值设为 radio 时，一次只能选择一个项目。

【范例 9-6】建立用户调查表单，对用户喜欢电影的类型做调查，并提交信息。每人只能有一个选择，所要选择的内容为“情感片，历史剧，科幻片，动画片，喜剧片，恐怖片，灾难片，动作片，记录片”等。其程序如示例代码 9-6 所示。

示例代码 9-6

```
01  <html>                                          <!--html 标签开始-->
02      <head>                                      <!--头标记开始-->
03          <title>                                 <!--文件标题标记开始-->
04              用户调查                            <!--文件标题-->
05          </title>                                <!--文件标题标记结束-->
06      </head>                                     <!--头文件结束-->
07      <body>                                      <!--body 标签开始-->
08          <H1>喜欢的电影</H1>                         <!--标题字 -->
```

```
09          <form action=url method=post>                <!--表单开始标记-->
10          <p><input name=movie type=radio>情感片       <!--单选按钮选项-->
11          <p><input name=movie type=radio>历史剧       <!--单选按钮选项-->
12          <p><input name=movie type=radio>科幻片       <!--单选按钮选项-->
13          <p><input name=movie type=radio>动画片       <!--单选按钮选项-->
14          <p><input name=movie type=radio>喜剧片       <!--单选按钮选项-->
15          <p><input name=movie type=radio>恐怖片       <!--单选按钮选项-->
16          <p><input name=movie type=radio>灾难片       <!--单选按钮选项-->
17          <p><input name=movie type=radio>动作片       <!--单选按钮选项-->
18          <p><input name=movie type=radio>记录片       <!--单选按钮选项-->
19          <p><input name=movie type=radio>其他         <!--单选按钮选项-->
20          <p><input type=submit value=发送>            <!--提交按钮-->
21          <input type=reset value=重置>                <!--重置按钮-->
22          </form>                                      <!--表单结尾标记-->
23      </body>                                          <!--body 标签结束-->
24  </html>                                              <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-6.html，查看其运行结果，即可看到结果如图 9-8 所示。

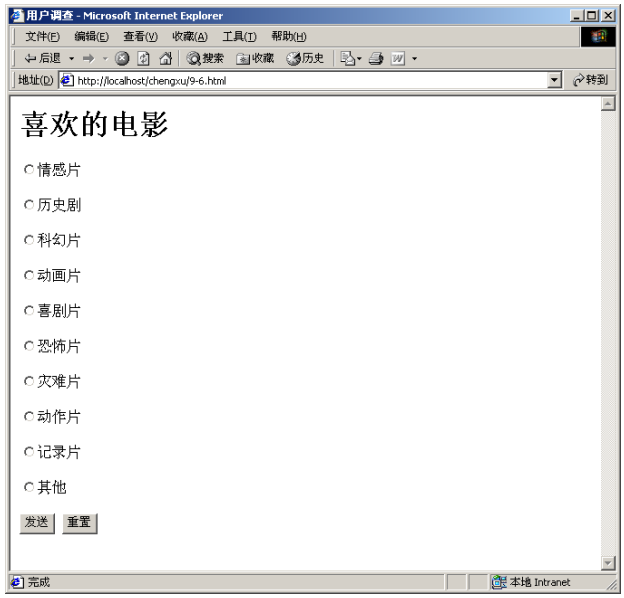

图 9-8　单选按钮

【代码解析】代码第 09～22 定义表单，第 10～18 行为定义的单选按钮的选项信息。第 20 行设置提交按钮，用来提交信息。第 21 行设置重置按钮，用来重新填写信息。

注意：不同的复选框应设置不同的名字，单选按钮可以设置同一个名字。

9.2.7　<select>下拉列表标记

如果可供选择的项目过多时，会在浏览器页面上占很大的空间，这样看起来会很不美观。HTML 还提供了另一种选择菜单——下拉列表，可以节省空间，正常状态下只能看到一个选项，单击下三角按钮打开列表就能看到全部选项。

列表可以显示一定数量的选项，如果超出了这个数量，会自动出现滚动条，浏览者可以通过拖动滚动条来查看各选项，其基本语法结构为：

```
<select name="name" size=value >
    <option value="value" selected>选项
<option value="value">选项
     ……
    </select>
```

name 为下拉列表的名字，size 设置下拉列表的高度、一次显示的个数，默认为 1。selected 表示当前被默认选中的项，value 表示该项对应的值，该项被选中后，其值将被发送到服务器上。

【范例 9-7】建立下拉列表，选择喜欢居住的城市做调查，可以选择的城市有“北京，上海，天津，重庆，西安，成都，南宁，郑州，武汉，长沙，广州，深圳，大连，青岛或其他城市”，其程序如示例代码 9-7 所示。

示例代码 9-7

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>                                           <!--文件标题标记开始-->
            用户调查                                      <!--文件标题-->
        </title>                                          <!--文件标题标记结束-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        <center><H1>用户调查表</H1></center>              <!--标题字-->
        <form action=url method=post name=questionary><!--表单开始标记-->
        请选择你所居住的城市：<br>                        <!--输出文字-->
        <select name= province >                          <!--下拉列表开始-->
        <option Value=beijing selected>北京               <!--下拉选项-->
        <option Value=shanghai >上海                      <!--下拉选项-->
        <option Value=tianjin > 天津                      <!--下拉选项-->
        <option Value=chongqing >重庆                     <!--下拉选项-->
        <option Value=xian >西安                          <!--下拉选项-->
        <option Value=chengdu >成都                       <!--下拉选项-->
        <option Value=nanning >南宁                       <!--下拉选项-->
        <option Value=zhengzhou >郑州                     <!--下拉选项-->
        <option Value=wuhan >武汉                         <!--下拉选项-->
        <option Value=changsha >长沙                      <!--下拉选项-->
        <option Value=guangzhou >广州                     <!--下拉选项-->
        <option Value=guangzhen >深圳                     <!--下拉选项-->
        <option Value=dalian >大连                        <!--下拉选项-->
        <option Value=qingdao >青岛                       <!--下拉选项-->
        <option Value=others >其他                        <!--下拉选项-->
        </select>                                         <!--下拉列表结束-->
        <p><input type=submit value=发送>                 <!--提交按钮-->
        <input type=reset value=重置>                     <!--重置按钮-->
        </form>                                           <!--表单结尾标记-->
    <hr>                                                  <!--横线标签-->
        <form action=url method=post name=invest>         <!--表单开始标记-->
        <p>请选择你喜欢的运动:<p>                         <!--输出说明文字-->
        <select name=spot size=4 >                        <!--下拉列表开始-->
        <option Value=swimming  selected>游泳             <!--下拉选项-->
        <option Value=tennisball >网球                    <!--下拉选项-->
        <option Value=badminton> 羽毛球                       <!--下拉选项-->
```

```
38          <option Value=pingpong >乒乓球                    <!--下拉选项-->
39          <option Value=basketball >篮球                    <!--下拉选项-->
40          <option Value=volleyball >排球                    <!--下拉选项-->
41          <option Value=football >足球                      <!--下拉选项-->
42          <option Value=mountaineering >登山                <!--下拉选项-->
43          <option Value=skate >溜冰                         <!--下拉选项-->
44          <option Value=skee >滑雪                          <!--下拉选项-->
45          <option Value=golf>高尔夫                         <!--下拉选项-->
46          <option Value=others > 其他                       <!--下拉选项-->
47          </select>                                         <!--下拉列表结束-->
48          <p><input type=submit value=发送>                 <!--提交按钮-->
49          <input type=reset value=重置>                     <!--重置按钮-->
50          </form>                                           <!--表单结尾标记-->
51      </body>                                               <!--body 标签结束-->
52  </html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-7.html，查看其运行结果，即可看到结果如图 9-9 所示。

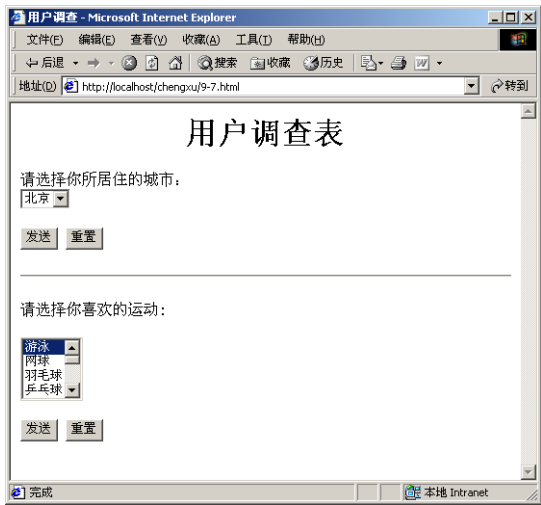

图 9-9　下拉列表

【代码解析】代码第 11～25 行定义一个只限制菜单名的下拉列表，其高度默认为 1。第 32～45 行定义另一个设置 size 高度为 4 的下拉列表。

提示：如果 select 标签没有默认被选中的项，其就预先选中第一个选项。

9.3　提交表单信息

填写完表单信息后，接着处理表单信息，提交表单信息的方式主要有 get 和 post 两种方法，下面分别讲这两种提交的方式。

9.3.1　get 方法提交

get 提交方法的本质上是将数据通过连接地址的形式传递到下一个页面，此方法提交表单，

表单不会明显地改变页面状态。get 方法是最简单的提交方法，主要用于静态 HTML 文档、图像或数据库查询结果的简单检索。

【范例 9-8】创建用户登录表单，输入用户名和密码，使用 get 方式提交表单。提交后显示“通过 GET 方法提交的信息!”，用户名和密码都为“php”，其程序如示例代码 9-8 所示。

示例代码 9-8

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>                                           <!--文件标题标记开始-->
            用户登录                                      <!--文件标题-->
        </title>                                          <!--文件标题标记结束-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        请输入账号和密码：<br>                            <!--表单说明文字-->
        <form method=get action="get.html">               <!--表单开始标记-->
        用户名：<INPUT maxLength=25 size=18 name=login ><br>
                                                          <!--输入用户名-->
           密码：<INPUT type=password size=19 name=passwd ><br>
                                                          <!--输入密码-->
          <input type="submit" name="submit" value="提交"> 
                                                          <!--提交按钮-->
        <input type="reset" name="reset" value="重填"></form>
                                                          <!--重置按钮-->
</body>                                                   <!--body 标签结束-->
</html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-8.html，查看其运行结果，即可看到示例代码的运行结果如图 9-10 所示。

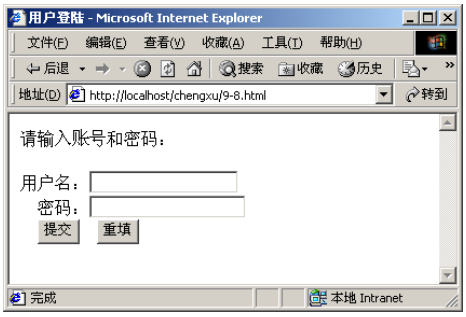

图 9-10　get 提交

处理表单的信息 get.html 文件代码如下：

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>                                           <!--文件标题标记开始-->
            get 提交                                      <!--文件标题-->
        </title>                                          <!--文件标题标记结束-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        通过 GET 方法提交的信息!                          <!--提交方式说明-->
    </body>                                               <!--body 标签结束-->
</html>                                                   <!--html 标签结束-->
```

【运行结果】用户名和密码输入“php”后单击“提交”按钮，即可看到如图 9-11 所示的内容。

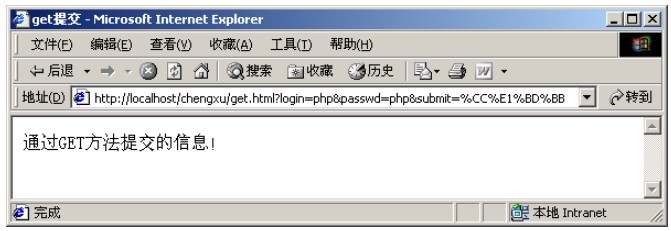

图 9-11　get 处理

【代码解析】代码第 09 行定义确定以 get 方式提交表单，处理界面为“get.html”。单击“提交”按钮后处理 get.html 文件内容，则输出“通过 GET 方法提交的信息!”。

注意：通过 get 方法提交的信息，其处理信息的内容将显示到 IE 地址栏中，如图 9-11 所示。填写用户名和密码信息都在地址栏中显示，因此 get 提交方式的安全性很差。

9.3.2　post 方法提交

post 方法比 get 方法有很多优势，例如，如果数据很多，用 get 提交会丢失很多数据；如果用 post 提交，可以尽可能多地传递数据。而且 post 也有更高的安全性，不会像 get 那样把传送的数据暴露在浏览器的地址栏中。

【范例 9-9】以 post 方式提交范例 9-8 所示的内容，并比较其异同，其程序如示例代码 9-9 所示。

示例代码 9-9

```
01  <html>                                                    <!--html 标签开始-->
02      <head>                                                <!--头标记开始-->
03          <title>                                           <!--文件标题标记开始-->
04              用户调查表                                    <!--文件标题-->
05          </title>                                          <!--文件标题标记结束-->
06      </head>                                               <!--头文件结束-->
07      <body>                                                <!--body 标签开始-->
08          请输入昵称和密码：<br>                            <!--说明文字-->
09          <form method=post action="post.html">             <!--表单开始标记-->
10          昵称：<input maxLength=25 size=18 name=login ><br>
                                                              <!--input 输入框-->
11          密码：<input type=password size=19 name=passwd ><br>
                                                              <!--密码域-->
12          <input type="submit" name="submit" value="提交">
                                                              <!--提交按钮-->
13          <input type="reset" name="reset" value="重填">
                                                              <!--重置按钮-->
14      </form>                                               <!--表单结尾标记-->
15  </body>                                                   <!--body 标签结束-->
16  </html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-9.html，查看其运行

结果，即可看到结果如图 9-12 所示。

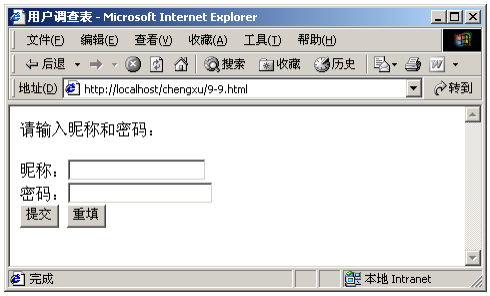

图 9-12　post 提交

处理表单信息的 post.html 文件代码如下：

```
01  <html>                                    <!--html 标签开始-->
02      <head>                                <!--头标记开始-->
03          <title>                           <!--文件标题标记开始-->
04              post 提交                     <!--文件标题-->
05          </title>                          <!--文件标题标记结束-->
06      </head>                               <!--头文件结束-->
07      <body>                                <!--body 标签开始-->
08          通过 post 方法提交信息!           <!--提交方式说明-->
09      </body>                               <!--body 标签结束-->
10  </html>                                   <!--html 标签结束-->
```

【运行结果】用户名和密码输入“php”后单击“提交”按钮，即可看到如图 9-13 所示的内容。

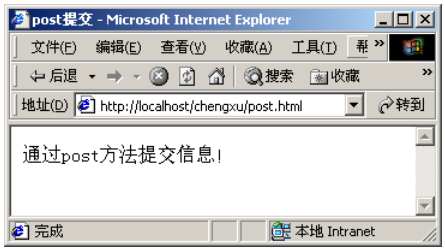

图 9-13　post 处理

【代码解析】代码第 09 行定义确定以 post 方式提交表单，处理界面为“post.html”。单击“提交”按钮后处理 post.html 文件内容，则输出“通过 post 方法提交的信息!”。

注意：PHP 为用户提供了众多方便易用的强大功能。在表单的处理方面，PHP 能够自动地将由客户端表单发送的数据赋值给相应变量，从而极大地简化了整个表单的处理过程。

9.4　PHP 处理表单

PHP 最有用的特性之一是其能够自动将表单中的变量赋予 PHP 变量，这使得表单的处理变得非常快捷。下面通过实例来讲解 PHP 处理表单，获取列表单提交的数据的方式。

【范例 9-10】使用 PHP 处理一周内的活动安排，其程序如示例代码 9-10 所示。

其活动内容为：“星期一：看比赛，星期二：去上班，星期三：旅游，星期四：运动，星期五：读书，星期六：交友，星期日：看电影”。

示例代码 9-10

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>                                           <!--文件标题标记开始-->
            一周活动                                      <!--文件标题-->
        </title>                                          <!--文件标题标记结束-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        本周安排：<br>                                    <!- 说明文字-->
        <form method=post action="tijiao.php">            <!--表单开始标记-->
            <select name="bmonth">                        <!--下拉列表开始-->
            <option value="看比赛" selected>星期一</option>
                                                          <!--下拉选项-->
            <option value="去上班">星期二</option>        <!--下拉选项-->
            <option value="旅游">星期三</option>          <!--下拉选项-->
            <option value="运动">星期四</option>          <!--下拉选项-->
            <option value="读书">星期五</option>          <!--下拉选项-->
            <option value="交友">星期六</option>          <!--下拉选项-->
            <option value="看电影">星期日</option>        <!--下拉选项-->
             </select>                                    <!--下拉列表结束-->
            <input type="submit" name="submit" value="提交">
                                                          <!--提交按钮-->
            <input type="reset" name="reset" value="重填">
                                                          <!--重置按钮-->
        <form><                                           <!--表单结尾标记-->
    </body>                                               <!--body 标签结束-->
</html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-10.html，查看其运行结果，即可看到结果如图 9-14 所示。

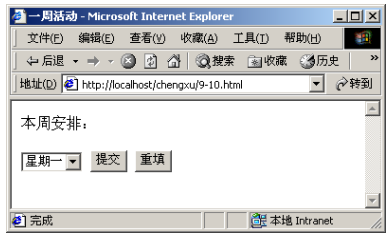

图 9-14　PHP 处理表单

处理表单信息的 tijiao.php 文件代码如下：

```
<?php                                                     //PHP 开始标记
    echo "今天的活动是：";                                //输出“今天的活动是:”
    echo $_POST['bmonth']  .  "<br>对吗？";               //输出提交信息
?>                                                        <!--PHP 结束标记-->
<p>                                                       <!--换行-->
<a href="9-10.php">返回 </a>                              <!--返回首页-->
```

【运行结果】选择“星期五”，然后提交，即可看到如图 9-15 所示内容。

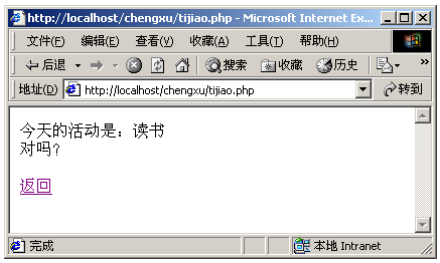

图 9-15　提交结果

【代码解析】示例代码 9-10 的第 09 行设置了提交的方式和处理提交信息的页面，处理页面的文件为 tijiao.php。第 10 行设置了下拉列表的名字，处理提交信息的为 tijiao.php 文件。tijiao.php 文件的第 03 行通过$_POST 函数取得提交的信息变量的值并输出变量值。

提示：PHP 处理 HTML 表单可以直接在 HTML 表单文件中处理，也可以在另一个单独的 PHP 文件中处理，这取决于 HTML 表单<form>标记中的“action”的属性值。

9.5　表单验证

在用户提交表单数据的时候，有时要对提交数据的合法性进行判断。如用户名长度、密码、电子邮件地址、电话号码、身份号码等。下面简单介绍一些常用的判断。

9.5.1　用户名验证

用户名几乎是每个表单所必填的，而且用户名常常有一些特殊的限制，如长度、特殊符号等限制。因此对用户的验证显得非常必要，下面通过实例进行简单的用户的验证。

【范例 9-11】给变量$name 赋予 php，判断其用户名是否为空，其长度是否在 3～10 之间。如果为空，则输出“姓名不能为空”。如果长度不合法，输出“姓名长度必须在 3 和 10 之间”，姓名必须是字母或数字的组合。如果合法，输出用户名。其程序如示例代码 9-11 所示。

示例代码 9-11

```
01  <?php                                                    //PHP 开始标记
02      $name="php";                                         //变量赋值
03      if(empty($name))                                     //判断是否为空
04      {
05          echo "姓名不能为空";                             //输出“姓名不能为空”
06      }
07          //判断长度
08      elseif((strlen($name)<3) || (strlen($name)>10)) //判断长度是否在3和10之间
09      {
10          echo "姓名长度必须在 3 和 10 之间";             //输出原因
11      }
12      elseif(!ereg("[0-9a-zA-Z]+",$name))
13          echo ""姓名格式不正确;
14      else                                                 //姓名合法情况
15      {
16          echo "用户名为：$name";                          //输出用户名
17      }
18  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-11.php，查看其运行结果，即可看到结果如图 9-16 所示。

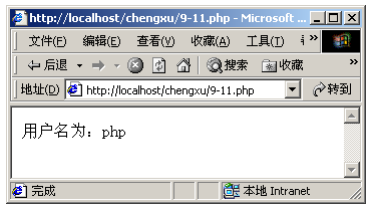

图 9-16　用户名验证

【代码解析】代码第 03 行判断名字是否为空，第 08 行判断用户名长度是否符合要求。

注意：以上的验证方式也适合一般的文本验证，如地址等信息。

9.5.2　邮件地址验证

相比上面的姓名验证，电子邮件验证稍微复杂一些，电子邮件最主要的特征是其中必须包含“@”符号。一般来说，“@”符号后面是一个邮件服务器地址，如 sina.com，其后面有含有符号“.”，不过其判断方式与“@”相同。

【范例 9-12】判断电子邮件地址“php@163.com”是否合法，判断是否已经填写，是否含有符号“@”，如果正确，则输出邮件地址。其程序如示例代码 9-12 所示。

示例代码 9-12

```
01  <?php                                                    //PHP 开始标记
02      $email="php@163.com";                                //定义变量 email
03      if(empty($email))                                    //判断变量是否为空
04      {
05          echo "邮件地址不能为空";                          //输出“邮件地址不能为空“
06      }
07      elseif(!ereg    ("([0-9a-zA-Z]+)([@])([0-9a-zA-Z]+)(.)([0-9a-zA-Z]+)",
$email)                                                      //判断电子邮件是否合法
08      {
09          echo "电子邮件不合法，请重新输！"                 //输出解释
10      }
11      else                                                 //输出电子邮件
12      {
13          echo "邮件地址为：$email";                        //输出邮件地址
14      }
15  ?>
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-12.php，查看其运行结果，即可看到结果如图 9-17 所示。

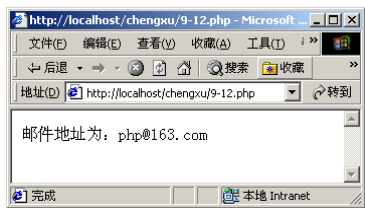

图 9-17　电子邮件验证

【代码解析】代码第 03 行判断电子邮件地址是否为空，第 07 行判断电子邮件是否含有“@”符号，第 13 行输出邮件地址。

9.5.3 密码验证

密码验证的特点就是要限制密码的长度，然后需要确认两次输入的密码是否一致。为了保证两次密码完全相同，需要验证两次密码的值是否相等。例如要判断密码长度是否在 6～20 之间和两次输入的密码是否一致，可通过下面的程序判断。

```
01  if(empty($password)>5 || empty ($cpassword)>21)      //判断长度是否在 6~20 之间
02  {
03      echo "密码长度在 6 和 20 之间";                    //输出“密码长度在6~20之间“
04  }
05  elseif(!(strlen($password) == strlen ($cpassword))) //比较长度是否相等
06  {
07      echo "密码输入有误！";                             //输出“两次输入密码不匹配！“
08  }
09  elseif (!($password === $cpassword))                  //判断两个密码是否一样
10  {
11      echo "密码不匹配！";                               //输出“两次密码不匹配!”
12  }
13  else                                                  //完全匹配的情况
14  echo "密码输出正确";                                  //输出“*”
```

代码第 01 行判断密码的长度，第 05 行判断两次的密码长度否相等，第 09 行判断两次的密码是否一样。

9.6 综合练习

1．创建个人信息表单，选择所在的学校，填写学生个人信息。填写姓名、学号等，学号必须为数字，并处理所提交的信息。三者不能为空，学号必须为数字。

学生信息为：北京师大 张三 200801。

所选的学校有：北京大学，清华大学，北京师大，中国人大，北京理工等。

提示：利用单选按钮选择学校信息，利用文本域填写姓名、学号信息，以 post 方法处理所提交的信息。其创建表单程序如示例代码 9-13 所示。

示例代码 9-13

```
01  <html>                                                <!--html 标签开始-->
02      <head>                                             <!--头标记开始-->
03      <title>个人信息表单</title>                         <!--文件标题-->
04      </head>                                            <!--头文件结束-->
05      <body>                                             <!--body 标签开始-->
06      <form action="action.php" method="POST">           <!--表单开始标记-->
07      请选择你的学校名称:                                 <!--输出说明文字-->
08      <hr><p>                                            <!--输出横线-->
09          <input type="radio" name="college" value="北京大学">  北京大学
                                                           <!--单选按钮选项-->
10          <input type="radio" name="college" value="清华大学">  清华大学
                                                           <!--单选按钮选项-->
11          <input type="radio" name="college" value="北京师大">  北京师大
                                                           <!--单选按钮选项-->
12          <input type="radio" name="college" value="中国人大">  中国人大
```

```
                                                        <!--单选按钮选项-->
13          <input type="radio" name="college" value="北京理工"> 北京理工
                                                        <!--单选按钮选项-->
14           <input type="radio" name="college" value="北航"> 北航
                                                        <!--单选按钮选项-->
15           <input type="radio" name="college" value="其他"> 其他
                                                        <!--单选按钮选项-->
16       </p>
17       <p> 姓名: <input name="student" type="text" size="20" maxlength="20"></p>
                                                            <!--输入姓名-->
18       <p> 学号: <input name="numb" type="text" size="20" maxlength="20"></p>
                                                            <!--输入学号-->
19       <p> <input name="submit" type="submit" value="提交">
                                                            <!--提交按钮-->
20          <input name="reset" type="reset" value="重置">    <!--重置按钮-->
21       </p>
22       </form>                                            <!--表单结尾标记-->
23       </body>                                            <!--body 标签结束-->
24   </html>                                                <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-13.php，查看其运行结果，即可看到结果如图 9-18 所示。

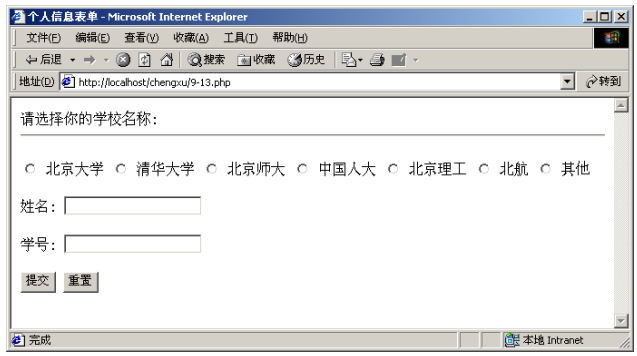

图 9-18　学生个人信息填写

处理信息页面程序 action.php 如下：

```
01  <?php                                          //PHP 开始标记
02      if (empty($_POST['college']))              //判断学校是否为空
03      echo "您没有选择学校名称。";                //输出“您没有选择学校名称。”
04      if (empty($_POST['student']))              //判断名字是否为空
05      echo  "您没有输入姓名。". "<br>";           //输出“没有输入姓名。”
06      if ($_POST['numb']<0)                      //判断学号是否为空
07      echo "您没有输入学号。". "<br>";            //输出“您没有输入学号。”
08          //输入的学号
09      if (!is_numeric($_POST['numb']))           //判断学号是否为数字
10      die ("学号应为数字。". "<br>");             //输出“学号应为数字。”
11      echo "<br>你的学校是:".$_POST['college']. "<br>";     //输出“学校”
12      echo "您的名字是：".$_POST['student']. "<br>";        //输出“名字”
13      echo "您的学号是：".$_POST['numb']. "<br>";           //输出学号
14  ?>                                             //PHP 结束标记
```

【运行结果】选择“北京师大”，填写用户名为“张三”，学号为“200801”，提交信息，即可看到如图 9-19 所示内容。

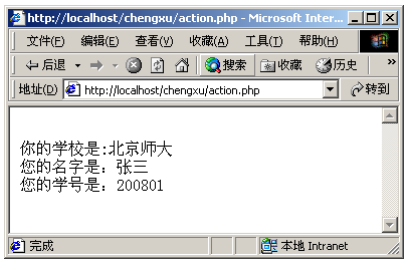

图 9-19　处理个人信息

2．利用下拉列表选择出生年月（2000—2008 年），选择日期 2008 年 8 月 8 号，并处理提交的信息，显示生日。

提示：建立表单，以 post 方法提交信息，利用<select>创建下拉列表。其创建表单的程序如示例代码 9-14 所示。

示例代码 9-14

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title>获取表单列表/菜单的数据</title>              <!--文件标题-->
    </head>                                               <!--头文件结束-->
    <body>                                                <!--body 标签开始-->
        <form action="birthday.php" method="POST">        <!--表单开始标记-->
            选择你的出生年月:                             <!--输出“选择你的出生年月:” -->
            <hr>                                          <!--显示横线-->
            <p> 出生年月:                                 <!--输出“出生年月:” -->
            <select name="select1" size="1">              <!--设置下拉列表名字-->
            <option value="2008" selected>2008</option>
                                                          <!--默认下拉选项-->
            <option value="2007">2007</option>            <!--年份下拉选项-->
        <option value="2006">2006</option>                <!--年份下拉选项-->
        <option value="2005">2005</option>                <!--年份下拉选项-->
        <option value="2004">2004</option>                <!--年份下拉选项-->
        <option value="2003">2003</option>                <!--年份下拉选项-->
        <option value="2002">2002</option>                <!--年份下拉选项-->
        <option value="2001">2001</option>                <!--年份下拉选项-->
        <option value="2000">2000</option>                <!--年份下拉选项-->
        </select> 年                                      <!--下拉选项结束-->
    <select name="select2">                               <!--设置下拉列表名字-->
    <option value="1">1</option>                          <!--月份下拉选项-->
    <option value="2">2</option>                          <!--月份下拉选项-->
    <option value="3">3</option>                          <!--月份下拉选项-->
    <option value="4">4</option>                          <!--月份下拉选项-->
    <option value="5">5</option>                          <!--月份下拉选项-->
    <option value="6">6</option>                          <!--月份下拉选项-->
    <option value="7">7</option>                          <!--月份下拉选项-->
    <option value="8">8</option>                          <!--月份下拉选项-->
    <option value="9">9</option>                          <!--月份下拉选项-->
    <option value="10">10</option>                        <!--月份下拉选项-->
    <option value="11">11</option>                        <!--月份下拉选项-->
    <option value="12">12</option>                        <!--月份下拉选项-->
    </select> 月                                          <!--输出“月” -->
    <select name="select3">                               <!--设置下拉列表名字-->
```

```
36      <option value="1">1</option>                    <!--日期下拉选项-->
37      <option value="2">2</option>                    <!--日期下拉选项-->
38      <option value="3">3</option>                    <!--日期下拉选项-->
39      <option value="4">4</option>                    <!--日期下拉选项-->
40      <option value="5">5</option>                    <!--日期下拉选项-->
41      <option value="6">6</option>                    <!--日期下拉选项-->
42      <option value="7">7</option>                    <!--日期下拉选项-->
43      <option value="8">8</option>                    <!--日期下拉选项-->
44      <option value="9">9</option>                    <!--日期下拉选项-->
45      <option value="10">10</option>                  <!--日期下拉选项-->
46      <option value="11">11</option>                  <!--日期下拉选项-->
47      <option value="12">12</option>                  <!--日期下拉选项-->
48      <option value="13">13</option>                  <!--日期下拉选项-->
49      <option value="14">14</option>                  <!--日期下拉选项-->
50      <option value="15">15</option>                  <!--日期下拉选项-->
51      <option value="16">16</option>                  <!--日期下拉选项-->
52      <option value="17">17</option>                  <!--日期下拉选项-->
53      <option value="18">18</option>                  <!--日期下拉选项-->
54      <option value="19">19</option>                  <!--日期下拉选项-->
55      <option value="20">20</option>                  <!--日期下拉选项-->
56      <option value="21">21</option>                  <!--日期下拉选项-->
57      <option value="22">22</option>                  <!--日期下拉选项-->
58      <option value="23">23</option>                  <!--日期下拉选项-->
59      <option value="24">24</option>                  <!--日期下拉选项-->
60      <option value="25">25</option>                  <!--日期下拉选项-->
61      <option value="26">26</option>                  <!--日期下拉选项-->
62      <option value="27">27</option>                  <!--日期下拉选项-->
63      <option value="28">28</option>                  <!--日期下拉选项-->
64      <option value="29">29</option>                  <!--日期下拉选项-->
65      <option value="30">30</option>                  <!--日期下拉选项-->
66      <option value="31">31</option>                  <!--日期下拉选项-->
67      </select> 日 </p>                               <!--输出“日”-->
68      <p>                                             <!--换行-->
69      <input name="submit" type="submit" value="提交"><!--设置提交按钮-->
70      <input name="reset" type="reset" value="重置">  <!--设置重置按钮-->
71      </form>                                         <!--表单结尾标记-->
72  </body>                                             <!--body 标签结束-->
73  </html>                                             <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/9-14.php，查看其运行结果，即可看到结果如图 9-20 所示。

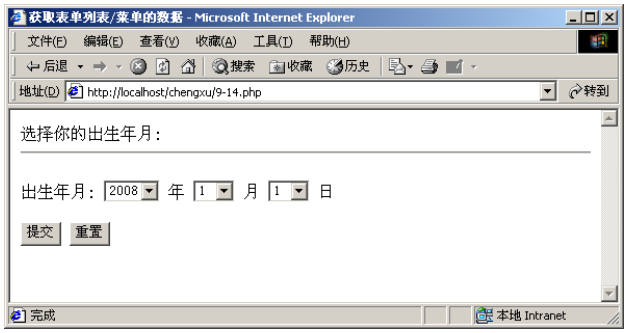

图 9-20　出生年月选择

处理页面程序 birthday.php 代码如下：

```
01  <?php                                                    //PHP 开始标记
02       echo "你生日的时间是:";                              //输出“你生日的时间是:”
03       /*显示时间。*/
04       if  (!empty($_POST['select1']))                     //判断年份
05       echo $_POST['select1']. "年";                       //输出年份
06       if (!empty($_POST['select2']))                      //判断月份
07       echo $_POST['select2']. "月";                       //输出月份
08       if (!empty($_POST['select3']))                      //判断日期
09       echo $_POST['select3']."日"."<p>";                  //输出日期
10  ?>                                                       //PHP 结束标记
11  <a href="9-14.php" title="返回首页。">返回首页</a>        //返回首页
```

【运行结果】选择 2008、8、8 后提交，即可看到如图 9-21 所示的内容。

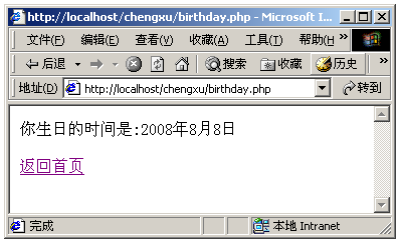

图 9-21　处理提交的生日信息

9.7　小结

本章主要讲述了表单的创建、处理方式、常用的表单的验证。通过本章的学习，可以掌握基本的表单创建方式和处理表单的方法。本章只是一个表单操作的基础，想了解更多的关于表单操作的知识，可以参阅《PHP Web 开发快速入门及实例精选》（陆昌辉等编著：电子工业出版社，2008）和《PHP 5 权威编程》（（美）古曼兹等著：电子工业出版社，2007）。

9.8　习题

一、填空题

1．表单以_____标记开始，以_____标记结束。
2．文字域的标记为_____。
3．密码域的标记为_____。
4．单选按钮标记为_____，复选框标记为_____。
5．_____，将信息送到指定地点，_____，重新填写表单信息。
6．下拉列表标记为_____。
7．提交表单信息的方式有_____、_____。
8．表单的验证一般包括_____、_____、_____。

二、选择题

1．表单的标签，是以（　　）开始的。

A．<table>　　B．<form>　　C．<tr>　　D．<input>

2．如果要用户实现多个选择时，选择下列哪个标记（　　）。

A．radio　　B．checkbox　　C．submit　　D．select

3．如果选项过多，为了节省空间，最好选择下列哪个标记（　　）。

A．radio　　B．checkbox　　C．submit　　D．select

4．为了保密，使输入的内容不被外界看到，选择下列哪个标记（　　）。

A．text　　B．file　　C．password　　D．reset

三、简答题

1．阐述 get 和 post 传递方式的差异。

2．阐述表单验证的方法。

四、编程题

1．制作一个表单，标题是个人信息。内容包括姓名（text），性别（radio，男、女）学历（text），爱好（checkbox，旅游、运动、读书、音乐、绘画），求职意向（radio，机械方面，电脑方面，苦力），提交（submit）、重置（reset）按钮。

2．编写一个注册用户界面，要求填写用户名、密码、电子邮件。并要求用户名的长度在3～12 之间，密码长度大于 6，电子邮件格式正确。

第 10 章　文件的基本操作

文件是所有操作系统数据存储的基本单位，根据类型和操作系统的不同，文件有多种格式和功能。文件就像记录东西的稿件，稿件的样式多种多样，每个稿件记录着不同的事件，有不同的功能。本章主要讲述对文件的判断、读写和目录的操作等内容。

学习本章，可以获得以下知识点：

- 文件访问；
- 文件读取；
- 写入文件；
- 上传文件；
- 目录操作。

10.1　文件访问

一般情况下，一个大型的网站由许多的文件构成，这些文件彼此之间的访问是非常常见的。文件的访问主要包括文件的判断、打开和关闭等操作，下面分别介绍这些操作。

10.1.1　判断文件是否存在

在对一个文件进行访问之前，一般需要先判断文件是否存在，因为用不恰当的方式访问一个并不存在的文件，就会导致错误。在 PHP 中，file_exists()函数能够检查文件是否存在，其结构形式如下：

file_exists($string)。

参数$string 为一个指向文件或目录的字符型变量，如果文件或目录存在，则返回值 true，反之返回 false。

【范例 10-1】如果网页需要调用 php.txt，首先判断文件是否存在。如果不存在，发出警告“指定的文件不存在”，如果存在，输出“可以调用文件”。其判断代码如示例代码 10-1 所示。

示例代码 10-1

```
01  <?                                                    //PHP 开始标记
02          //file.exists()使用实例
03      $filename="php.txt";                              //定义变量
04      if(file_exists(filename))                         //判断文件是否存在
05      {
06          echo "可以调用 $filename 文件";               //输出文件名
07      }
08      else                                              //当文件不存在时执行操作
09      {
10          echo "指定文件 $filename 不存在";             //输出文件名
11      }
12  ?>                                                    //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-1.php，查看其运行结果，即可看到结果如图 10-1 所示。

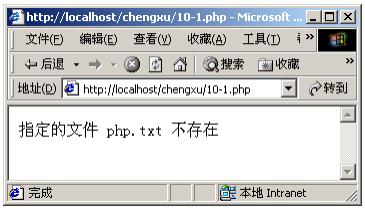

图 10-1　判断文件是否存在

【代码解析】代码第 03 行定义变量$filename，将要访问的文件名赋给变量。第 04 行判断文件是否存在。第 06、10 行根据判断结果输出。

10.1.2　访问文件属性

在进行处理文件之前，有时需要知道文件的一些属性，如文件的大小、类型、修改时间、访问时间和文件的权限等。PHP 提供了一些函数可直接获取文件的属性，如表 10-1 所示。

表 10-1　获取文件属性函数

函 数 名	作　用	参数及返回值
filesize(string)	获取文件的大小	参数 string 为一个指向文件或目录的字符型变量。函数的返回值为整型变量，返回文件的大小。如果出错，则返回 false。函数参数不能为远程文件，返回结果会被缓存
filetype(string)	获取文件的类型	参数 string 为一个指向文件或目录的字符型变量。函数的返回值为字符型变量，可能出现的值有 fifo，char，dir，block，link，file 和 unknown，返回值会被缓存
filemtime(string)	获取文件修改的时间	参数 string 为一个指向文件或目录的字符型变量。函数的返回值为整型变量，返回文件的修改时间
fileatime(string)	获取文件的访问时间	参数 string 为一个指向文件或目录的字符型变量。函数的返回值为整型变量，内容为文件的访问时间
fileperms(string)	获取文件	参数 string 为一个指向文件或目录的字符型变量。函数的返回值为整型变量，内容为

	的权限	文件相应的权限，同其他这一类的函数一样，参数不能为远程文件，另外返回结果同样会被缓存

【范例 10-2】在 chengxu 文件夹内创建 text.txt 文件，并在其中输出“php”字符串。保存文件，然后判断一个文件 text.txt 的大小、类型、访问时间、文件权限等属性。其代码如示例代码 10-2 所示。

示例代码 10-2

```
01  <?                                                          //PHP 开始标记
02      $filename="text.txt";                                   //定义变量
03      echo $filename."的大小为: ".filesize($filename)."<br>";  //输出文件大小
04      echo $filename."的类型为: ".filetype($filename)."<br>";  //输出文件类型
05          //使用 fileatime()函数并格式化返回日期
06      echo $filename."的修改时间为:".date("Y 年 n 月 t 日",fileatime($filen
ame))."<br>";                                                   //修改时间
07          //使用 fileatime()函数并格式化返回日期
08      echo $filename."访问时间为:".date("Y 年 n 月 t 日", fileatime($filen
ame))."<br>";                                                   //最后访问时间
09          //使用 fileperms()函数
10      echo $filename."的权限为 :".fileperms($filename)."<br>";  //输出文件权限
11  ?>                                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-2.php，查看其运行结果，即可看到结果如图 10-2 所示。

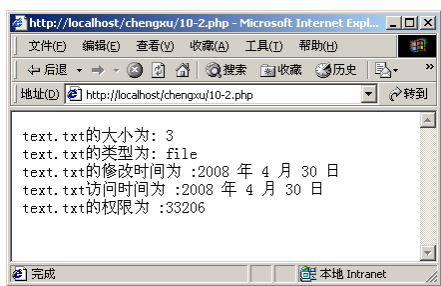

图 10-2　文件属性

【代码解析】代码第 02 行定义变量，第 03 行输出文件大小。第 04 行输出文件类型，第 06 行输出文件修改时间。第 08 行输出文件访问时间，第 10 行输出文件权限。

注意：在使用这个例子之前，要保证同级目录下所判断的文件存在。

10.1.3　打开文件

读取文件中的数据之前，必须要首先打开一个文件，这是其他文件操作的基础。PHP 使用 fopen()函数打开一个文件，其结构形式如下：

```
resource fopen (string  $filename, string $mode)
```

其中参数 filename 是所要打开的文件名，可以打开本地文件，其形式为“scheme://…”。也可打开远程 Web 服务器上的文件，其形式为“http://....”。如果其形式为以“ftp://...”开头，则其表示打开的文件是远程 FTP 服务器上的。参数 mode 是要打开文件的方式，参数$mode 是可以接收的模式。代码如表 10-2 所示，其中代码“b”和代码“t”仅作用于 Windows 系统。

表 10-2　文件打开方式

模　式	说　明
r	只读方式打开，将文件指针指向文件头
r+	读写方式打开，将文件指针指向文件头
w	写入方式打开，将文件指针指向文件头并将文件大小截为 0。如果文件不存在，则尝试创建一个新文件
w+	写入方式打开，将文件指针指向文件头并将文件大小截为 0。如果文件不存在，则尝试创建一个新文件
a	写入方式打开，将文件指针指向文件末尾。如果文件不存在，则尝试创建一个新文件
a+	写入方式打开，将文件指针指向文件末尾。如果文件不存在，则尝试创建一个新文件
x	创建和打开本地文件只用于写入，从文件起点开始。如果文件已经存在，则 fopen()返回假，而且 PHP 发送一个警告。如果文件不存在，则创建它
x+	创建和打开本地文件进行读取和写入，从文件起点开始。如果文件已经存在，fopen()返回假，而且 PHP 发送一个警告。如果文件不存在，则创建它
b	默认模式，与其他模式配合使用，用于对二进制文件和文本文件区别对待的文件系统。对于 Windows 是必需的
t	与其他文件配合使用，代表 Windows 文本文件。把行结束符\n 转换为\r\n。与 b 模式配合使用以提高可移植性

fopen()返回一个值，这个值包含一个文件句柄的整数，用来向执行文件操作的函数标识该文件。有时这个值被称为文件指针，指针就形如内存地址中的一个个小房间的门号。如果 PHP 打开文件失败，那么这个值是 false。利用常用的打开文件的方式打开本地文件和远程文件，例如：

```
01      $fh=fopen("../myfile/php.txt",'a');              //写入方式打开本地文件
02      $fh=fopen ("http://www.index.php/",  'r');       //打开远程 Web 服务器文件
03      $handle=fopen("ftp://www.hostname.com/ ");       //打开远程 FTP 服务器文件
04      $handle=fopen("myfile.txt","w");                 //打开本地文件
05      $handle=fopen("c:/data/phpo.txt","r");           //打开本地文件
06          //只使用于 Windows 系统
07      $handle=fopen("c:\\data\\phpo.txt","r");         //打开本地文件
```

代码第 01 行以写入方式打开本地文件，第 02 行 Web 服务器上文件，第 03 行打开的文件是远程 FTP 服务器上的文件。

注意：fopen()返回的文件指针可以在后续的文件操作函数中使用，这些函数包括：fclose()、feof()、fgets()等。

10.1.4　关闭文件

文件在打开操作后必须关闭，否则可能会引起错误。前面的示例已经使用了关闭文件函数，即 fclose()函数，fclose()函数的声明如下：

```
bool fclose(resource handle)
```

fclose 函数将 handle 指向的文件关闭，如果成功，则返回 True，失败，则返回 False。关闭打开的文件，例如：

```
01  $handle=fopen('text.txt','r');          //打开文件
02  fclose($handle);                        //关闭文件
```

代码第 01 行打开文件 text.txt，第 02 行利用 fclose()函数关闭打开的文件。

注意：文件指针必须有效，并且是通过 fopen()或 fsockopen()成功打开的。

10.2 读取文件

打开文件后，接下来一般要读取文件内容。在 PHP 中，有多个函数可以完成此项操作，如 readfile()读取整个文件的内容，fgetc()读取文件中的某一个字符。下面分别介绍这些函数。

10.2.1 读取文件相应字符

使用 fgetc()函数读取一个文件中的某一个字符，其结构形式如下：

```
string fgetc(resource $handle)
```

该函数的参数$handle 是已经被打开的文件指针，函数返回当前文件指针所指向的字符。如果文件指针指向文件末尾，则返回 false。

【范例 10-3】利用 fgetc()函数读取 text.txt 文件的开头的字符，其程序如示例代码 10-3 所示。

示例代码 10-3

```
01  <?                                          //PHP 开始标记
02      $filename=fopen("text.txt","r");        //打开文件
03      $char=fgetc($filename);                 //读取文件内容
04      echo $char;                             //输出变量值
05      fclose($filename);                      //关闭打开文件
06  ?>                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-3.php，查看其运行结果，即可看到结果如图 10-3 所示。

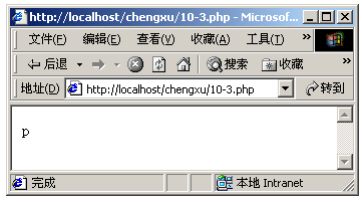

图 10-3 读取相应字符

【代码解析】代码第 02 行打开文件，第 03 行文件指针指向文件头，指针处内容赋值给变量。第 04 行输出变量，第 05 行关闭文件。

10.2.2 读取整个文件

读取整个文件内容，可以通过循环实现。下面通过实例讲解如何利用循环读取整个文件。

【范例 10-4】打开 text.txt 文件，另起一行输入“读取整个文件。”，保存文件，然后利用 do 循环读取 text.txt 文件的整个内容。其程序如示例代码 10-4 所示。

示例代码 10-4

```
01  <?                                          //PHP 开始标记
02      //循环读取文件所有内容
```

```
03      $filename=fopen("text.txt","r");    //用只读打开文件指针指向文件开头
04          //通过循环判断指针是否指向文件末尾
05      do
06      {
07          $mychar=fgetc($filename);       //读取文件指针处字符并赋值给变量
08          echo $mychar;                   //显示变量
09          }while(!feof($filename)) ;      //判断是否结束
10      fclose($filename);                  //关闭打开的文件
11  ?>                                      //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-4.php，查看其运行结果，即可看到结果如图 10-4 所示。

图 10-4　读取整个文件

【代码解析】代码第 03 行打开文件，第 05～09 行执行 do…while 循环，显示 text.txt 文件内容。第 10 行关闭文件。

注意：打开的文件不同，输出的结果也不一样，查看打开页的源文件，会发现内容与 test.txt 完全一样，说明文件内容被正确读取。

10.2.3　按行读取

使用 fgets()函数按行读取文件内容，其结构形式如下：

```
string fgets(resource $handle[,int $length])
```

参数$handle 是已经被打开的文件句柄，参数$length 为要读取的字符的个数。函数返回当前文件指针所指向行指定的字符个数，如果文件指针指向文件末尾，则返回 false。

【范例 10-5】读取文件 text.txt 的第一行的内容，其程序如示例代码 10-5 所示。

示例代码 10-5

```
01  <?                                      //PHP 开始标记
02      $myfile=fopen("text.txt","r");      //打开文件
03      $file=fgets($myfile,1024);          //用 fgets()读取文件指针处行并赋值给变量
04      echo $file;                         //显示变量
05      fclose($myfile);                    //关闭打开的文件
06  ?>                                      //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-5.php，查看其运行结果，即可看到结果如图 10-5 所示。

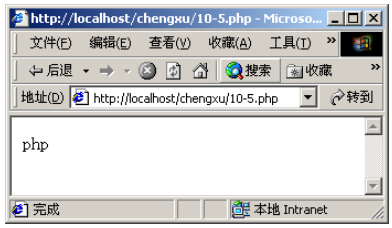

图 10-5　按行读取

【代码解析】代码第 02 行以只读方式打开 text.txt 文件，指针在文件开头。第 03 行读取文件第一行，并将结果赋给变量$file。第 04 行输出结果，第 05 行关闭文件。

提示：如果查看 HTML 文件开头，会发现什么也没有，但查看源文件就会发现“<html>”。因为文件指针指向文件头，指针指向的第一行将“<html>”赋值给了变量。

10.3　写入数据

PHP 不但能实现对文件的读取，也能对文件进行改写。文件的写入按写入次数主要分单行写入和多行写入，单行写入即一次只能写入一行文字，多行写入即一次能写入多行文字。下面分别介绍这两种写入方式。

10.3.1 写入单行数据

fwrite()函数可以实现单行写入文件，其结构形式如下：

```
int fwrite(resource $handle, string $string [,int $length])
```

参数 handle 是要被写入的文件，参数 string 是要写入的内容，参数 length 是要写入的长度。如果指定了$length，当写入了$length 个字节或者写完了$string 以后，写入就会停止。fwrite()返回写入的字符数，出现错误时则返回 false。

【范例 10-6】创建文件 onlyone.txt，然后以只写方式打开文件。利用 fwrite()函数将“我后被写入的!”和“我后被写入的!”写入到文件 onlyone.txt 中，程序如示例代码 10-6 所示。

示例代码 10-6

```
<?php                                                    //PHP 开始标记
    $filename='onlyone.txt';                             //将文件名付给变量
    $wr="我先被写入的! ";                                 //变量赋值
    $wr1="我后被写入的! ";                                //变量赋值
        //首先要确定文件存在并且可写
    if(is_writable($filename))                           //判断是否可写
        {
        //如果原来文件中有其他数据，则会被删除
        if(!$handle=fopen ($filename,"w")){              //判断文件是否被打开
            print "不能打开文件 $filename ";              //输出“不能打开文件”
            exit;                                        //跳出 if 语句
        }
            //将$wr 写入到打开的文件中
        if(!fwrite($handle,$wr)){                        //判断是否写入成功
            print "不能写入到文件 $filename";             //输出“不能写入到文件”
            exit;                                        //跳出 if 语句
        }
        print "成功地将 \"$wr\" 写入到文件$filename<br>";
                                                         //输出成功信息
        fclose($handle);                                 //关闭打开的文件
        $handle=fopen($filename,"w");                    //以写入方式打开文件
        fwrite($handle,$wr1);                            //将“我后被写入的!”写入到文件
        fclose($handle);                                 //关闭文件
        print "成功地将 \"$wr1\" 写入到文件$filename";
                                                         //输出写入成功信息
    }
    else{                                                //文件不可写的情况
        print "文件 $filename 不可写";                    //输出“文件不可写”
    }
?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-6.php，查看其运行结果，即可看到结果如图 10-6 所示。

【代码解析】代码第 02～04 行定义三个变量，第 04 行判断文件是否可写。第 09 行判断文件是否打开，第 14 行将变量$wr 的值写入文件，第 21 行将变量$wr1 的值写入文件。打开 onlyone.txt 文件可以看到只有一行文字如图 10-7 所示，因为以函数写入，是单行写入，所以当写入新内容后，原来的内容被覆盖掉。

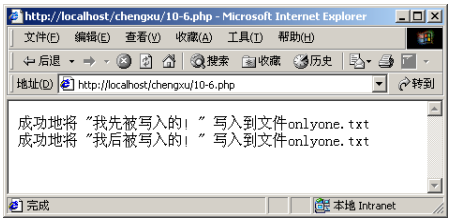

图 10-6　写入单行数据

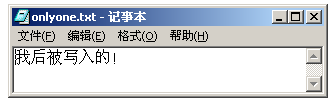

图 10-7　onlyone.txt 文件

提示：上述示例虽然两句话都被写入到文件中，但实际上只有最后一句话被写入文件了。这是因为使用了只写模式打开，在每次写入新数据时，原始数据都会被删除。

10.3.2　写入多行数据

单行数据写入在每次重新打开文件后，都会删除原来的数据，这样很不方便。比如要将个人的基本信息保存在一个文本文件中，但不是一次性写入而是需要经常打开文件追加，这个时候，就需要使用多行写入数据来完成。

【范例 10-7】创建文件 many.txt，将下列学生信息写入到文件中。

姓名：张三　　年龄:15　　性别：男
姓名：王二　　年龄:14　　性别：男
姓名：李四　　年龄:16　　性别：男
姓名：张丽　　年龄:14　　性别：女

其程序如示例代码 10-7 所示。

示例代码 10-7

```
<?php                                                    //PHP 开始标记
    $filename='many.txt';                                //将文件名赋给变量
    $student1="姓名：张三\t 年龄:15\t 性别：男\r\n";      //变量赋值
    $student2="姓名：王二\t 年龄:14\t 性别：男\r\n";      //变量赋值
    $student3="姓名：李四\t 年龄:16\t 性别：男\r\n";      //变量赋值
    $student4="姓名：张丽\t 年龄:14\t 性别：女\r\n";      //变量赋值
        //首先使用只写模式打开$filename
    If(!$handle=fopen($filename,"w")){                   //判断是否打开文件
        print "不能打开文件 $filename ";                 //输出“不能打开文件”
        exit;                                            //跳出 if 语句
    }
    //将$student1 写入打开的文件中
    if(!fwrite($handle,$student1)){                      //判断是否写入成功
        print "不能写入到文件$filename";                 //输出写入失败信息
    exit;                                                //跳出 if 循环
    }
    print  "成功地将 \" $student1\" 写入到文件$filename<br>";
                                                         //输出成功信息
    fclose($handle);                                     //关闭文件
        $handle=fopen($filename,"a");                    //以写入方式打开，指针指到末尾
        //继续添加其他信息
    fwrite($handle,$student2);                           //将变量 student2 的内容写入文件
    print "成功地将 \" $student2\" 写入到文件$filename<br>";
                                                         //输出成功信息
```

```
23        fwrite($handle,$student3);                    //将变量 student3 的内容写入文件
24        print "成功地将 \" $student3\" 写入到文件$filename<br>";
                                                        //输出成功信息
25        fwrite($handle,$student4);                    //将变量 student4 的内容写入文件
26        print "成功地将 \" $student4\" 写入到文件$filename<br>";
                                                        //输出成功信息
27        fclose($handle);                              //关闭文件
28    ?>                                                //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-7.php，查看其运行结果，即可看到结果如图 10-8 所示。

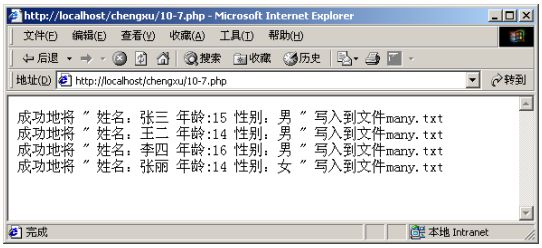

图 10-8　写入多行数据

【代码解析】代码第 03～07 定义了 4 个变量，第 08 行判断文件是否被打开。第 13 行判断变量$student1 是否被写入文件，第 19 行以写入方式打开文件，指针指到末尾。第 21、23、25 行将变量的值写入文件，第 27 行关闭打开的文件。

提示：打开 3.txt 文件可以看到写入的行文字。

10.4　指针

文件被打开后内部文件指针位于文件起始位置。当对文件执行完一定的操作后，很难判断指针的位置了，如果在此时进行文件读取或写入操作很容易引起错误。PHP 能通过一些函数来控制指针的位置。下面分别介绍这些函数。

10.4.1　查找指针位置

函数 ftell()用来查找指针位置，其结构形式如下：

```
int ftell(resource  $handle)
```

参数$handle 为要查找指针的文件，通过该函数能够判断文件指针的位置，以整数形式输出。

【范例 10-8】打开 many.txt，将指针向后移动 15 位，然后利用 ftell()函数查找指针位置。其程序如示例代码 10-8 所示。

示例代码 10-8

```
01  <?php                                               //PHP 开始标记
02      $fp=fopen("many.txt","r");                      //打开文件
03      //获得前 15 个字符
04      $data=fgets($fp,15);                            //将指针后移 15 个字节
05      //获取当前指针
```

```
06      echo ftell($fp);                                    //输出当前位置
07      fclose($fp);                                        //关闭打开文件
08  ?>                                                      //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-8.php，查看其运行结果，即可看到结果如图 10-9 所示。

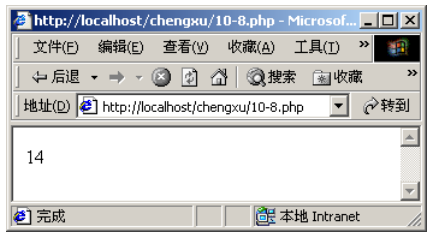

图 10-9　查找指针位置

【代码解析】上述代码第 02 行打开文件 many.txt，第 04 行指针向后移动 15 个字节，第 06 行输出指针位置。

提示：指针从 0 开始计算，所以返回位置为 14。

10.4.2　rewind()函数

rewind()函数文件位置指针设到文件的开头，其结构形式如下：

```
bool rewind (resource handle)
```

如果成功则返回 true，失败则返回 false。

【范例 10-9】读取 many.txt 文件前三行内容，然后重新读取第一行内容，其程序如示例代码 10-9 所示。

示例代码 10-9

```
01  <?php                                                  //PHP 开始标记
02      $handle=fopen("many.txt","r");                     //打开文件
03          //读取首行
04      $buffer=fgets($handle,1024);                       //读取第一行文字
05      echo $buffer ."<br>";                              //输出读取内容
06          //读取第二行
07      $buffer=fgets($handle,1024);                       //接着读取第二行
08      echo $buffer ."<br>";                              //输出读取内容
09          //读取第三行
10      $buffer=fgets($handle,1024);                       //接着读取第三行
11      echo $buffer ."<br>";                              //输出读取内容
12          //将指针回到文件开始，继续读取第一行数据。
13      rewind($handle);                                   //将指针指到开头
14      $buffer=fgets($handle,1024);                       //读取第一行内容
15      echo $buffer ."<br>";                              //输出读取内容
16      fclose($handle);                                   //关闭打开文件
17  ?>                                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-9.php，查看其运行结果，即可看到结果如图 10-10 所示。

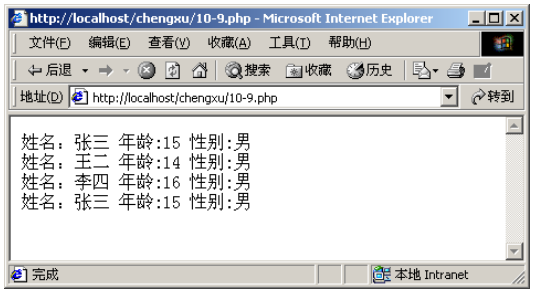

图 10-10　返回指针

【代码解析】代码第 02 行打开文件，第 04 行读取文件首行，第 07 行读取文件第二行。第 11 行将指针置于文件开头，第 12 行读取文件第一行。

10.4.3　指针定位

fseek()函数在文件中设定文件指针位置，该函数结构形式如下：

```
int fseek (resource $handle, int $offset [,int $whence])
```

参数$offset 为要指针要移动的位置。新指针的位置从文件头开始以字节数度量，也可以在参数$whence 指定的位置加上$offset。参数$whence 的值定义如下所示。

- seek_set：设定位置等于$offset 字节
- seek_cup：设定位置为当前位置加上$offset.
- seek_end：设定位置为文件尾加上$offset（要移动到文件尾之前的位置，需要给 offset 传递一个负值）。

提示：如果没有设定参数$whence，则默认为 seek_set。如果调用成功则返回 0，否则返回-1。

【范例 10-10】读取 many.txt 文件第 1 行内容，然后从第 7 个字节开始重新读取第 1 行内容。最后，指针回到文件开头，再次读取第 1 行内容。其程序如示例代码 10-10 所示。

示例代码 10-10

```
01  <?php                                                    //PHP 开始标记
02      $handle=fopen ("many.txt","r");                      //打开文件
03          //首先读取第 1 行，读取完毕后，指针在最后一行
04      $buffer=fgets($handle,1024);                         //读取第 1 行内容
05      echo $buffer ."<br>";                                //输出读取结果
06          //接着将指针向后移动 5 个字节
07      fseek($handle,6,seek_cup);                           //指针向后移动 6 个字节
08          //接着读取一行
09      $buffer=fgets($handle,1024);                         //接着读取第 1 行内容
10      echo $buffer . "<br>";                               //输出读取结果
11          //将指针返回到到文件开始，重新读取第 1 行，因为是绝对位置为 0
12      fseek($handle,0);                                    //指针返回到文件开始
13          //读取一行数据，因为指针回到文件开始，所以会读取第 1 行数据
14      $buffer=fgets($handle,1024);                         //再读取第 1 行文字
15      echo $buffer ."<br>";                                //输出读取内容
16      fclose ($handle);                                    //关闭打开文件
17  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-10.php，查看其运行结果，即可看到结果如图 10-11 所示。

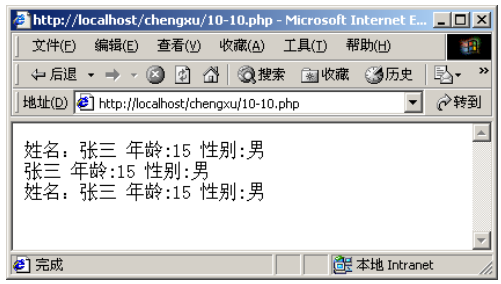

图 10-11　指定指针位置

【代码解析】代码第 02 行打开文件。第 04 行读取文件第 1 行。第 07 行指针向后移动 5 个字节，因为是相对于当前位置，所以需要设定 seek_cup 参数。第 09 行接着读取第 1 行。第 13 行将指针放置文件开头。第 15 行读取文件第 1 行。

10.5　目录操作

文件一般存放在某个文件夹下，将其在文件下的位置列出，就称为目录。常用目录的操作主要包括目录的创建、目录的遍历、目录的删除等。下面分别介绍这几种常用的操作。

10.5.1　打开目录

函数 opendir()用来打开一个目录，其语法结构如下：

```
opendir(string $path)
```

参数$path 为打开目录的路径，函数将返回一个打开目录的句柄，用于存储当前的目录资源，打开目录前像打开文件一样，要先检测目录是否存在，如果打开一个不存在的目录，程序将出错。

【范例 10-11】利用函数 opendir()，打开 D 盘下的 images 目录，其程序如实例代码 10-11 所示。

示例代码 10-11

```
01  <?php                                          //PHP 开始标记
02      $dir="D://images";                             //设置目录路径
03      if(is_dir($dir)){                              //判断是否为目录
04          $dp=opendir($dir);                         //打开目录
05          print_r("目录已被打开");                    //输出“目录已被打开”
06      }
07      else                                       //不是目录的情况
08      echo "目录不存在";                          //输出“目录不存在”
09  ?>                                             // PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-11.php，查看其运行结果，即可看到结果如图 10-12 所示。

【代码解析】代码第 02 行设置打开的目录的路径，第 03 行判断是否是目录，第 04 行打开目录，第 05 行输出打开成功信息，第 08 行输出目录不存在的信息。

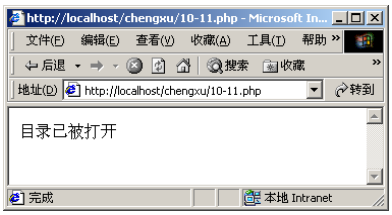

图 10-12　打开目录

10.5.2　关闭目录

关闭文件利用函数 closedir()，其语法结构为：

```
closedir($dp)
```

参数 dp 为使用函数 opendir()打开的目录返回的资源对象，以下是一个关闭目录的实例。

```
01  <?php                                    //PHP 开始标记
02      $dir="D://images";                       //设置目录路径
03          $dp=opendir($dir);                   //打开目录
04          closedir($dp);                       //关闭目录
05  ?>                                       // PHP 结束标记
```

代码第 02 行设置目录路径，第 03 行打开目录，第 04 行关闭刚才打开的目录。

提示：关闭成功不返回 1，故不能通过 if 语句判断其是否关闭成功。

10.5.3　读取目录

在 PHP 中使用 readdir()函数读取目录中的文件并返回文件名，其语法结构如下：

```
readdir(($dp)
```

参数 dp 为使用函数 opendir()打开的目录返回的资源对象，函数返回目录下的文件名。下面通过示例讲解读取目录的方法。

【范例 10-12】利用函数 opendir()，打开 D 盘下的 images 目录，并读取其下面的文件，其程序如实例代码 10-12 所示。

示例代码 10-12

```
01  <?php                                    //PHP 开始标记
02      $dir="D://images";                       //设置目录路径
03      if(is_dir($dir)){                        //判断是否为目录
04          $dp=opendir($dir);                   //打开目录
05          while($filen=readdir($dp)){          //读出目录中的每一个子项
06          print_r($filen."<br>");}             //输出文件或目录名
07      }
08      else                                 //不是目录的情况
09      echo "目录不存在";                       //输出“目录不存在”
10  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-12.php，查看其运行结果，即可看到结果如图 10-13 所示。

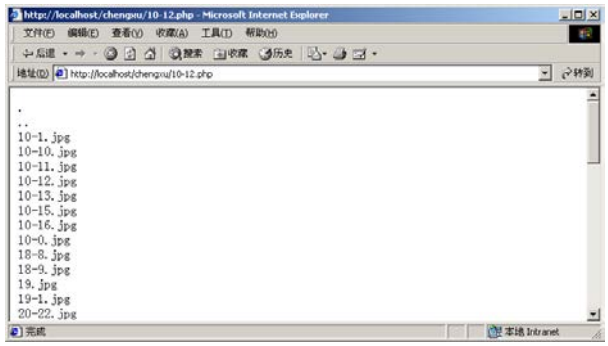

图 10-13　读取目录

【代码解析】程序第 02 行设置打开的目录的路径，第 03 行判断是否是目录，第 04 行打开目录，第 05 行利用 while 循环逐个读取目录下的文件或目录名，第 06 行输出读取结果，第 09 行输出目录不存在的信息。

提示：如果读取目录下的目录，是返回目录的文件名，则不会再读取其子目录下的文件名。

10.5.4　遍历目录

一个目录下往往会存放很多文件，如果想查看某个目录下存放的文件，可通过遍历目录，显示目录下的所有文件信息。目录就是一个典型的树形结构，实现遍历的方法很多，这里通过实例讲解目录的遍历。

【范例 10-13】利用递归实现遍历 D 盘 images 文件夹下所有的文件信息，其程序如示例代码 10-13 所示。

示例代码 10-13

```
01  <table border="1">                                  //定义表格
02  <tr>                                                //开始行
03      <th>文件</th>                                   //列标记
04  </tr>                                               //结束行
05  <?php                                               //PHP 开始标记
06      function Directory($dir) {                      //声明 Directory 函数
07          $dp = opendir($dir);                        //建立操作目录的句柄
08          while ($file = readdir($dp)) {              //读出目录中的每一个子项
09              if ($file!='.' && $file!='..') {        //判断是否含有“.”“..”的文件
10              $path = $dir ."/". $file;               //取得目录或文件路径
11              if (is_dir($path)) {                    //判断是否为目录
12                  echo "<B>目录</B>:" .$path;        //输出目录路径
13                  echo "<br>";                        //换行
14                  Directory($path);                   //返回路径
15                  }
16              else {                                  //是文件的情况
17                  echo "<tr>";                        //开始行
18                  echo "<td>" .$path."</td>";         //输出文件路径
19                  echo "</tr>";                       //结束行
20                  }
21              }
22          }
```

```
23            closedir($dp);                          //关闭目录
24        }
25            //调用
26        Directory(realpath("D:\\images"));          //实例化函数
27    ?>                                              //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-13.php，查看其运行结果，即可看到结果如图 10-14 所示。

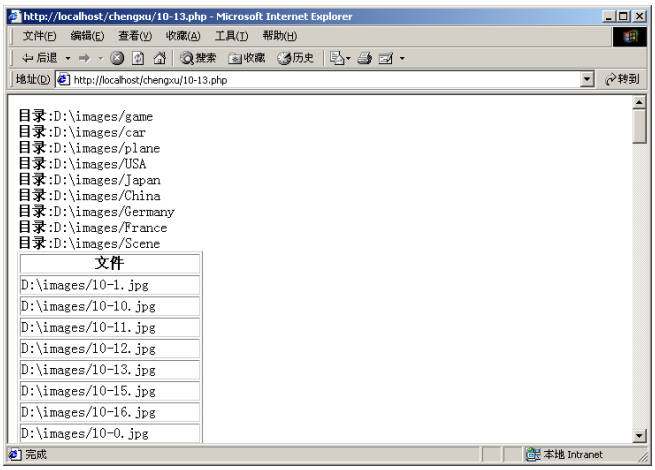

图 10-14　遍历目录

【代码解析】代码第 01~04 行定义一个表格。第 06 行声明名为 Directory 的一个函数。第 07 行打开文件。第 11 行判断是否为目录。第 14 行返回路径参数。第 18 行输出文件路径，第 26 行实例化 Directory 函数。

提示：利用函数 opendir()打开目录，读取目录利用函数 readdir()。

10.5.5　创建目录

在备份当天数据或存放用户注册信息时，经常需要创建目录，mkdir()函数可以实现目录的创建，其结构形式如下：

```
mkdir(string $dirname，int $mode)
```

参数$dirname 为字符变量，内容为想要创建目录的名称。参数$mode 为整型变量，表示创建模式。执行此函数将在指定目录下创建新的目录。

【范例 10-14】在 chengxu 文件夹内创建存放照片的文件夹 pic，其程序如示例代码 10-14 所示。

示例代码 10-14

```
01  <?                                              //PHP 开始标记
02          //使用 mkdir()函数创建目录
03      $dirname="pic";                             //定义变量
04      $str=mkdir($dirname,100);                   //用 mkdir 创建目录
05      if($str)                                    //判断创建结果
06      echo "创建成功";                            //输出“创建成功”
07      else                                        //创建失败的条件
```

```
08        echo "创建失败" ;                          //输出“创建失败”
09    ?>                                            //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-14.php，查看其运行结果，即可看到结果如图 10-15 所示。

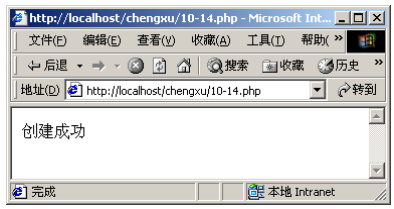

图 10-15　创建目录

【代码解析】代码第 03 行定义变量，第 04 行创建目录。运行提示成功后，则目录已被创建，如果已存在，则提示错误信息。

警告：创建的目录名不能与已经存在的目录名相同，如果出现了同样的目录名，就会出现出错提示信息。

10.5.6　删除目录

PHP 提供一个 rmdir()函数可以用来删除目录，其结构形式如下：

```
rmdir(string  $dirname)
```

参数$dirname 为字符变量，为要删除目录的名称。

【范例 10-15】删除 D 盘 images 的目录，其程序如示例代码 10-15 所示。

示例代码 10-15

```
01   <?                                            //PHP 开始标记
02       $pathdir="D:\\images";                    //指定目录路径
03       rmdir($pathdir);                          //删除目录
04   ?>                                            //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-15.php，查看其运行结果，即可看到结果如图 10-16 所示。

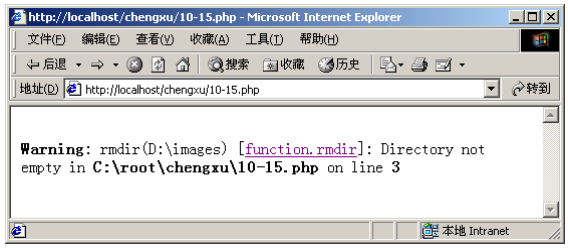

图 10-16　删除目录

【代码解析】代码第 02 行指定删除目录的路径，第 03 行删除指定的目录。因为目录不为空，所以程序出错。

警告：使用 rmdir()删除目录时，目录必须为空，否则将出错。

10.6 综合练习

1．将 GOOD LUCK 写入到 zhufu.txt 文件中。

提示：以写入方式打开文件，利用 fwrite()函数将数据写入文件。也可以通过其他方式实现，读者可以自己编写其他方式实现写入，其程序如示例代码 10-16 所示。

示例代码 10-16

```
<?php                                          //PHP 开始标记
    $filename='zhufu.txt';                     //变量赋值
    $handle=fopen($filename,"w");              //以写入方式打开文件
    fwrite($handle,"G\r\n");                   //写入字母“G”
    fwrite($handle,"O\r\n");                   //写入字母“O”
    fwrite($handle,"O\r\n");                   //写入字母“O”
    fwrite($handle,"D\r\n");                   //写入字母“D”
    fwrite($handle,"L\r\n");                   //写入字母“L”
    fwrite($handle,"U\r\n");                   //写入字母“U”
    fwrite($handle,"C\r\n");                   //写入字母“C”
    fwrite($handle,"K\r\n");                   //写入字母“K”
    fclose($handle);                           //关闭文件
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-16.php，运行结果代码后，打开 zhufu.php 文件即可看到如图 10-17 所示的结果。

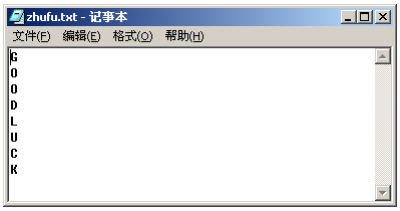

图 10-17 写入 GOOD LUCK

2．遍历目录，并显示该目录下文件的大小、类型等属性。

提示：遍历目录可以通过 while()循环实现，文件的属性可通过 10.1.2 节讲述的函数来实现。为了显示的条理性，可建立相应的表格。其程序如示例代码 10-17 所示。

示例代码 10-17

```
<?                                                          //PHP 开始标记
                                                            // 使用表格浏览目录的结构
    print("<table border=\"1\">\n");                        //开始建立表格
                                                            // 创建表格的头
    print("<tr><font color='red'>\n");                      //开始行
    print("<th>文件名</th>\n");                             //输出“文件名”
    print("<th>文件大小</th>\n");                           //输出“文件大小”
    print("<th>文件类型</th>\n");                           //输出“文件类型”
    print("<th>文件修改时间</th>\n");                       //输出“文件修改时间”
    print("</font></tr>\n");                                //结束行
    $Directory = opendir(".");                              //建立操作目录的句柄
                                                            // 读出目录中的每一个子项
    while($eName = readdir($Directory))                     //读出目录中的每一个子项
    {
```

```
15          print("<tr>");                                    //开始行
16          print("<td>$eName</td>");                         //输出文件名
17          print("<td ALIGN=\"right\">");                    //右对齐
18          print(filesize($eName));                          //输出文件大小
19          print("</td>");                                   //结束行
20      print("<td ALIGN=\"right\">");                        //右对齐
21      print(filetype($eName));                              //输出文件类型
22      print("</td>");                                       //结束列
23      print ("<td>");                                      //开始列
24      print (date("Y 年 n 月 t 日",filemtime($eName)));     //输出修改日期
25      print "</td>";                                        //结束列
26      print("</tr>\n");                                     //结束行
27      }
28      closedir($Directory);                                 //关闭目录
29      print("</table>\n");                                  //表格结束标记
30  ?>                                                        //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/chengxu/10-17.php，查看其运行结果，即可看到结果如图 10-18 所示。

图 10-18　遍历目录

提示：遍历所得结果和本地目录下的文件有关，所以得到的内容会不一样，但只要程序运行正确即可。

10.7　小结

本章主要介绍了 PHP 编程中对文件的基本操作，包括判断文件是否存在、获取文件属性、读取文件内容、写入文件、上传文件、遍历目录、创建目录等操作。文件的操作在 PHP 编程中有着广泛的应用，所以熟练掌握这些函数的使用及文件操作十分必要。

本章只是一个文件操作的基础，想了解更多的关于文件操作的知识，可参阅《PHP Web 开发快速入门及实例精选》（陆昌辉等编著.电子工业出版社,2008）和《PHP 5 权威编程》（（美）古曼兹等著．电子工业出版社，2007）。

10.8 习题

一、填空题

1．对一个文件进行访问之前，一般需要先判断文件是否存在，利用 file_exists()函数判断文件是否存在，如果文件或目录存在，则返回值_____，反之返回_____。

2．在进行处理文件之时，要知道文件的大小、类型、访问时间、文件权限的属性，分别用到的函数是_____、_____、_____、_____。

3．如果可以打开本地文件，其形式为_____；如果打开远程 Web 服务器上的文件，其形式为_____；如果打开远程 FTP 服务器上文件，其形式为_____。

4．读取一个文件中的某一个字符的函数为_____，可以读取文件某一行的内容的函数为_____。

5．函数_____可以实现单行写入文件。

6．函数_____用来查找指针位置，函数_____在文件中设定文件指针位置。

7．用来打开文件的函数为_____，用来关闭打开的文件的函数是_____。

8．创建目录的函数为_____，函数_____可以用来删除目录。

二、选择题

1．以只读方式打开，将文件指针指向文件头的代码是（　　）。

A．r　　B．r+

C．w　　D．w+

2．通过 rewind()函数，可将指针设置到（　　）。

A．文件结尾　　B．文件开头

C．任意位置　　D．不能操作指针

3．选择下面程序的运行结果（　　）。

```
$fp=fopen("many.txt","r");
//获得前 15 个字符
$data=fgets($fp,1);
//获取当前指针
echo ftell($fp);
fclose($fp);
```

A．0　　B．1

C．2　　D．程序有误

4．多次运行下列程序，其运行结果为（　　）。

```
01      $dirname="php";
02      $str=mkdir($dirname,10);
03      if($str)
04      echo "创建成功";
05      else
06      echo "创建失败" ;
```

A．第 02 行发生编译错误　　B．输出“创建成功”

C．输出“创建失败”　　D．输出“创建失败”并提示错误信息。

三、简答题

1．试说明文件处理的步骤。

2．试列出打开文件的方式。

3．说明如何向文件中写入数据。

四、编程题

1．将获奖同学的名单输入到 zhufu.txt 文件下。

一等奖：王轮

二等奖：张静、李丽

三等奖：赵无、丁一、王六

2．遍历 C 盘 root 文件，并输出该目录下文件的名、文件大小、文件类型、访问时间等信息。

第 11 章　类和对象

通过前几章对 PHP 语言中的常量、变量、函数等介绍，相信读者已经可以掌握基本的 PHP 语言。有时在编写程序时经常会遇到很复杂的程序，令人倍感头痛。但在引入类和对象之后能使程序简化，把原来庞大的程序分割成多个小的具有一定功能的模块。就好像一台大的电视机由一个一个小的零件组成，这一个个不同功能的零件构成了一个大的功能强大的电视机。

学习本章，可以获得以下知识点：

- 类的创建；
- 类的访问控制；
- 构造函数与析构函数；
- static 关键字；
- $this 关键字。

11.1　类的使用

PHP 提供了面向对象的编程功能，面向对象语言的包括类（class）、对象（object）、函数（function）或方法（method）等。下面先来介绍类的概念和使用方法。

11.1.1　类和对象

类是一组具有相同属性和行为的对象的抽象，是抽象的、概念上的定义。类仅仅是对象的描述，就好像设计好的零件图纸，而这个零件并不存在。

对象是实际上存在的该类事物的个体，就像照着图纸制造好的零件一样，如图 11-1 所示。对象都属于某一个类，而每个对象都是某个类的实例。例如水星、金星、地球都是行星类，而地球就是行星类的一个实例。对象是一个封装数据属性和行为的实体，属性是指事物本身具有的特征。例如人的姓名、性别、身高、体重等或者产品的品牌、价格、功能等都是对象的属性。

同一个类的所有对象具有相同的属性，属性含义相同，但状态可以不一样。水星、金星都有行星属性，但属性取值不同。

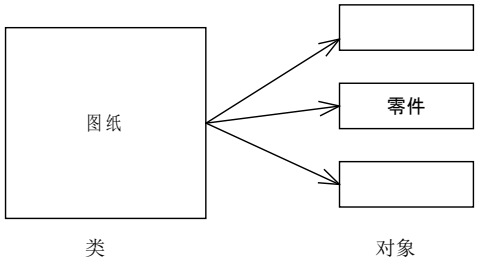

图 11-1　类和对象的关系

提示：一个对象可以有多个属性和多个行为，对象的属性和行为构成一个整体。

11.1.2 创建类

在 PHP 中，用户可以创建类，即声明类，声明类即是在要创建的类名前加上关键字 class，类名可以是任何非 PHP 保留字。其一般创建语法为：

```
class class_name
{
var  属性1;
var  属性2;
function function_name1{
        //method declarations defined here
}
function function_name2{
        //method declarations defined here
}
}
```

代码说明如下所示。

- class 是声明类的关键字；
- class_name 是创建的类名；
- 大括号内是类的声明的变量和行为。

例如：使用关键字 class 声明一个学生类：

```
01  <?php                                    //PHP 开始标记
02      class Student{                       //声明 Studen 类
03      var $name                            //设定属性 name
04      }
05  ?>                                       //PHP 结束标记
```

程序第 02 行声明了 Student()类，第 03 行为设置的 name 属性，这样就拥有了第一个 PHP 类。

提示：PHP 中使用关键字 class 来定义一个类，类名一般首字符大写。

11.1.3 实例化类

创建类是为了使用这个类，使用类就是要将这个类实例化。实例化过程就像按照图纸生产的一个个零件，实例化的方式是使用关键字 new 加上类名，这个过程也称创建对象。例如，创建学生类后，实例化类。

```
01      class Student{                       //声明 Studen 类
02      var  $name                           //设定属性 name
03      }
04      $p=new Student();                    //实例化
```

代码第 01～03 行声明一个 Student 类，第 04 行实例化一个 Student 类。

11.1.4 方法的使用

对对象的属性进行的操作称为对象的方法。方法（function）与前面第 5 章讲的函数非常相似，只是此处的函数用来定义特定类的行为。方法与函数一样，可以直接接受输入参数，可以为调用者返回一个值，其声明方法也和前面讲的函数相同，只是在对象的名称前面加 function，其一般形式为：

```
function functionName( )
{
  // method declarations defined here
}
```

【范例 11-1】创建 Student 类，定义变量 name，创建 setName()方法设置$name 数据成员。创建 getName()方法，返回$name 值。实例化对象，输出名字“Li Lei”，程序如示例代码 11-1 所示。

示例代码 11-1

```
01  <?php                                          //PHP 开始标记
02      class Student{                             //声明 Student 类
03      var $name;                                 //设定属性
04      function setName($name){                   //声明方法 setName
05          $this->name=$name;                     //设定成员变量，得到 name 属性值
06      }
07      function getName(){                        //声明方法 getName
08          return $this->name;                    //返回姓名
09          }
10      }
11                                                 //实例化一个类
12      $n=new Student ( );                        //实例化 Student 类
13      $n->setname("Li Lei");                     //设定
14      echo $n->getname();                        //调用方法 getname
15  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-1.php，查看其运行结果，即可看到如图 11-2 所示的结果。

图 11-2　定义方法

【代码解析】代码第 03 行将变量 name。第 04～06 行声明一个方法 setName()，用来设置变量 name 数据成员，成员 setName()传入参数 name，并将其设置给$this->name。第 07～09 行声明方法 getName()，此成员返回数据成员的值。第 12 行实例化一个类。第 13 行利用成员将参数“Li Lei”设置给数据成员$name。第 14 行利用 getName()方法取得$name 的值，并输出所得值。

注意：这里，方法内部调用本地属性时，使用$this->name来获取属性。在这个例子中，设置了公开的getName()方法，即用户只能获取$name，而无法改变它的值。这就是封装的好处 。

11.2　类的访问控制

在 PHP 5 中，限定字符指在 class 中声明的字段。在声明字段时，必须使用 public、private、

protected、final、const、static 之一进行修饰，用来描述对象的数据元素的作用域，所以这类字符称为限定访问控制字符。下面分别对这几个限定字符进行介绍。

11.2.1 公有属性

由关键字 public 声明的属性称为公有属性，可以自由地在类的内部、外部读取、修改，这样做显然不安全，破坏了其封装性的特点。

【范例 11-2】定义 Student 类，将变量 name，age 的值分别设为 PHP，20，用 public 声明其属性，实例化后，输出其名字和年龄。其程序如示例代码 11-2 所示。

示例代码 11-2

```
01  <?php                                          //PHP 开始标记
02      class Student                              //声明 Student 类
03      {
04      public $name="Li lei";                     //设定属性 name
05      public $age=20;                            //设定属性 age
06      }
07      $p=new Student();                          //实例化
08      echo "他的名字是：".$p->name."<br>";          //调用属性 name
09      echo "他的年龄是：".$p->age;                  //调用属性 age
10  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-2.php，查看其运行结果，即可看到如图 11-3 所示的结果。

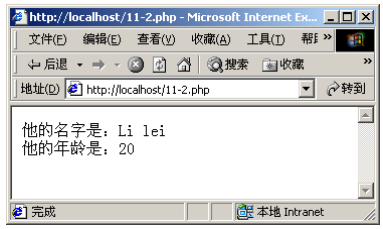

图 11-3 public 关键字

【代码解析】第 03、04 行设置了 Student 类的两个属性$name 和$age，在实例化后，使用$p->name 和$p->age 直接调用属性的值。

提示：如果字段没有被声明，则会默认为 public。

11.2.2 改变属性值

当属性声明为 public 时，在定义属性时可以不设置初始值，可以通过传递来给属性赋值，但如果开始定义属性时设置初始值，在外部调用时也可根据需要改变。

【范例 11-3】将示例代码 11-2 中的 Student 类实例化，输出名字为 Jim，年龄为 21，其程序如示例代码 11-3 所示。

示例代码 11-3

```
01  <?php                                          //PHP 开始标记
02      class Student                              //声明 Student 类
03      {
```

```
04      public $name="Li Lei";                          //设定属性 name
05      public $age=20;                                 //设定属性 age
06      }
07      $p=new Student();                               //实例化
08      $p->name=Jim;                                   //修改属性 name 值
09      $p->age=21;                                     //修改属性 age 值
10      echo "他的名字是：".$p->name."<br>";              //调用属性 name
11      echo "他的年龄是：".$p->age;                      //调用属性 age
12  ?>                                                  //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-3.php，查看其运行结果，即可看到如图 11-4 所示的结果。

图 11-4　改变属性值

【代码解析】程序 02～06 行声明了 Student 类，第 07 行实例化 Student 类。第 08 和第 09 行分别给属性$name 和$age 重新赋值，第 10～11 行输出属性值。

提示：一般情况下，如果想要使用类中的某一字段，可通过设置公共接口来访问。

11.2.3　私有属性

Private 修饰的属性在当前对象以外不能访问，设置私有属性是为了进行数据的隐藏。若字段被声明为 private（私有），则代表字段只能在该类中使用。在该类中可以通过$this->来调用，如果在外部引用私有属性，则将报错。

【范例 11-4】定义 Student 类，用 private 修饰其属性$name，实例化后，调用其属性。其程序如示例代码 11-4 所示。

示例代码 11-4

```
01  <?php                                               //PHP 开始标记
02      class Student                                   //声明 Student 类
03      {
04      private $name="Li Lei";                         //设定属性 name
05      }
06      $p=new Student();                               //实例化
07      echo "他的名字是：".$p->name."<br>";              //输出姓名
08  ?>                                                  //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-4.php，查看其运行结果，即可看到如图 11-5 所示的结果。

【代码解析】上述代码第 04 行设置$name 为私有属性，外部不能访问。指定为私有的字段不能由实例化的对象直接访问，即在外部$p->name 变得无效。

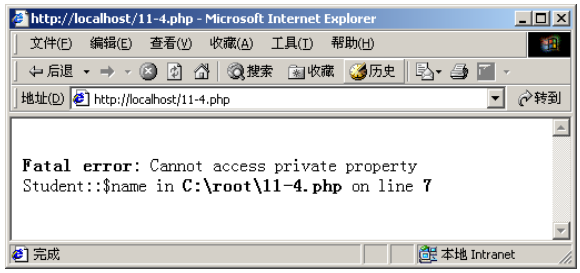

图 11-5　Private 关键字

提示：子类也不能使用被设置为私有属性的字段，如果要在子类中使用私有字段，可以考虑设置公共接口来访问，设置公共接口可以满足其封装性特点。

11.2.4　protected 属性

protected 限定的字段作用域在 public 和 Private 之间，若该成员被声明成 protected（保护），则代表在该类和该类的子类中能使用该字段。

【范例 11-5】设置一个 me 类，现有 100 元钱，设置为 protected 属性，有一个价格为 60 元的商品，客户给出 30 的价格，判断能否卖出此商品，最后并输出最终的钱数。其程序如示例代码 11-5 所示。

示例代码 11-5

```
<?php                                                    //PHP 开始标记
    class me{                                            //声明类 me
    protected $Money =100;                               //设定属性 Money
    protected $price1=60;                                //设定属性 price1
                                                         //卖东西的方法
    public function Sell($price){                        //声明方法 Sell
        if($this->price1<=$price){                       //判断是否能卖
            echo "好，卖给你了。<br>";                    //输出卖的信息
            $this->Money = $this->Money+$price;          //计算现在的钱
            return "我现在总共有 ".$this->Money." 元钱"; //输出现在的钱数
        }
    else{                                                //不能卖的情况
        echo "我不卖 ,$price 太便宜了<br>";              //输出不卖信息
        return "现在我还是 ".$this->Money." 元钱";       //输出现在的钱数
            }
        }
    }

    $now=new me;                                         //实例化
    echo $now->Sell(30);                                 //调用方法
?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-5.php，查看其运行结果，即可看到如图 11-6 所示的结果。

【代码解析】代码第 02 行声明 me 类，第 03、04 行将变量$Money 和 price1 设置为 protected 属性。第 06 行声明方法 Sell，第 07 行判断是否能卖。第 10、14 行调用私有属性 Money，第 19 行实例化 me 类。第 20 行调用方法 Sell 判断是否卖，由于出价低于 60，故不能卖。

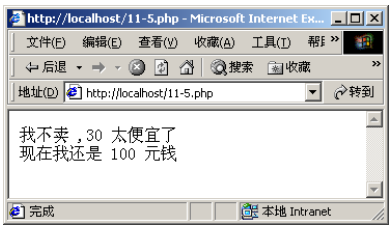

图 11-6　protected 关键字

11.2.5　const 属性

在 PHP5 类中，可以使用 const 定义一个常量，定义的这个常量不能被改变。const 定义的常量与定义变量的方法不同，const 定义的常量不需要加$修饰符。类中的常量使用起来类似于静态变量，不同点只是常量的值不能被改变。定义常量的结构形式为：

```
const 常量名称；//常量名称不能用$符号
```

如果字段被声明为 const（常量），其值将不能再改变。

【范例 11-6】定义 weight 类将 9.8 和 3.14159 设置为常量，分别赋给 G 和 PI，然后输出其值，其程序如示例代码 11-6 所示。

示例代码 11-6

```
01  <?                                                    //PHP 开始标记
02      Class weight                                      //声明 weight 类
03          {
04              const  G='9.8';                           //定义常量 G
05              const  PI='3.14159';                      //定义常量 PI
06      }
07                                                        //调用常量 9.8
08      echo "g=".weight::g."<br>";              //输出常量 G 的值
09      echo "pi=".weight::pi;                   //输出常量 PI 的值
10  ?>                                                    //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-6.php，查看其运行结果，即可看到如图 11-7 所示的结果。

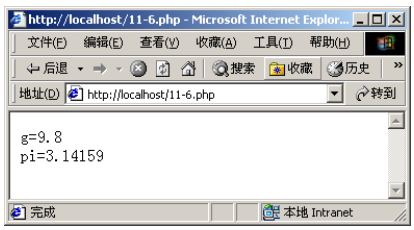

图 11-7　const 关键字

【代码解析】代码第 02～05 行定义 weight 类，第 04～05 行将变量声明为 const，第 08～09 行调用类，并输出常量值。

提示：程序员要养成一个良好的命名方式，使用 const 定义的常量名称一般都大写，这是一个约定，在任何语言中都是这样。如果定义的常量由多个单词组成，使用_连接，这也是约定。比如 MAX_MUMBER 这样的命名方式。

11.3 类的特性

面向对象语言的三大特色是：封装、继承、多态。封装即是将数据与操作此数据的方法包含在一起的特性。例如打开电视机收看电视节目，很少有人知道电视机里面的结构，电视机把一些零部件封装到机壳内来实现接收外部信号然后转化为视频信号的功能，在收看时不必去了解内部结构，直接通过按钮选出我们所需要的。这种用接口将用户与实际应用程序的内部工作原理分离的方法称为封装。由于其使用起来比较简单，这里不多做介绍。下面详细介绍其继承性和多态性。

11.3.1 继承性

继承性是面向对象程序设计设计语言的一个主要特点，继承就是实现对类的复用。简单地说，继承就如提到老师，教书育人是老师共同的特点，当提到物理老师时，其是物理课老师，也有教书育人的特性，即继承了老师教书育人的特性。通过“继承”一个现有的类，可以避免相似类的重复定义，提高编程效率。继承而产生的类叫做子类，被继承的类叫做父类。其结构形式为：

```
Class newclassname extends oldclassname
```

关键字“extends”表示继承关系，oldclassname 为被继承的类，newclassname 为要创建的子类。其特点如下所示。

- PHP 是单继承的，一个子类只可以继承一个父类，但一个父类却可以被多个子类所继承。
- 继承是从子类的角度看的，子类继承自父类，而派生是从父类角度看的，父类派生了子类。它们指的都是同一个动作，只是角度不同而已。
- 在 PHP5 中类的方法可以被继承。
- 子类不能继承父类的私有属性和私有方法。

【范例 11-7】设置一个 sell 类，现有 100 元钱，设置为 protected 属性，有一个价格为 60 元的商品，客户给出 110 的价格，判断能否卖出此商品，声明 now 类继承 sell 的属性和方法，输出最终的钱数。其程序如示例代码 11-7 所示。

示例代码 11-7

```
<?php                                                    //PHP 开始标记
    class sell{                                          //声明类 sell
        protected $Money =100;                           //设定属性 Money
        protected $price1=60;                            //设定属性 price1
        public function selled($price){                  //声明方法 selled
            if($this-> price1<=$price){                  //判断是否能卖
            echo "好，$price 元卖给你了。<br>";             //输出卖信息
            $this->Money = $this->Money+$price;          //计算销售后的钱数
            }
    else{                                                //不卖的情况
                echo "我不卖 ,$price 太便宜了<br>";        //输出不卖信息
            }
        }
    }
            //继承后的类可以方法父类的 protected 属性。
        class now extends sell {                         //声明 now 类
            public function getMoney(){                  //声明 getMoney 方法
```

```
18                 return $this->Money;                          //返回钱数
19                 }
20             }
21             $Me = new now ();                                 //实例化
22             $Me->selled(110);                                 //调用方法 sell
23         echo "现在我的钱数是:<br>".$Me->getMoney();          //输出现在钱数
24     ?>                                                        //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-7.php，查看其运行结果，即可看到如图 11-8 所示的结果。

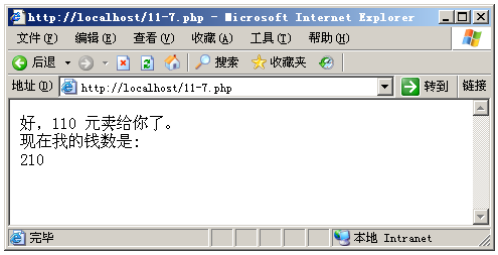

图 11-8　类的继承

【代码解析】代码第 03 行将变量$Money，设置为 protected 属性。第 05 行声明方法 selled，判断是否能卖，第 16 行定义一个 sell 类的子类 now，第 21 行实例化 now 类，第 22 行调用方法 selled，判断是否卖，第 23 行调用方法 getMoney，输出现在的钱数。

11.3.2　多态性

多态包括重载和重写，当从父类继承的方法不能满足子类的需求时，对父类进行改写的过程称为重写。当对父类的方法进行重写时，子类中的方法必须和父类中对应的方法具有相同的名称。子类中的方法不能使用比父类中被重写的方法更严格的访问权限。

【范例 11-8】定义一个名为 Child 的类，设置 getlegNum、speack、run 三个方法。其作用分别为“设置数据成员 legNum、输出“It can　speack Chinese”、输出“It running….”。实例化对象并输出结果，再定义一个 Mychild 的类，继承 Child 类的方法和属性，设置名字为“LiLei”。由于年龄太小，对其方法 speack 进行改写，使其分别输出“he is so small that he can speack nothing.”。实例化输出结果，其程序如示例代码 11-8 所示。

示例代码 11-8

```
01  <?php                                                  //PHP 开始标记
02      class Child{                                       //声明类 Child
03      protected $legNum =2;                              //设定属性 legNum
04      public function getlegNum (){                      //声明方法 getlegNum
05          return $this->legNum;                          //返回所得属性值
06      }
07      public function speack(){                          //声明方法 speack
08          return "It can  speack Chinese.<br>";          //返回“It can speack Chinese.”
09          }
10      public function run(){                                    //声明方法 run
11          return  "It running….<br>";                           //返回 “It running…”
12          }
13          }
14      $ch = new Child();                                        //实例化
15      echo "Child have ".$ch-> getlegNum ()."legs.<br>";      //输出 getlegNum 的值
```

```
16      echo $ch-> speack();                                    //调用方法 speack
17      echo $ch-> run();                                       //调用方法 run
18      class Mychild extends Child{                            //声明 Child 的子类
                                                                Mychild
19          private $name="Li L ei";                            //设定属性
20          public function getname(){                          //改写方法 getname
21              return $this->name;                             //设定成员变量
22              }
23          public function speack(){                           //改写方法 speack
24              return $this->name.",he is so small that he can speack nothing.";
25              }
26      }
27      $myChild=new Mychild();                                 //实例化
28      echo $myChild->getname()." have ".$myChild->getlegNum()." legs";
                                                                //调用方法
29      echo "<br>".$myChild->speack()."<br>";                  //调用方法 speack
30      echo $myChild->run();                                   //调用方法 run
31  ?>                                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-8.php，查看其运行结果，即可看到如图 11-9 所示的结果。

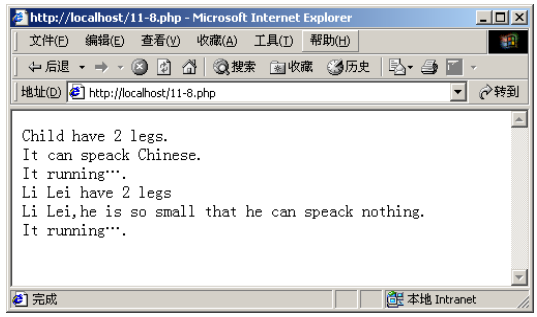

图 11-9　重写

【代码解析】代码第 02 行声明类 Child，第 04 行声明方法 getlegNum。第 07 行声明方法 speack，第 10 行声明方法 run。第 14 行实例化 Child 类，第 15～17 行分别调用方法 getlegNum、speack、run。第 18 行 Mychild 类继承 Child 类的属性和方法。第 23 行对方法 speak 进行重写，第 27 行实例化类 Mychild，然后再次调用方法。

警告：PHP 不支持重载。

11.4　构造函数

有时需要立即为实例化的对象的多个字段赋值，如果通过手工，会带来很多不可预测的问题，而如果在对象创建过程中自动执行，则会带来很多方便。构造函数就是当类被实例化时，会自动执行的函数，又称构造方法。下面就开始介绍构造函数。

11.4.1　创建构造函数

声明构造函数，同声明其他函数一样，只是构造函数的名称是固定的，即函数名称必须是__construct。PHP 的这种新的声明构造函数使用一个独一无二的名称，这样在改变类的名称时，

就不需要改变构造函数的名称。其一般结构如下：

```
class classname{
function __construct([argument1, argument2, argument2])
{
   /*class initialization code */
}
}
```

构造函数定义为对象实例化时自动执行的一段代码。

【范例 11-9】创建 user 类，设置属性 name、password、lastLogin，为了保密，将密码和用户登录时间设为私有属性。利用构造函数设置其成员变量，再利用方法 getLastLogin()获得最后访问时间，输出用户名和最后访问时间。其程序如示例代码 11-9 所示。

示例代码 11-9

```
<?php                                                    //PHP 开始标记
                                                         //定义一个跟踪用户的类
    class user {                                         //声明类 user
                                                         //属性
    public $name;                                        //设定属性 name
    private $password;                                   //设定属性 password
    private $lastLogin;                                  //设定属性 lastLogin
                                                         //方法
    public function __construct($name, $password) {     //声明构造函数
        $this->name = $name;                             //设定成员变量
        $this->password = $password;                     //设定成员变量
        $this->lastLogin = time();                       //设定成员变量
        }
                                                         // 获取最后访问的时间
    function getLastLogin() {                            //声明获得时间的方法
        return(date("M d,Y", $this->lastLogin));         //返回时间
        }
    }
                                                         //创建一个对象的实例
    $user = new user("Jun", "sdf123");                   //实例化
                                                         //获取最后访问的时间
    print("访问时间：".$user->getLastLogin() ."<br>");
                                                         //输出用户名
    echo  "   用户名：".$user->name."<br>";
?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-9.php，查看其运行结果，即可看到如图 11-10 所示的结果。

【代码解析】上述第 21 行创建一个对象实例 Leon，第 23 行在获得最后访问时间后自动执行构造函数__construct。通过$this->lastLogin = time()获取当前时间，访问次数也增加一次。

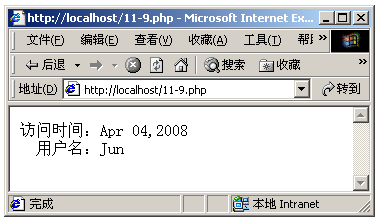

图 11-10　构造函数

提示：在以前的版本中构造函数的名称必须与类同名，在执行过程中，PHP 首先在类中搜索是否有__construct()函数，如果没有，则继续搜索是否有与类同名的函数。

11.4.2　调用父类构造函数

构造函数还能调用其他构造函数，但不会自动调用父类的构造函数，在调用时必须使用 parent 关键字调用。

【范例 11-10】在 student 类中定义构造函数，输出“I am a student.”，再定义 teacher 类继承 student 类的属性和方法。在 teacher 类中定义构造函数调用父类的构造函数，然后输出“I am a teacher.”。其程序如示例代码 11-10 所示。

示例代码 11-10

```
01  <?php                                          //PHP 开始标记
02      class student                              //声明 student 类
03      {
04          protected $name;                       //设定属性
05          protected $score;                      //设定属性
06      function __construct(){                    //声明构造函数
07          echo  "I am a student.<br>";          //输出“I am a student”
08          }
09      }
10      class teacher extends student{             //声明 student 的子类 teacher
11      function __construct()                     //声明构造函数
12      {
13          parent::__construct();                 //调用父类的构造函数
14          echo "I am a teacher.<br>" ;           //输出“I am a teacher.”
15      }
16      }
17      $peo=new teacher();                        //实例化
18  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-10.php，查看其运行结果，即可看到如图 11-11 所示的结果。

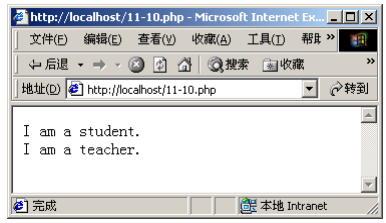

图 11-11　调用父类构造函数

【代码解析】代码第 17 行实例化了一个类，然后自动执行构造函数，parent::__contruct()调用 student 类中的构造函数。

提示：如果上述实例不出现 parent::__contruct()，则 student 类不会被调用，只能执行 teacher 函数。构造函数还可以调用类方法或其他函数。可以接受参数，能够在创建对象时赋给特定的对象字段。

11.5 析构函数

构造函数是在对象实例化时自动执行的函数。与构造函数相反，在对象被销毁时自动执行的函数称为析构函数。

11.5.1 析构函数的调用

默认情况下，PHP 仅释放对象属性所占用的内存，并不销毁对象相关的资源，而利用析构函数在使用一个对象之后执行代码来清除内存，并将对象从内存中销毁。析构函数__destruct()的结构形式如下：

```
class classname{
function __ destruct ()
{
   /*class initialization code */
}
}
```

析构函数是由系统自动调用的，不能带有参数。

【范例 11-11】声明 Counter 类，在类中定义析构函数，输出“析构函数被执行。”，实例化类后利用 do 循环输出小于 5 的整数的各数和，其程序如示例代码 11-11 所示。

示例代码 11-11

```
<?php                                          //PHP 开始标记
    class Counter {                            //声明 Counter 类
        function __destruct() {                //声明析构函数
        echo "<br>析构函数被执行。";             //输出“析构函数被执行。”
        }
        }
    $p=new Counter();                          //实例化
    $i=1;                                      //定义变量
    $sum=0;                                    //定义变量
    do{
        $sum=$sum+$i;                          //求和
        echo $sum.”<br>”;                      //输出结果
        $i++;                                  //变量值加 1
        }while($i<5)                           //判断是否结束
    echo "循环结束";                            //输出“循环结束”
    }
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-11.php，查看其运行结果，即可看到如图 11-12 所示的结果。

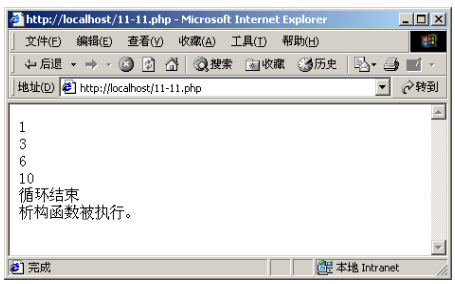

图 11-12　析构函数

【代码解析】代码第 02 行声明一个名为 Counter，第 03 行创建一个析构函数，第 07 行实例化类，第 08、09 行给变量赋值，第 10~14 行利用 do 循环输出小于 5 的整数的和。

11.5.2　使用其他方法调用析构函数

当 PHP 脚本不再与对象相关时，将调用析构函数。如果要明确地销毁一个对象，可以给指向该对象的变量不分配任何值。通常将变量赋值为 NULL 或者调用 unset。

【范例 11-12】在 Counter 类中构造析构函数，输出“析构函数被执行。”实例化后，消除变量，最后输出不大于 3 的整数，程序如示例代码 11-12 所示。

示例代码 11-12

```
01  <?php                                          //PHP 开始标记
02      class Counter {                            //声明类 Counter
03      function __destruct() {                    //声明析构函数
04          echo "<br>析构函数被执行。";            //输出“析构函数被执行”
05      }
06      }
07      $p=new Counter();                          //实例化
08      unset($p);                                 //销毁变量 p
09      for($i=0;$i<=3;$i++){                      //for 循环
10          echo "<br>".$i;                        //输出变量 i 的值
11      }
12  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-12.php，查看其运行结果，即可看到如图 11-13 所示的结果。

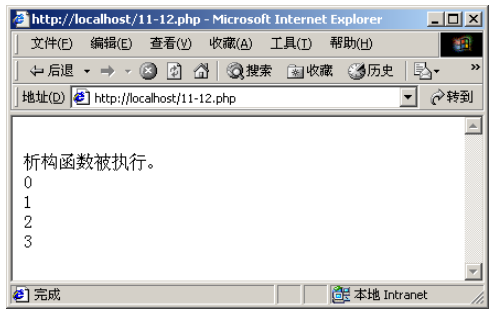

图 11-13　调用析构函数

【代码解析】代码第 03 行创建析构函数，第 07 行实例化。第 08 行调用 unset 将变量$p 清除，析构函数被执行。第 09～11 行执行 for 循环，输出变量 i 的值。

11.6　$this 关键字

PHP 5 中为解决变量的命名冲突和不确定性问题，引入关键字“$this”代表其所在当前对象。

- $this 在构造函数中指该构造函数所创建的新对象。
- 在类中使用当前对象的属性和方法，必须使用$this->取值。
- 方法内的局部变量不属于对象，不使用$this 关键字取值。

11.6.1 调用变量

调用当前对象的属性值必须使用$this 关键字，通过符号“->”来调用。下面通过实例看通过$this 关键字调用的私有属性值。

【范例 11-13】创建一个 price 类，反映价格的变化，原来的价格为 88，目前的价格为 108，通过调用变量输出目前的商品价格和原来的价格，程序如示例代码 11-13 所示。

示例代码 11-13

```
01  <?                                          //PHP 开始标记
02      class price{                            //声明类 price
03      private $price = 88;                    //设定属性
04      public function printprice($price){     //声明一个打印参数的方法
05          echo "现在的价格是 $price 元。";     //输出传递的参数值
06          echo "<br>";                        //换行
07          echo "原来的价格是 $this->price 元。";//输出属性 a 的值
08          }
09      }
10      $a = new price();                       //这里的$a 可不是类中的任何一个变量了
11      $a->printprice(108);                    //调用方法 printInt
12  ?>                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-13.php，查看其运行结果，即可看到如图 11-14 所示的结果。

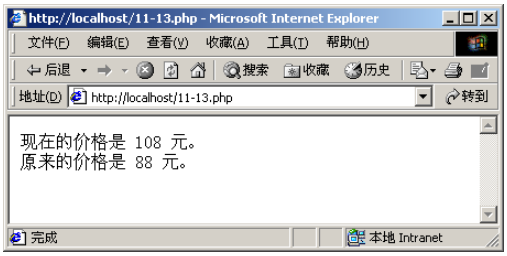

图 11-14 调用变量

【代码解析】代码第 02 行声明 price 类，第 03 行设定 price 为私有属性，第 04 行声明方法 printprice，第 05 行输出传递参数的值，第 07 行使用$this 调用当前对象的属性。第 10 行实例化，第 11 行调用方法 printprice，输出结果。

注意：局部变量和属性可以同名，但用法不一样。在使用中，要尽量避免这样使用，以免混淆。

11.6.2 调用其他方法

通过$this 不但可以调用变量，而且可以直接调用类中的函数。下面通过实例来查看$this 调用函数的方法。

【范例 11-14】声明类 Comp，设置方法 sum 计算两变量之和。定义方法 prt，计算两变量之积。再定义方法 product，通过调用前两个方法计算（a+b）*c 的值。程序如示例代码 11-14 所示。

示例代码 11-14

```
01  <?                                                          //PHP 开始标记
02      class comp{                                             //声明类 comp
03          public function sum($a,$b){                         //定义方法 sum
04          return $a+$b;                                       //返回和
05      }
06          public function prt($a,$c){                         //定义方法 prt
07          return $a*$c;                                       //返回乘积
08      }
09      public function product($a,$b,$c){                      //声明方法 product
10      $a = $this->sum($a,$b);                                 //调用方法 sum
11      return $this->prt($a,$c);                               //调用方法 prt
12          }
13      }
14      $math = new comp();                                     //实例化
15      echo "(5+10)*15 的结果为".$math->product(5,10,15);      //返回结果
16  ?>                                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-14.php，查看其运行结果，即可看到如图 11-15 所示的结果。

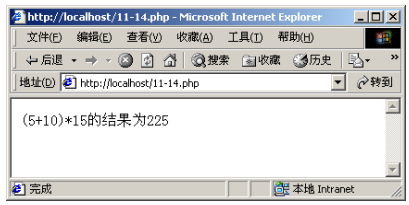

图 11-15　调用方法

【代码解析】代码第 02 行声明类 Comp，第 03 行定义方法 Sum。第 10~11 行通过$this 调用以前定义的 sum 方法。第 15 行输出结果。

11.7　static 关键字

前面讲过静态变量，同样，属性和方法也能声明为静态，称为静态属性和静态方法。static 关键字声明的属性或方法是和类相关的，而不是和类的某个特定的实例相关。因此，这类属性或方法也称为“类属性”或“类方法”。

11.7.1　静态属性

一个类的所有实例，共用类中的静态属性，也就是说，在内存中即使有多个实例，静态的属性也只有一份。访问该类时，可不必实例化，而直接使用属性名加两个冒号“::”即直接访问类中 static 的属性。

【范例 11-15】将 count 和 sum 定义为两个静态属性，利用其计算 10 的阶乘，程序如示例代码 11-15 所示。

示例代码 11-15

```
01  <?                                                          //PHP 开始标记
02      class user{                                             //定义类
03      private static $count = 10;                             //定义静态变量
04      private static $sum=1;                                  //定义静态变量
```

```
05      public function __construct(){                          //定义方法
06      for($i=1;$i<11;$i++){                                   //利用 for 循环
07          self::$sum = self::$sum * self::$count;             //调用静态变量
08          self::$count--;                                     //调用静态变量
09      }
10      }
11      public function getCount(){                             //定义方法 getCount
12          return self::$sum;                                  //返回计算结果
13      }
14      }
15      $user1 = new user();                                    //实例化
16      echo "10 的阶乘为：".$user1->getCount();                 //调用方法
17  ?>                                                          //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-15.php，查看其运行结果，即可看到如图 11-16 所示的结果。

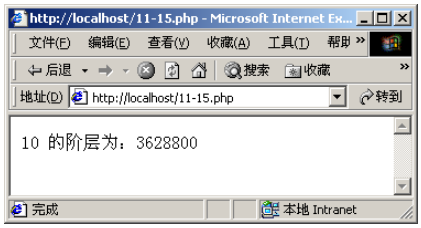

图 11-16　静态属性

【代码解析】代码第 03、04 行设置了由 private 和 static 共同修饰的$count 和$sum 属性。第 05 行建立构造函数，第 06~09 行利用 for 循环调用静态变量，计算 10 的阶乘。第 11~13 行定义方法 getCount 返回计算结果。第 15 行实例化对象，第 16 行调用方法 getCount，输出计算结果。

提示：static 的属性和方法可以被访问，但不能访问非静态的属性和方法。因为静态属性和方法被创建时，可能还没有任何这个类的实例可以调用。

11.7.2　静态方法

如果将方法声明为静态，则在类中不能用 this 来引用该方法，可以使用下面两种方法引用：

在类中：self::静态成员名称。
在类外：类名称::静态成员名称。

【范例 11-16】定义类 Math，声明带参数 num1、num2 的静态方法 sum 和 product，分别计算两数之和和两数之积。调用静态方法，分别计算 88 和 100 两数之和，66 和 88 两数之积。其程序如示例代码 11-16 所示。

示例代码 11-16

```
01  <?                                                          //PHP 开始标记
02      class Math{                                             //声明类 Math
03          public static function sum($num1,$num2){           //声明方法 Max
04          return $num1 + $num2;                               //返回较大值
05          }
06          public static function product($num1,$num2){       //声明方法 Max
07          return $num1 * $num2;                               //返回较大值
08      }
09          }
```

```
10      $a = 88;                                    //定义变量 a
11      $b = 100;                                   //定义变量 b
12      echo "两数之和为：";                        //输出变量值
13      echo "<br>";                                //换行
14      echo Math::sum($a,$b);                      //调用静态方法
15      echo "<P>";                                 //换行
16      $a = 66;                                    //定义变量 a
17      $b = 88;                                    //定义变量 b
18      echo "两数之积为：";                        //输出变量值
19      echo "<br>";                                //换行
20      echo Math::product($a,$b);                  //调用静态方法
21  ?>                                              //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-16.php，查看其运行结果，即可看到如图 11-17 所示的结果。

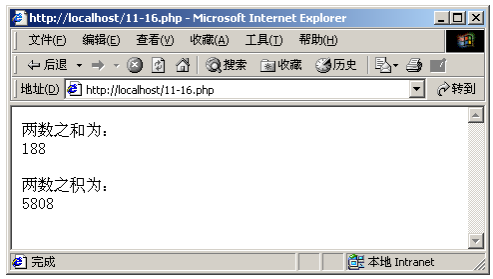

图 11-17　静态方法

【代码解析】代码第 02 行定义类 Math，第 03~05 行定义静态方法 sum，计算两数之和。第 06~08 行定义静态方法 product，计算两数之积。第 14、20 行调用静态方法，分别计算 88 和 100 的和，66 和 88 的积。

11.8　final 关键字

PHP 5 引入“final”关键字，当在一个函数声明前使用 final 关键字时，这个函数将不能被任何子类重载。

11.8.1　final 函数的调用

final 函数是指在 function 前使用 final 关键字修饰的函数，调用 final 函数的方法与调用一般的方法相同。一般是实例化后，直接使用“->”符号即可实现对 final 函数的调用。

【范例 11-17】声明 circle 类，通过声明 area 方法计算半径为 10 的圆，并将方法定义为 final 函数。实例化后调用此方法，其程序如示例代码 11-17 所示。

示例代码 11-17

```
01  <?php                                          //PHP 开始标记
02      class circle{                               //声明类 circle
03      public $radius=10;                          //设定属性 radius
04      const pi=3.14159;                           //设定常量 pi
05      final function area(){                      //声明方法
06          echo "半径=",$this->radius;             //输出半径大小
07          echo "<br>面积=";                       //输出“面积”
```

```
08          echo self:: pi*$this->radius*$this->radius;     //输出面积大小
09          }
10          }
11                                                          //实例化一个类
12      $cir=new circle();                                  //实例化
13      $cir->area();                                       //调用方法
14  ?>                                                      //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-17.php，查看其运行结果，即可看到如图 11-18 所示的结果。

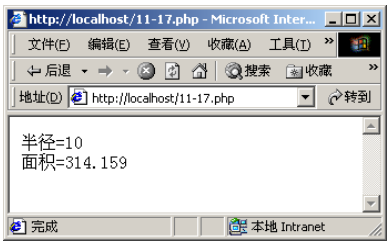

图 11-18　final 函数调用

【代码解析】代码第 05 行用 final 定义了一个终结继承函数，其不能再被重写。pi 是一个被声明为一个常数值的字段，self::pi 对它的调用。$this 直接调用了字段 radius。$cir=new circle()实例化 circle 类；$cir->area()是对函数 area()的调用。

提示：final 函数不影响外部的调用，但会影响其被改写，这一点在后面将会讲述。

11.8.2　final 类不能被继承

final 类不但能修饰方法，也能修饰类。被 final 修饰的类称为 final 类，被 final 修饰的类不能被继承，如果继承将会出现错误。

【范例 11-18】设定的 Math 为 final 类，声明一个 area 方法，计算圆的面积。定义 circle 类并继承 Math 类的属性和方法，改写 area 方法，计算圆的周长，其程序如示例代码 11-18 所示。

示例代码 11-18

```
01  <?                                                  //PHP 开始标记
02                                                      //声明一个 final 类 Math
03      final class Math{                               //声明类 Math
04      public static $pi = 3.14;                       //设定静态属性 pi
05      public function area($radius){                  //声明方法
06          return self:: pi*$radius*$radius;               //返回计算结果
07          }
08          }
09      //声明 circle 继承自 Math 类.
10      class circle extends Math{                          //声明子类 circle
11          function area($r){                              //改写方法
12              return 2*3.14 * $r;                         //返回周长
13          }
14      }
15      $cir=new circle();                                  //实例化
16      echo $cir->area(3);                                 //调用方法
17  ?>                                                      //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-18.php，查看其运行结果，即可看到如图 11-19 所示的结果。

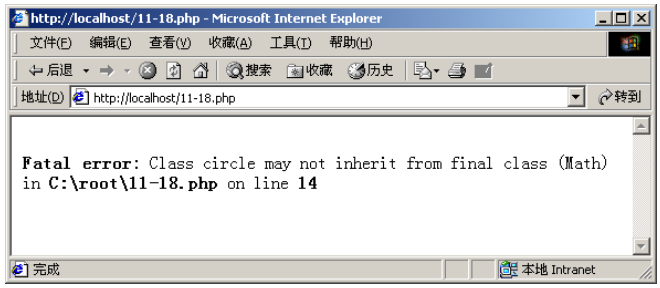

图 11-19　继承 final 函数

【代码解析】代码第 03 行声明一个 final 类 Math，第 05~07 行声明一个方法 area，计算圆的面积，第 10 行声明类 circle 并继承 Math 类的方法和属性，第 11~13 行改写方法 area。第 15 行实例化，第 16 行调用方法 area。因为类 Math 定义为 final 类，不能被继承，故程序报错。

11.8.3　final 方法不能被重写

如果不希望类中的某个方法被子类重写，只需要在这个方法前加上 final 修饰符，即设置为 final 方法。

【范例 11-19】创建 Math 类，声明方法 String 和 sum，利用方法 String 返回“这是 Math 类”，利用方法 sum 结算两数之和，并将方法 sum 设为 final 方法，再创建 SuperMath 类，继承 Math 类，并重写方法 sum。程序如示例代码 11-19 所示。

示例代码 11-19

```
01  <?                                          //PHP 开始标记
02                                              //声明一个 final 类 Math
03      class Math{                             //定义类 Math
04          const pi=3.14159;                   //设定属性
05          public function string(){           //设置方法 string
06          return "这是 Math 类.";             //返回“这是 Math 类”
07          }
08          final function sum($a,$b){          //定义方法 sum
09              return $a + $b;                 //返回两数之和
10          }
11      }
12      //声明 SuperMath 继承自 Math 类.
13      class SuperMath extends Math{           //定义 Math 的子类 SuperMath
14          public  function sum($a,$b){;           //改写方法 sum
15          return $a * $b;                         //返回两数之积
16          }
17      }
18      $cir=new SuperMath();                       //实例化
19      echo $cir->sum(4,5)                         //调用方法 sum
20  ?>                                              //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-19.php，查看其运行结果，即可看到如图 11-20 所示的结果。

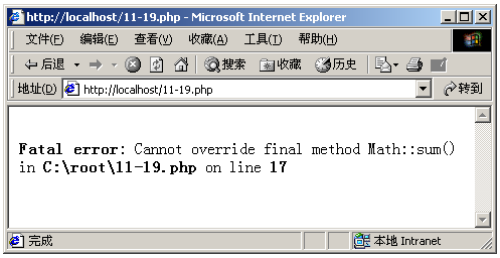

图 11-20　final 方法

【代码解析】代码第 02 行声明 Math 类，第 05~07 行声明了方法 String。第 08~10 行定义一个 sum 方法，计算变量 a 和 b 的和。第 13 行创建 SuperMath 类并继承 Math 类的方法和属性。第 14 行重写 max 方法，程序出错。

警告：如果 final 方法被重写，将会发生错误。

11.9　综合练习

1．建立一个类，实现输出九九乘法表，再利用这个类，输出多个维数的乘法表。

提示：先建立一个名为 Text 的类，定义带参数的方法 print_Text，利用双重循环，即可输出变量一个单边乘法表，这样 Text 类就编写完成了；实例化对象$Text 为 Text 类；对象$Text 调用其函数 print_Text，这样打印出了九九乘法表；实例化对象$Text，实现多次输出。

其程序如示例代码 11-20 所示。

示例代码 11-20

```
<?php                                          //PHP 开始标记
    class  Text{                               //声明类 Text
        function   print_Text($x)                  //声明方法 print_Text
        {
            for($i=1; $i<=$x;$i++){                //循环 i
            for($j=1;$j<=$i;$j++){                     //循环 j
            echo $j."x".$i."=";                    //输出公式
            echo $i*$j." ";                    //输出两数的积
    }
    echo  "<br>";                                 //换行
        }
    }
    }
        $Textl=new Text;                           //实例化
        $Textl->print_Text(9);                     //调用方法 print_Text
        echo  "<br>";                             //换行
        $Text2=new Text;                           //实例化
        $Text2->print_Text(6);                     //调用方法 print_Text
        echo  "<br>";                             //换行
        $Text3=new Text;                           //实例化
        $Text3->print_Text(3);                     //调用方法 print_Text
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-20.php，查看其运行结果，即可看到如图 11-21 所示的结果。

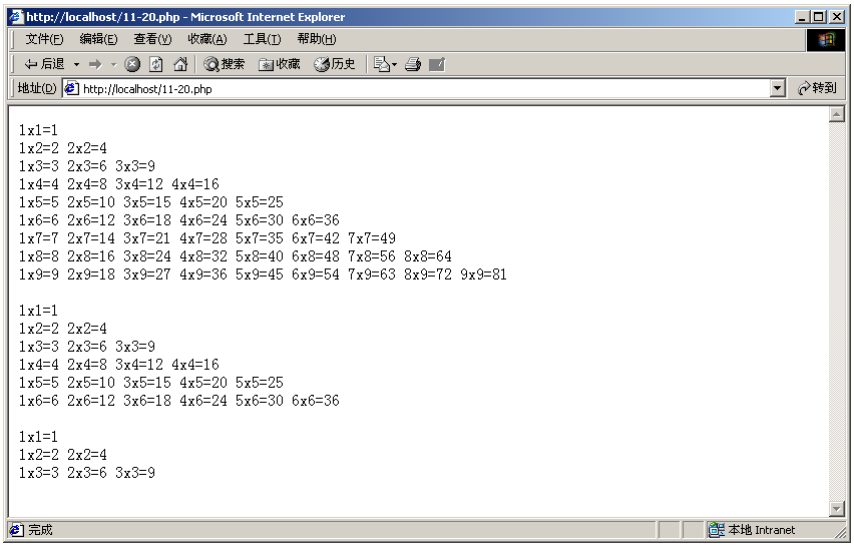

图 11-21　乘法表

2．制作一个记录用户注册的页面，利用构造函数和析构函数执行“++、--”运算。

提示：利用构造函数制动执行“++”运算，利用析构函数执行“--”运算。其程序如示例代码 11-21 所示。

示例代码 11-21

```
<?php                                          // PHP 开始标记
    class Counter {                            //声明类 Counter
        private static $count = 0;             //设定静态属性

    function __construct() {                   //构造函数
        self:: $count++ ;                      //调用静态变量
    }

    function __destruct() {                    //析构函数
        self:: $count--;                       //调用静态变量
    }
    function getCount() {                      //声明方法 getCount
        return self:: $count;                  //调用静态变量
    }
    }

    $c = new Counter;                          //实例
    print($c->getCount()."<br>");              //输出 1
    $c2 = new Counter();                       //建立第二个实例
    print($c->getCount() . "<br>");            //输出 2
    $c2 = NULL;                                //销毁实例
    print($c->getCount()."<br>");              //输出 1
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/11-21.php，查看其运行结果，即可看到如图 11-22 所示的结果。

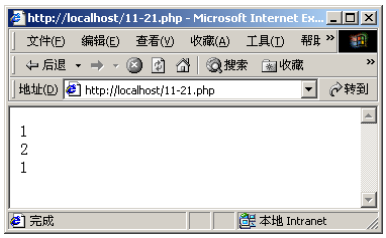

图 11-22　构造与析构函数

11.10　小结

本章主要学习了类的创建和常见类的使用，其中很多问题是编程时常会遇到的。利用类把复杂的问题分成一个个模块，可以使问题简单，也可以避免程序的冗杂。如果读者对本章内容还有疑问，可参考《精通 PHP5 应用开发》（秦涛，曾文玉．北京：人民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民．北京：中国水利水电出版社，2003）。

11.11　习题

一、填空题

1．类是一组具有相同属性和行为的对象的抽象，是_____、_____的定义。

2．面向对象的三大语言特点是：_____、_____、_____。

3．关键字_____声明的字段可以被该类和该类的子类访问，关键字_____声明的字段可以被外部直接访问。

4．继承，一个_____只可以继承一个_____，但一个_____却可以被多个_____所继承，子类不能继承父类的_____和_____。

5．如果从父类继承的方法不能满足子类的需求，可以对其进行改写，这个过程叫方法的_____。

6．调用父类的构造函数时，必须使用关键字_____调用。

7．析构函数的作用是_____。

8．在类中使用当前对象的属性和方法时，必须使用_____取值。

二、选择题

1．使用哪个指令可以实例化类（　　）。

A．class　　B．new　　C．create　　D．method

2．继承性是面向对象程序设计设计语言的一个主要特点，表示继承关系的关键字是（　　）。

A．class　　B．new　　C．extends　　D．static

3．下列哪个是构造函数名（　　）。

A．__destruct()　　B．_construct()　　C．不确定　　D．以上都不是

4．final 关键字修饰的类（　　）。

A．可以被继承　　B．可以被覆盖

C．即能被继承又能被覆盖　　D．既不能被继承又不能被覆盖

5．选择下面程序的运行结果（　　）。

```
01  class student
02  {
03   function __construct(){
04   echo  "I am a student.<br>";
05       }
06
07   function teacher() {
08   echo "I am a teacher.";
09           }
10           }
11
12  $peo=new student();
```

A．I am a student.
　　I am a teacher.

B．I am a student.

C．I am a teacher.

D．无任何输出

三、简答题

1．简述类和对象的定义。

2．阐述限定字符 public、private、protected、final、const、static 的作用。

3．解释构造函数和析构函数的差异。

四、编程题

1．创建一个 oblong 类，编写一个方法计算长方形的周长。

2．创建一个购物类，创建物品名称、价格，判断是否有足够的资金购买，如果买，输出物品名称及价格，如果不买输出原因。

第 12 章　会 话 管 理

本章主要讨论 Web 应用程序开发中的一个重要内容：会话管理。会话管理包括 Session 和 Cookie 两种技术，两者既有区别又有相同之处。其主要的功能都是把用户端与服务器有机地关联起来，以便能够有效管理和查看用户在网站中的状态。

学习本章，可以获得以下知识点：

- Cookie 概述；
- Cookies 的使用；
- Session 的使用；
- Session 函数的使用。

12.1　Cookie 概述

Cookie 可以用来存储用户的信息，如存储用户名、密码、用户访问该站点的次数等。在访问某站点时，Cookie 将 HTML 网页发送到浏览器中的一小段信息，以脚本的形式保存在客户端计算机上。

12.1.1 Cookie 的作用

延续前一个话题，如果服务器想要记录各用户上站的次数，使用文件存储数据，那么数据量会非常大，对服务器来说负担会很重。因此，可以将数据存入用户计算机中，如此一来服务器就不需要记录大量的数据，也不用识别不同的用户，即将个人上站信息存储在个人计算机上。在网页系统上，常用 Cookie 来记录会员的 ID 和密码，其记录和运用的结构示意图如图 12-1 所示。

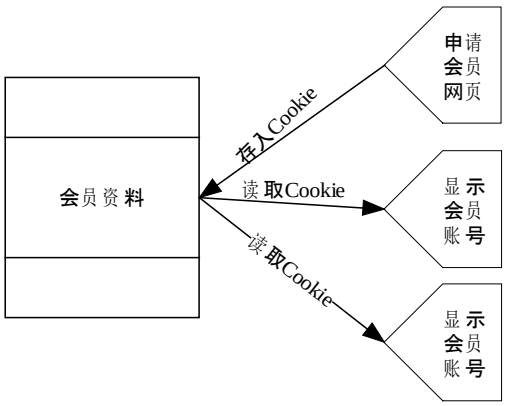

图 12-1　存储和读取 Cookie

一般来说，Cookie 通过 HTTP Headers 从服务器端返回浏览器。首先，服务器端在响应中利用 Set Cookie Header 来创建一个 Cookie。然后浏览器在请求中通过 Cookie Header 包含这个已经创建的 Cookie，并且将它返回至服务器，从而完成浏览器的论证。

12.1.2 Cookie 的限制

因为 Cookie 常常用来存储用户的重要信息，为了防止乱用导致用户的信息泄漏，所以对 Cookie 做出以下限制。

- 浏览器记录 Cookie 的容量不得大于 4KB。
- 每个浏览器只能保存某个服务器上的 20 个 Cookie。如果试图保存更多的 Cookie，则最先保存的 Cookie 就会被删除。
- 每个用户的浏览器最多只能访问 300 个 Cookie。

由于 Cookie 保存着重要信息，出于安全考虑，用户可以拒绝服务器存储这些数据。因此，要使用 Cookie 存储信息，必须先确认浏览器中 Cookie 的功能是否已经打开。打开浏览器“属性”|“安全”|“自定义级别”选项，即可看到如图 12-2 所示的内容，选中启用即可。

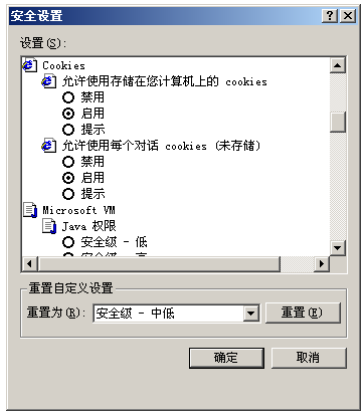

图 12-2　Cookie 的限制

说明：浏览器创建了一个 Cookie 后，对于每一个针对该网站的请求都会在 Header 中带着这个 Cookie。而且浏览器会这样一直发送，直到 Cookie 过期为止。不过对于其他网站的请求，Cookie 是绝对不会跟着发送的。

12.2　PHP 中 Cookie 的使用

Cookie 是一个允许在用户的系统上存储数据的 HTTP 工具。Cookie 对于存储用户信息，以及在用户转移到新的浏览器页面时保留其他信息是非常有用的。

12.2.1 创建 Cookie

确认 Cookie 的权限后，接着要创建 Cookie。创建 Cookie 需要调用 setcookie()函数，其格式如下：

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

name 参数指定 Cookie 的名称，其他参数为可选参数。value 参数是存入 Cookie 的值，此值保存在客户端，不能用来保存敏感数据。expiration 参数指定 Cookie 的过期时间，过了这个时间，将无法访问 Cookie。path 为 Cookie 在服务器上的有效路径，用来指定 Cookie 将被发送

到服务器的哪一个路径下。secure 指明 Cookie 是否通过安全的 HTTPS 链接传送，当设为 true 时，Cookie 仅在安全的链接中被设置。

创建名为 A、B 的 Cookie 方法为：

```
01  <?php
02      setcookie("A","2000");
03      setcookie("B","I am a student.");
04  ?>
```

代码第 02 行创建名为 A 的 Cookie 的值为“2000”，第 03 行设置名为 B 的 Cookie 值为“I am a student.”。

注意：Cookie 不会在设置它的本页内生效，要测试一个 Cookie 是否被成功设定，可以在设置的 Cookie 到期之前通过另外一个页面来访问其值。

12.2.2 访问 Cookie

大多数变量都会在包含它们的 PHP 脚本终止时消失。与此相反，Cookie 是可以无限期保留其值的变量，因此为了保留它们的值，用户的浏览器必须在用户的本地硬盘驱动器上存储 Cookie。

【范例 12-1】创建 A，B 的值后，通过 Cookie 访问变量 A、B 的值。其程序如示例代码 12-1 所示。

示例代码 12-1

```
01  <?php                                          //PHP 开始标记
02      echo "A= ".$_COOKIE["A"]."<br>";           //输出“A=2000”
03      echo "B= ".$_COOKIE["B"]."<br>";           //输出 “B=I am student.”
04      echo "取出 Cookie 值";                      //输出“取出 Cookie 值”
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/12-1.php，查看其运行结果，即可看到结果如图 12-3 所示。

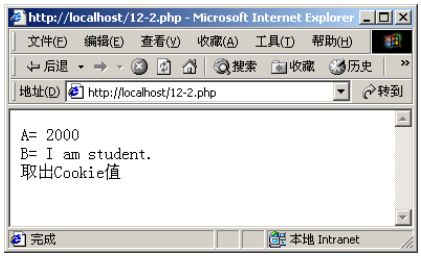

图 12-3　Cookie 访问

【代码解析】代码第 02 行输出名为 test 的 Cookie 值。第 03 行按数组的形式输出 Cookie 值。

注意：从上述结果可以看出，实际上 Cookie 已经跟踪了用户的状态，即能够在第二个页面得到用户在第一个页面留下的状态信息。

12.2.3 Cookie 工作时间

重新打开一个浏览器，然后执行范例 12-1，此时会发现 Cookie 不见了，这是因为没有设

置有效的 Cookie 工作时间所致。如果要保留或浏览器之间共用 Cookie，就必须设置有效时间，其设置方法如下：

```
time()+秒数
```

time()函数取得目前系统的时间标签，后面的秒数即是 Cookie 的有效时间。如果要指定某个日期，可以使用 mktime()，其结构形式为：

```
mktime（时，分，秒，月，日，年）
```

Cookie 在到达过期时间后将被自动删除。不过也可以立即删除一个 Cookie。要想实现这一点，将 Cookie 的过期时间设置为过去的一个时间即可。

创建 A、B、C 的 Cookie 值都为“10”，A 的失效时间为一小时之前，B 的失效时间为一小时后，C 的失效时间为 2010 年 1 月 1 日前。

```
01  setcookie("A", “10”, time() -3600);
02  setcookie("B", “10”, time() +3600);
03  setcookie("C","10", mktime(0,0,0,1,1,2010);
```

代码第 01 行将 CookieA 的过期时间设置为一小时之前，其值已经被删除。第 02 行将 CookieB 的过期时间设置为一小时之后。第 03 行设定 CookieC 的失效时间为 2010 年 1 月 1 日前。

【范例 12-2】制作一个网页计数器，用 Cookie 记录在 5 分钟以内浏览页面的次数，其程序如示例代码 12-2 所示。

示例代码 12-2

```
01  <?php                                                    // PHP 开始标记
02      $count=$_COOKIE[nam];                                //设定变量 nam
03      $count++;                                            //变量加 1
04      setcookie("nam",$count,time()+300);                  //设定时间为 300 秒
05      echo "欢迎光临! <br>你是第 $count 位光临本站者";      //输出变量 count 的值
06  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/12-2.php，查看其运行结果，即可看到结果如图 12-4 所示。

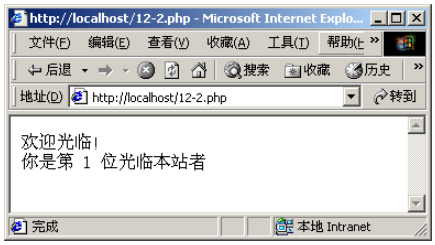

图 12-4　访问次数

【代码解析】代码第 02 行设置名为 name 的 Cookie 并指定给变量$counter。第 03 行将变量值加 1，表示浏览数增加 1 次。第 04 行设定有效访问时间，有效时间为 5 分钟。第 05 行输出访问人数。

注意：当重新登录页面时，浏览次数加 1。当过了设置的时间 5 分钟后，则 Cookie 停止更新，Cookie 消失，重新从 1 开始记录。

12.2.4 Cookie 数组

PHP 可以将 Cookie 各个变量联系起来，组成数组。下面通过实例介绍如何创建一个 Cookie 数组。

【范例 12-3】通过 setcookie 设定三个 Cookie 变量 A、B、C，其值分别为“2008”、“I am a student.”、“I am a teacher.”，然后以数组形式输出。其程序如示例代码 12-3 所示。

示例代码 12-3

```
<?php
    setcookie("cookie[A]","2008");                          //设定变量
    setcookie("cookie[B]","I am a student.");               //设定变量
    setcookie("cookie[C]","I am a teacher.");               //设定变量
    if(isset($_COOKIE['cookie'])){                          //判断是否为数组
      foreach($_COOKIE['cookie'] as $name=>$value){         //遍历数组
        echo "$name:$value<br>";                            //输出键和键值
      }
    }
    else                                                    //不是数组的情况
        echo "不是数组."                                    //输出“不是数组”
?>
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/12-3.php，查看其运行结果，即可看到结果如图 12-5 所示。

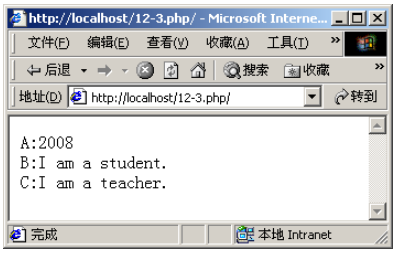

图 12-5 Cookie 数组

【代码解析】代码第 02～04 行分别设定三个 Cookie 变量。第 05 行判断是否为数组。第 06 行遍历数组。第 07 行输出其键和键值。第 11 行输出不是数组的情况。

12.3 Session 常见函数及用法

Session 是指有始有终的一系列动作或信息。比如吃饭从坐到桌前到吃完饭中间的一系列过程可以称之为一个 Session。Session 将不定量的变量存储在服务器端，而 Cookie 将数据存储在客户端计算机上。Session 应用比较广泛，下面介绍一下 PHP 中 Session 的应用。

12.3.1 启动一个 Session

Session 将信息存储在服务器端，每一个链接进入网站服务器后便会产生一个 Session。所以第一步就是告诉服务器要使用 Session 的功能来存储数据。使用 Session 功能的方式有两种：第一种是使用 Session_start()函数启动会话，第二种是利用 php.ini 设置文件。

第一种方式启动一个会话，其结构形式如下：

```
Session_start()
```

其开始一个会话或者返回已经存在的会话，函数没有参数，且返回值均为 True。使用此函数时必须在任何输出前调用此函数，否则会显示如下错误：

```
Warning: Cannot send session cache limiter - headers already sent (output started at /usr/local/apache/htdocs/cga/member/1.php:2)……
```

以第二种方式启动会话，可以在 php.ini 里设置参数 session.auto_start。将此参数设为 1，表示 enable，即当连接到服务器后 Session 的功能将自动开启。

启动 Session 之后，在使用 Session 变量之前首先要注册变量，之后才能使用，反之不使用某个变量时就要取消注册。Session_ragister（变量名）可以注册一个 Session 变量；Session_unregister（变量名称）可以取消注册一个 Session 变量，例如：

```
01  session_start();
02  session_register("user");
03  session_unregister("user");
```

第 01 行启动 Session 变量，第 02 行注册 user 变量，第 03 行注销 user 变量。

12.3.2 调用 Session 变量

PHP 5 可以直接使用$_SESSION 函数创建函数变量，其格式如下：

```
$_SESSION[$sting]=$str;
```

参数$sting 为要定义的变量名，$str 为要定义的变量值。注册的变量为全局变量，然后是可以使用$_SESSION[$sting]调用变量，其使用方法和 get、post、cookie 相似。

【范例 12-4】创建 Session 变量 count 和 user，其值设为“2008”和“Hello”。然后调用该变量，使之输出“2009”和“Hello My friend”。其程序如示例代码 12-4 所示。

示例代码 12-4

```
01  <?php                                                  //PHP 开始标记
02      session_start();                                   //启用 session
03      $_SESSION['count']=2008;                           //定义变量
04      $_SESSION['user']="Hello ";                        //定义变量
05  ?>                                                     <!--PHP 结束标记-->
06  <html>                                                 <!--html 标签开始-->
07      <title>调用 Session</title>                        //显示标题
08      <body>                                             <!-- body 标签开始-->
09  <?php                                                  // PHP 开始标记
10      echo "访问前 <br>";                                //输出“访问前”
11      echo "count= ".$_SESSION['count']."<BR>";          //输出变量值
12      echo "user= ".$_SESSION['user']."<P>";             //输出变量值
13      $_SESSION['count']++;                              //变量值加 1
14      $_SESSION['user'].="My friend.";                   //连接字符串
15      echo "访问后<br>";                                 //输出“访问后”
16      echo "count= ".$_SESSION['count']."<BR>";          //输出变量值
17      echo "user= ".$_SESSION['user']."<BR>";            //输出变量值
18  ?>                                                     <!--PHP 结束标记-->
19  </BODY>                                                <!-- body 标签结束-->
20  </HTML>                                                <!-- html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/12-4.php，查看其运行结果，即可看到结果如图 12-6 所示。

【代码解析】程序第 02 行启用 Session，第 03、04 行创建 Session 变量 count 和 user。第 11、12 行调用 Session 变量，第 13 行使变量 count 的值加 1。第 04 行连接字符串“My friend.”

第 16、17 行输出变量 count 和 user 的值。

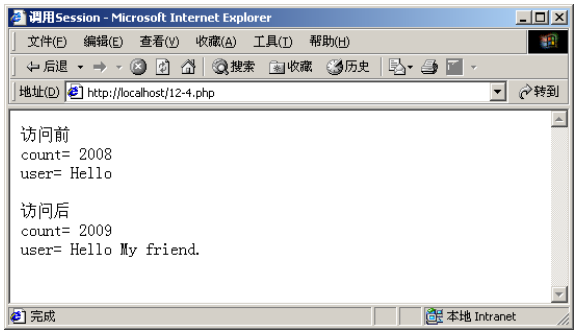

图 12-6　访问次数

12.3.3　查看 session_id 的值

服务器利用不同的 session_id 来区别不同的 Session，用户可以通过 session_id()函数查看其本身使用的 session_id，其结构形式如下：

```
session_id()
```

函数 session_id()取得当前 session_id 值，也可以用其设定 session_id 值，使用该函数前必须启动会话，如果当前会话没有启动，则返回空字符串。

【范例 12-5】通过 session_id()函数，查看当前网页的 session_id 值，其程序如示例代码 12-5 所示。

示例代码 12-5

```
01  <?php                                    //PHP 开始标记
02      session_start();                     //启用变量
03      $sessionID= session_id();            //取得 sessionid
04      echo "ID=$sessionID ";               //输出 ID
05  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/12-5.php，查看其运行结果，即可看到结果如图 12-7 所示。

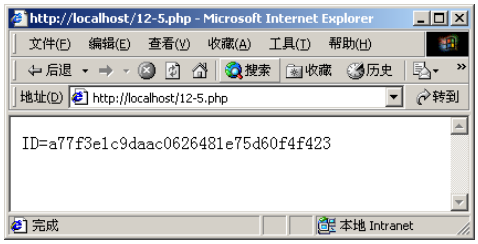

图 12-7　session_id 的值

【代码解析】代码第 02 行启用 Session，第 03 行取得本身 session_id 的值，第 04 行输出 ID 值。

提示：同一台机子上打开不同的浏览器，显示的 ID 也不会一样。

12.3.4 Session 的使用

下面通过一个实例来讲解 Session 的使用。注册 Session 变量，用户名为“PHP 基础教程”，密码为“php”，通过 Cookie 或 URL 方式传递变量。通过 session.php 文件显示用户名、密码和登录时间。其程序如示例代码 12-6 所示。

示例代码 12-6

```
01  <?php                                                    //PHP 开始标记
02      Session_start();                                     //启用 session
03      $_SESSION['name']="PHP 基础教程";                    //注册 SESSION 变量
04      $_SESSION['passwd']="php";                           //注册 SESSION 变量
05      $_SESSION['time']=time();                            //设定时间变量
06      echo '<a href="session.php" >通过 COOKIE 传递 SESSION</a>';
                                                             //COOKIE 方式传递
07      echo '<p ><a href="session.php?' . SID . '">通过 URL 传递 SESSION</a>';
                                                             //URL 方式传递
08  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/12-6.php，查看其运行结果，即可看到结果如图 12-8 所示

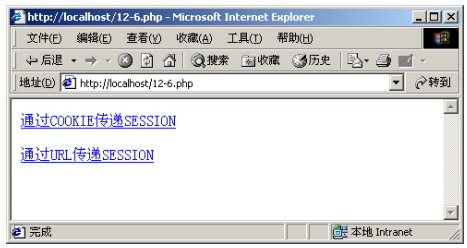

图 12-8　Session 的传递

调用的 session.php 文件的代码如下：

```
01  <?php                                                    //PHP 开始标记
02      session_start();                                     //启用 Session
03      echo $_SESSION['name'];                              //输出用户名
04      echo "<br>";
05      echo $_SESSION['passwd'];                            //输出密码
06      echo "<br>";
07      echo date('Y m d H:i:s', $_SESSION['time']);         //输出时间
08      echo '<br /><a href="12-7.php">返回上一页</a>';      //输出“返回上一页”
09  ?>                                                       //PHP 结束标记
```

【运行结果】单击“通过 Cookie 传递 SESSION”或“通过 URL 传递 SESSION”链接，即可看到如图 12-9 所示的页面。

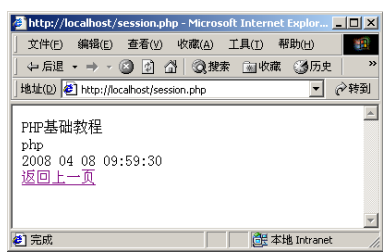

图 12-9　传递 Session 变量

警告：如果将图 12-2 所示的“允许使用每个对话cookies（未存储）”改为“禁用”，则单击“通过COOKIE传递SESSION”后将看不到用户名和访问时间。

12.4 综合练习

1．使用 Cookie 实现用户登录，判断用户名和密码是否输入正确，即是否为“php”。如果不正确，输出“你输入的用户名和密码不正确，请尝试新的输入。”。如果正确，输出“恭喜您!登录成功!”。

提示，建立表格，通过“get”或“post”提交方式，将所得变量赋给 Cookie 变量，并输出其值。其程序如示例代码 12-7 所示。

示例代码 12-7

```
<html>                                                    <!--html 标签开始-->
    <head>                                                <!--头标记开始-->
        <title> COOKIE 实现用户登录</title>                 <!--文件标题-->
    </head>                                               <!--头文件结束-->
       <body>                                             <!--body 标签开始-->
    <center>                                              <!--居中开始标记-->
       <form  name="forml"  method='post'   action='cookie.php'>
                                                          <!--定义表单-->
       <table width='280' height='128'  border='0' aligh='center'
                                                          <!--定义表格-->
    Cellpadding='0'  cells: pacing='1' bgcolor='999999'>  <!--定义间距、颜色-->
    <tr>                                                  <!--行开始-->
      <td colspan='2' align='center' bgcolor='#FFFFFFF'>用户登录</td>
                                                          <!--显示用户登录-->
    </tr>                                                 <!--行结束-->
    <tr>                                                  <!--开始行-->
    <td align='right'  bgcolor='#ffffff'>用户名：</td>     <!--显示用户名-->
     <td align='left'  bgcolor='#ffffff'>                 <!--左对齐-->
       <input type='text'   name='user_name'  size='20' > <!--显示输入框-->
    </td>                                                 <!--列结束-->
    </tr>                                                 <!--行结束-->
    <tr>                                                  <!--开始行-->
     <td align='right' bgcolor='#FFFFFFF'>口令：</td>      <!--显示口令-->
     <td align='left' bgcolor='#FFFFFFF'>                 <!--左对齐-->
       <input  type='password'  name='user_pw'  size="20" > </td>
                                                          <!--显示输入框-->
    </tr>                                                 <!--行结束-->
    <tr>                                                  <!--行开始-->
      <td  colspan='2' align='center'  bgcolor='#FFFFFF'>  <!--居中-->
      <input  type='submit' name='Submit' Value='提交'>  
                                                          <!--提交按钮-->
      <input  type='reset' name='Submit1' Value='重置'></td> <!--重置按钮-->
       </tr>                                              <!--结束行-->
      </table>                                            <!--表格结束标记-->
    </form>                                               <!--表单结束标记-->
    </center>                                             <!--居中结束标记-->
     </body>                                              <!--body 标签结束-->
</html>                                                   <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/12-7.php，查看其运行结果，即可看到示例代码的运行结果如图 12-10 所示。

图 12-10　用户登录

调用的文件 cookie.php 的代码如下：

```
01  <html>                                                    <!--html 标签开始-->
02      <head>                                                <!--头标记开始-->
03          <title>COOKIE 实现用户登录 </TITLE>               <!--文件标题-->
04      </head>                                               <!--头文件结束-->
05      <body>                                                <!-- body 标签开始-->
06  <?php                                                     //PHP 开始标记
07      $_COOKIE{"user_name"}=$_POST{"user_name"};            //取得用户名
08      $_COOKIE{"user_pw"}=$_POST{"user_pw"};                //取得密码
09      if ($_COOKIE{"user_name"}=="php" && $_COOKIE{"user_pw"}=="php"){
                                                              //判断用户名和密码
10          echo "恭喜您!登录成功! ";                        //输出“恭喜您!登录成功!”
11      }else{                                                //用户名或密码不对的情况
12          echo "你输入的用户名为: ".$_COOKIE{"user_name"}; //输出用户名
13          echo  "<br>口令为:".$_COOKIE{"user_pw"};         //输出密码
14          echo "你输入的用户名和密码不正确，请尝试新的输入。";  //输出重新输入提示
15      }
16  ?>                                                        <!--PHP 结束标记-->
17      <p><a href="12-7.php">单击检测 Cookie 的值是否可以页间传递</a>
                                                              <!-- body 标签结束-->
18      </body>                                               <!-- body 标签结束-->
19  </html>                                                   <!-- html 标签结束-->
```

【运行结果】用户名和密码都输入“php”，即可看到图 12-11 所示的结果。

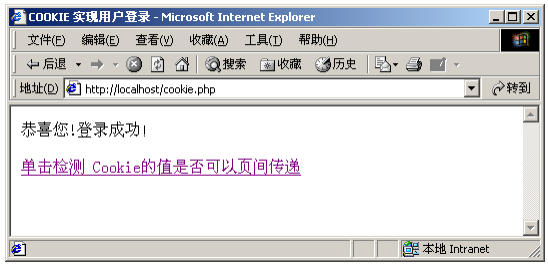

图 12-11　Cookie 传递

2．王军的 math 成绩是 85，chinese 成绩是 96，使用 session 变量输出王军的个人信息，并计算出他的平均成绩，同时输出 sessionid 值。

提示，使用 session_id 函数可获得当前 ID 值。通过$_SESSION 可以调用设定的 session 变量。其示例代码如 12-8 所示。

示例代码 12-8

```
<?php                                          // PHP 开始标记
    session_start();                           //启用 session
    $_SESSION['name']="王军";                  //定义变量
    $_SESSION['math']="85";                    //定义变量
    $_SESSION['chinese']="96";                 //定义变量
    $sessionID= session_id();                  //取得当前 ID
    echo "ID=$sessionID<P> ";                  //输出 ID 值
?>                                             // PHP 结束标记

<html>                                         <!--html 标签开始-->
    <title>调用 Session</title>                //显示标题
    <body>                                     <!--body 标签开始-->
<?php                                          // PHP 开始标记
    echo "<B>学生信息</B> <P>";                //输出“学生信息”
    echo "name= ".$_SESSION['name']."<BR>";    //输出变量值
    echo "math= ".$_SESSION['math']."<BR>";    //输出变量值
    echo "chinese= ".$_SESSION['chinese']."<P>";//输出变量值
    echo "<B>平均成绩</B><P>";                 //输出“平均成绩”
    echo "average=";                           //输出“average=”
    echo ($_SESSION['math']+$_SESSION['chinese'])/2;//计算平均成绩
?>                                             <!--PHP 结束标记-->
</BODY>                                        <!--BODY 标签结束-->
</HTML>                                        <!--html 标签结束-->
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/12.8.php，查看其运行结果，即可看到结果如图 12-12 所示。

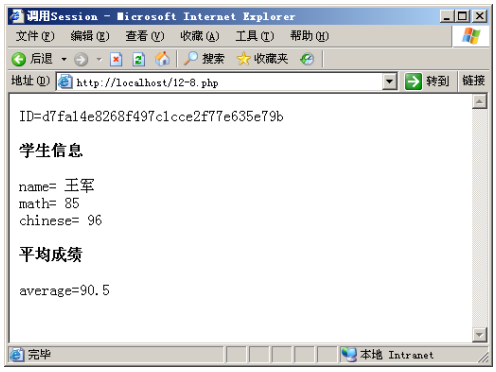

图 12-12　王军个人信息

12.5　小结

本章介绍了 PHP 中的两种存储用户信息的机制，Cookie 和 Session。通过本章的学习，会使读者对如何使用这两种方式存储用户信息有一个比较深刻的认识。其实不管是采用 Session 作为存储的载体还是采用 Cookie 作为存储的载体，关键要看怎么用。有时还可以把二者结合起来使用。

如果读者对本章内容还有疑问，可参考《精通 PHP5 应用开发》（秦涛，曾文玉．北京：人民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民．北京：中国水利水电出版社，2003）。

12.6 习题

一、填空题

1．设置 Cookie 变量 A、B 的值分别为 2008、北京，_____、_____。

2．设置 Cookie 的 3600 秒后失效_____。

3．使用 Session 功能的方式有两种：第一种是_____，第二种是_____。

4．服务器利用_____来区别不同的 Session，用户可以通过_____函数查看。

5．设定 Cookie 变量 A 的值为 2008，失效时间为 2008 年 12 月 31 日前_____。

6．创建 Cookie 的方式是_____。

7．设置 Session 变量 name、password 的值分别为 polo、123456，_____、_____。

8．调用 Cookie 变量的方式是_____，调用 Session 的方式是_____。

二、选择题

1．每个浏览器只能保存某个服务器上 Cookie 数是（　　）。

A．10　　B．20　　C．30　　D．40

2．Cookie 最大长度是（　　）。

A．2KB　　B．3KB　　C．4KB　　D．5KB

3．能够注销变量的函数是（　　）。

A．Session_start()　　B．Session_ragister()　　C．Session_unregister()　　D．session_id()

4．下列哪个变量会立即失效（　　）。

A．setcookie("A","10", time() -3600);　　B．setcookie("B",“10”, time() +3600);

C．setcookie("C","10", mktime(0,0,0,1,1,2010)　　D．setcookie("D","0", mktime(0,0,0,1,1,2009）

三、简答题

1．阐述 Cookie 的作用。

2．阐述调用 Session 的方法。

3．试解释 Cookie 和 Session 的异同。

四、编程题

1．创建一个登录页面，输入昵称和密码，将输入的昵称和密码通过另一个页面显示出来。

2．编写程序查看浏览器 sesion_id。

第 13 章　正则表达式和式样匹配

正则表达式其实就是与字符串匹配的一系列字符，或称为一个字符式样。PHP 支持两种类型的正则表达式：POSIX 和 Perl 样式，每种类型都有一组函数来实现正则表达式。下面详细介绍正则表达式的匹配和 Perl 样式的函数。学习本章，可以获得以下知识点：

- 掌握一般的匹配的模式；
- 熟记常用的元字符意义；
- 掌握一些匹配函数。

13.1　正则表达式的匹配

正则表达式（regular expression）是由一系列的普通字符和元字符组成的一个字符样式，把其放在两个前斜线之间就可以用来匹配字符串。形如/good/：good 包含在两个前斜线之间，就构成了正则表达式。可以匹配字符串“he is a good student”，也就是说，如果字符串里包含 good，样式就与之匹配。

13.1.1　一般匹配

正则表达式的普通字符是指所有的大写和小写字母字符、所有数字、所有标点符号等一些一般意义的字符。由普通字符组成的正则表达式其匹配过程比较简单，下面通过实例讲解简单样式的匹配过程。

【范例 13-1】利用正则表达式判断字符串“Oh my love”中是否含有表达式“love”。如果含有，输出“匹配成功”。如果没有，则输出“匹配不成功”。其程序如示例代码 13-1 所示。

示例代码 13-1

```
01  <?php                                                    //PHP 开始标记
02      $result=preg_match("/love/","Oh my love");           //匹配样式“love”
03      if($result){                                         //判断匹配结果
04          echo "匹配成功<br>";                             //输出“匹配成功”
05      }
06      else                                                 //没有匹配上的情况
07          echo "匹配不成功"                               //输出“匹配不成功”
08  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-1.php，查看其运行结果，即可看到结果如图 13-1 所示。

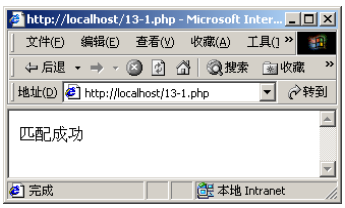

图 13-1　普通匹配

【代码解析】代码第 02 行利用正则表达式/love/判断查找字符串 Oh my love 中是否含有字

符 love。如果找到返回 true，否则返回 flase。第 03～07 行判断查找结果根据判断结果，输出不同结果。

提示：只要字符串中含有连着的“love”4 个字母，即匹配成功。

13.1.2 特殊匹配

如果正则表达式只能由普通字符组成，匹配字面上的字符串则其用处不会太大。实际上，其功能远不止这些。下面我们通过实例了解一下具有特殊功能的匹配。

【范例 13-2】查找字符串“many students have received praise”中是否含有单词“student”。如果含有，输出“匹配成功”；如果没有，则输出“匹配不成功”。其程序如示例代码 13-2 所示。

示例代码 13-2

```
01  <?php                                                    //PHP 开始标记
02      $result=preg_match("/\bstudent \b/","many students have received praise ");
                                                             //匹配样式“student”
03      if($result){                                         //判断匹配结果
04          echo "匹配成功<br>";                             //输出“匹配成功”
05      }
06      else                                                 //没有匹配上的情况
07          echo "匹配不成功"                               //输出“匹配不成功”
08  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-2.php，查看其运行结果，即可看到结果如图 13-2 所示。

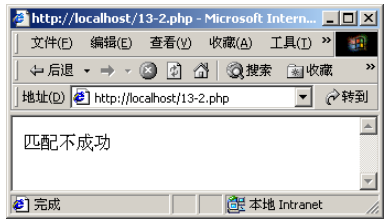

图 13-2 特殊匹配

【代码解析】代码第 02 行利用正则表达式/\bstudent \b/判断查找字符串“many students have received praise”中是否含有单词“student”。\b 表示空格，表示 student 前后要有一个空格，即是独立单词才能匹配得上。如果找到则返回 true，否则返回 flase；第 03～07 行判断查找结果，并根据判断结果，输出结果。

注意：如果去掉\b，则程序将返回真，输出“匹配成功”。

13.2 元字符

正则表达式还有另一组字符——元字符。正则表达式的元字符并不代表其本身，它们被赋予特殊的功能，利用元字符能够以特定的方式控制查找式样。例如脱字符（^）只匹配位于行

首的字符串，句点（.）匹配包括空格在内的任意单个字符。下面分别介绍几个常用的元字符。

13.2.1 行首位置

元字符“^”被称为行首端位，匹配在行首或字符串起始找到的式样。例如/^a/只匹配以 a 开头的字符串。下面通过实例来讲解行首元字符的用法

【范例 13-3】判断获奖名单（"王晓华　刘晓莉　张坤三"）中是否有王晓华，再判断刘晓莉或王晓华是否为第一个，其程序如示例代码 13-3 所示。

示例代码 13-3

```
01  <?php                                          //PHP 开始标记
02      echo "<center> 获奖名单</center> ";        //输出“获奖名单”
03      $string="王晓华　刘晓莉　张坤三";            //变量赋值
04      if(ereg ("刘晓莉", $string)){               //匹配“刘晓莉”
05          echo "有我<br>";                       //输出“有我”
06      }
07      if(ereg ("^刘晓莉", $string)){              //匹配行首位置
08      echo "刘晓莉是第一个<br>";                 //输出“刘晓莉是第一个”
09      }
10      if(ereg ("^王晓华", $string)){              //匹配行首位置
11      echo "王晓华是第一个<br>";                 //输出“王晓华是第一个”
12      }
13  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-3.php，查看其运行结果，即可看到结果如图 13-3 所示。

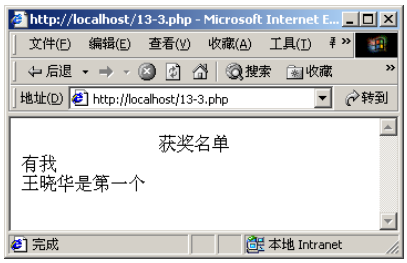

图 13-3　行首的匹配

【代码解析】程序第 02 行将“获奖名单”居中显示。第 04 行判断变量$string 中是否含有刘晓莉。第 07 行判断刘晓莉是否在行首，第 11 行判断王晓华是否在行首。

提示：ereg 函数是 POSIX 风格的函数，用来查找匹配样式，其能方便地验证数据的格式是否正确。

13.2.2 行尾位置

行尾端位元字符是$，它匹配行的末尾位置。美元符号必须是式样里的最后一个字符，只能放在紧靠正则表达式结束斜线符号的位置，否则其就不能匹配行尾位置了。

【范例 13-4】判断获奖名单中（王晓华　刘晓莉　张坤三）谁处在行尾。其程序如示例代码 13-4 所示。

示例代码 13-4

```
01  <?php                                                    //PHP 开始标记
02      echo "<center> 获奖名单</center> ";                  //输出“获奖名单”
03      $string="王晓华  刘晓莉  张坤三";                     //变量赋值
04      if(ereg ("王晓华$", $string)){                       //匹配行尾位置
05          echo "王晓华是最后一个<br>";                     //输出“王晓华是最后一个”
06      }
07      else                                                 //王晓华不再最后位置的情况
08      if(ereg ("张坤三$", $string)){                       //匹配行尾位置
09          echo "张坤三是最后一个<br>";                     //输出“张坤三是最后一个”
10      }
11      else echo "刘晓莉是最后一个"                        //输出“刘晓莉是最后一个”
12  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-4.php，查看其运行结果，即可看到结果如图 13-4 所示。

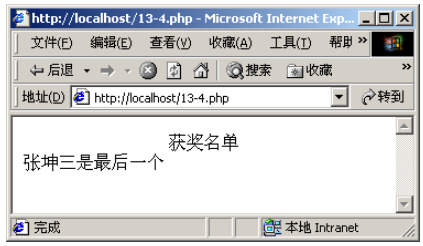

图 13-4　行尾的匹配

【代码解析】程序第 02 行将“获奖名单”居中显示。第 04、08 行分别判断“王晓华”、“张坤三”是否处于行尾。

13.2.3　元字符+

元字符+用来匹配前面的子表达式一次或多次。在正则表达式/a+bc/里加号附加于字符 a，表示匹配一个或多个字符。比如字符 abc、aabc、aaaabc 都能匹配这个表达式。

【范例 13-5】判断邮箱地址 php@php.com 是否合法，要求判断是否含有@字符，字符@前面要有若干个字母或数字的组合。再判断网址是否合法，如果合法输出正确的邮件地址，如果不合法则提示 E-mail 地址不合法。其程序如示例代码 13-5 所示。

示例代码 13-5

```
01  <?php                                                    //PHP 开始标记
02      $Email="php@php.com";                                //变量赋值
03      if (ereg ("([0-9a-zA-Z]+)([@])([0-9a-zA-Z]+)(.)([0-9a-zA-Z]+)", $Email,
$regs)) {                                                    //匹配邮件地址
04          echo "$regs[1]$regs[2]$regs[3]$regs[4]$regs[5]";   //输出正确的地址
05      }
06      else {                                               //地址不正确情况
07          echo "Invalid Email format : $Email";           //输出错误提示
08      }
09  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-5.php，查看其运行结果，即可看到结果如图 13-5 所示。

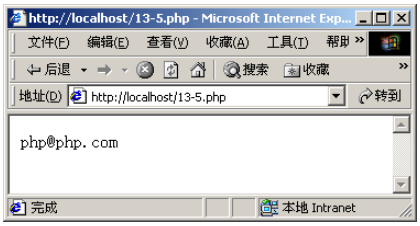

图 13-5　重复匹配

【代码解析】程序第 02 行地址赋给变量$Email，第 03 行利用元字符判断邮件地址是否合法，[0-9a-zA-Z]匹配数字或字母中的任意一个。第 04 行输出正确的邮件地址，第 07 行提示邮件地址格式不正确。

提示：除了利用元字符+表示一次或多次，还可以利用一对大括号包围的数字来指定匹配的次数。例如{2，4}表示匹配重复 2～4 次，{4}表示匹配重复 4 次。。

13.2.4　字符类

字符类是方括号[]中的一列字符，可以匹配方括号内出现的任意单个字符。例如[abc]匹配 a、b 或 c；[0-9]匹配 0～9 之间的任一个数字。另外，方括号内也可以包含元字符，例如包含脱字符的[^abc]匹配不含 a、b 和 c 的单个字符。

【范例 13-6】判断日期变量$date 是否具有 YYYY-MM-DD 格式。如果具有此格式，则按年月日格式输出，否则，提示其没有合法的格式。其程序如示例代码 13-6 所示。

示例代码 13-6

```
01  <?php                                                    //PHP 开始标记
02      $date="2008-08-08";                                  //变量赋值
03      if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
                                                             //匹配日期格式
04          echo "$regs[1] 年 $regs[2] 月 $regs[3] 日";      //按年月日格式输出
05      }
06      else {                                               //格式不正确的情况
07          echo "Invalid date format: $date";             //输出错误提示
08      }
09  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-6.php，查看其运行结果，即可看到结果如图 13-6 所示。

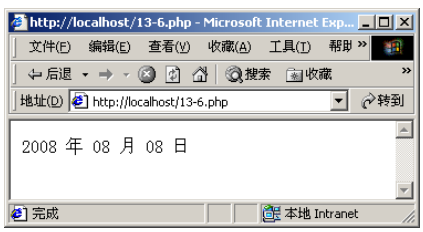

图 13-6　字符类匹配

【代码解析】代码第 02 行将日期“2008-08-08”赋给变量$date。第 03 行判断其格式是否符合要求。第 04 行按年月日格式输出此日期。第 07 行提示没有合适的日期格式。

提示：PHP 还提供了代表字符类的元符号，符号\d 和\D 分别代表单个数字和单个非数字，相当于[0-9]和[^0-9]。\w 和\W 分别代表单个字符和单个非字符，相当于[A-Za-z 0-9]和[^ A-Za-z0-9]。

13.3 样式匹配函数

PHP 支持 Perl 样式的正则表达式，Perl 类型提供一组函数来实现正则表达式，其函数名称前缀是 preg_。在使用这些函数之前，要查看 PHP 是否支持 Perl 表达式，检查测试脚本 phpinfo()函数的输出结果是否有如图 13-7 所示的两项。如果有，即表示已支持 Perl 提供的函数；如果没有，下载相关文件，复制到 php5 文件夹里即可。下面具体讲解 Perl 提供的常用函数。

pcre

PCRE (Perl Compatible Regular Expressions) Support	enabled
PCRE Library Version	7.0 18-Dec-2006

图 13-7 phpinfo()函数的输出结果

13.3.1 表达式匹配

函数 preg_match()查找样式，其结构形式如下：

```
int preg_match ( string $pattern , string $subject [, array $matches [, int $flags ]] )
```

preg_match()函数返回参数 pattern 所匹配的次数，返回 0（没有匹配）或 1，因为 preg_match()在第一次匹配之后将停止搜索。第一个参数 pattern 提供正则表达式。第二个参数 subject 指定要被匹配的项。第三个参数 matches 可选，如果提供了参数 matches，则返回一个数组。$matches[0]包含与整个模式匹配的文本，$matches[1]包含与第一个捕获的括号中的子模式所匹配的文本，依此类推。第四个参数 flags 也是可选参数，只能使用 PREG_OFFSET_CAPTUR E 标记。如果设定本标记，对每个出现的匹配结果也同时返回其附属的字符串偏移量。

【范例 13-7】分别在字符串“PHP is the web scripting language of choice”和“PHP is the website scripting language of choice”中查找是否含有单独的单词 web，进行表达式匹配。其程序如示例代码 13-7 所示。

示例代码 13-7

```
01  <?php                                                    //PHP 开始标记
02  $string=preg_match ("/\bweb\b/i", "PHP is the web scripting language of
choice.",$matches);                                          //匹配
03  if ($string==true){                                      //判断匹配结果
04       print "A match was found.<br>";                     //输出“A match was found”
05       print_r($matches);                                  //输出数组变量
06       echo "<br>";                                        //换行
07       } else {                                            //没有匹配上的情况
08       print "A match was not found.<br>";                 //输出“A match was not found”
09       print_r($matches);                                  //输出数组变量
10       echo "<br>";                                        //换行
11       }
12       $string=preg_match ("/\bweb\b/i", "PHP is the website scripting language
of choice.",$matches);
13       if ($string==true) {                                //判断匹配结果
```

```
14        print "A match was found.";                    //输出“A match was found”
15       print_r($matches);                               //输出数组变量
16       echo "<br>";                                     //换行
17       } else {                                         //没有匹配上的情况
18        print "A match was not found.<br>";            //输出“A match was not found”
19        print_r($matches);                              //输出数组变量
20       }
21   ?>                                                   //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-7.php，查看其运行结果，即可看到结果如图 13-8 所示。

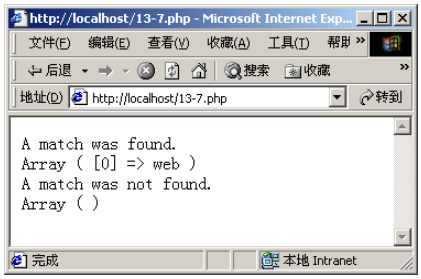

图 13-8　preg_match 进行正则表达式匹配

【代码解析】代码第 02 行利用 preg_match 函数匹配字符串中是否含有“web”。模式中的\b 表示单词的边界，因此只有独立的“web”单词会被匹配，而不会匹配例如“cobweb”中的一部分。第 03 行判断匹配结果，第 09 行输出匹配后的结果。第 12 行再次利用 preg_match 函数匹配字符串中是否含有“web”。因为字符串中没有单独的单词“web”，因此匹配失败返回 0，则输出失败提示。

注意：设定参数 flags 后，返回的是一个二维数组，其中第一项为匹配字符串，第二项为其偏移量。

13.3.2　全局匹配

函数 preg_match_all()进行全局匹配正则表达式，其结构形式为：

```
int preg_match_all ( string $pattern , string $subject , array $matches [, int $flags ] )
```

函数 preg_match_all()和 preg_match()的用法类似，但前者是查找整个字符串，匹配所有和正则表达式匹配的样式，而不是只找到第一个，并且返回找到的式样数量。

参数 flags 有两个标记：第一个为 PREG_PATTERN_ORDER 标记，对结果排序。第二个为 PREG_OFFSET_CAPTURE 标记，对每个出现的匹配结果也同时返回其附属的字符串偏移量。

【范例 13-8】查找字符串“PHP is a language,PHP is a web langage,php”中 PHP 出现的次数，并以数组形式输出，如果不存在则输出“PHP was not found”。其程序如示例代码 13-8 所示。

示例代码 13-8

```
01   <?php                                                          //PHP 开始标记
02        $string="PHP is a language,PHP is a web langage,php";     //变量赋值
03        $result=preg_match_all("/PHP/",$string,$matches, PREG_PATTERN_ORDER);
                                                                    //匹配
04        if ($result==true){                                       //判断结果
```

```
05        print "$result PHP was found.<br>";                    //输出匹配上信息
06        print_r($matches);                                     //输出数组变量
07       echo "<br>";                                          //换行
08       } else {                                               //没有匹配上的情况
09        print "PHP was not found.<br>";                       //输出没有配上的情况
10        print_r($matches);                                     //输出数组变量
11        echo "<br>";                                          //换行
12       }
13   ?>                                                         //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-8.php，查看其运行结果，即可看到结果如图 13-9 所示。

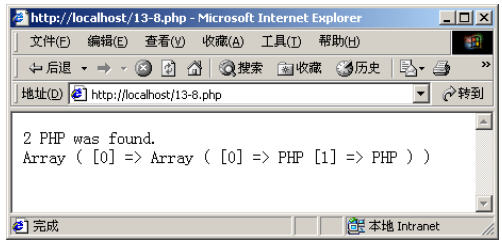

图 13-9　preg_match_all 全局匹配

【代码解析】代码第 03 行匹配查找字符串是否含有几个字符串“PHP”，并返回查找到的数目。第 04 行判断查找结果，第 06 行输出返回的数组元素值。第 08～12 行输出找不到的情况。

警告：如果没有给出标记，则默认为 PREG_PATTERN_ORDER，函数返回整个模式匹配的次数（可能为零）。如果出错则返回 false。

如果要知道匹配的样式的位置，则可设定参数 flags 的标记为 PREG_OFFSET_CAPTURE。

【范例 13-9】标出范例中 13-8 所示字符串“PHP”出现的位置的位置（不区分大小写），其程序如示例代码 13-9 所示。

示例代码 13-9

```
01   <?php                                                      //PHP 开始标记
02       $string="PHP is a language,PHP is a web langage,php"; //变量赋值
03       $result=preg_match_all("/PHP/i",$string,$matches,PREG_OFFSET_CAPTURE);
                                                                //匹配
04       if ($result==true){                                    //判断匹配结果
05       print "$result PHP was found.<br>";                    //输出匹配上信息
06       print_r($matches);                                     //输出数组变量
07       echo "<br>";                                          //换行
08       }
09       else {                                                 //没有匹配的上情况
10            print "PHP was not found.<br>";                   //输出没有匹配上的
                                                                信息
11            print_r($matches);                                //输出数组变量
12            echo "<br>";                                     //换行
13       }
14   ?>                                                                 //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-9.php，查看其运行结果，即可看到结果如图 13-10 所示。

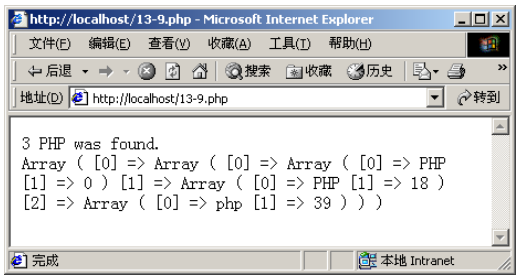

图 13-10　返回位置

【代码解析】程序第 03 行查找字符串变量$string 中是否含有几个字符串“PHP”，PREG_OFFSET_CAPTURE 标记可以显示每个式样在字符串中出现的位置，正则表达式/PHP/i 表示不区分大小写的搜索。第 04～08 行输出查找到的情况。第 06 行输出返回字符数组的值。第 09～11 行输出返回失败的情况。

注意：把 PREG_PATTERN_ORDER 和 PREG_SET_ORDER 合起来用没有意义。

13.3.3　搜索和替换

函数 preg_replace()执行正则表达式的搜索和替换，其结构形式为：

```
mixed preg_replace ( mixed $pattern , mixed $replacement , mixed $subject [, int $limit ] )
```

第一个参数 pattern 为要被替换的表达式样式。第二个参数 replacement 为要替换的新样式。第三个参数 subject 为被查找的目标。第四个参数 limit 可选，其为要替换操作的次数。

【范例 13-10】将字符串“The quick brown fox jumped over the lazy dog.”中的“quick”、“brown”、“fox”分别替换为“bear”、“black”、“slow”，然后将字符串逆序排列，再进行替换。其程序如示例代码 13-10 所示。

示例代码 13-10

```
01  <?php                                                          //PHP 开始标记
02      $string = "The quick brown fox jumped over the lazy dog.<br>";
                                                                   //变量赋值
03      $patterns[0] = "/quick/";                                  //被替换的值
04      $patterns[1] = "/brown/";                                  //被替换的值
05      $patterns[2] = "/fox/";                                    //被替换的值
06      $replacements[2] = "bear";                                 //要替换的新值
07      $replacements[1] = "black";                                //要替换的新值
08      $replacements[0] = "slow";                                 //要替换的新值
09      print preg_replace($patterns, $replacements, $string);     //替换
10      ksort($replacements);                                      //对数组元素排序
11      print preg_replace($patterns, $replacements, $string);     //替换
12  ?>                                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-10.php，查看其运行结果，即可看到结果如图 13-11 所示。

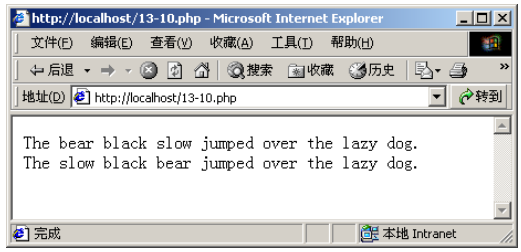

图 13-11　preg_replace 替换

【代码解析】程序第 03～05 行定义了要被替换的字符串。第 06～08 行定义要替换的新字符串。第 09 行进行字符串的替换。第 10 行对数组元素$replacements 进行逆序排列。第 11 行再次替换原来的字符串。

注意：参数 limit 的默认值为-1，替换所有的匹配项。

13.3.4　分隔字符串

函数 preg_split()用来分隔字符串，其结构形式为：

```
array preg_split ( string $pattern , string $subject [, int $limit [, int $flags ]] )
```

第一个参数 pattern 设定按哪种样式分隔字符串（如空格、逗号等），第二个参数 subject 为要被分隔的字符串。第三个参数 limit 可选，设定分隔的数目，第四个参数 flags 可选，设定一些特殊的功能。其可选标记和功能如表 13-1 所示。

表 13-1　可选标记和功能

标　记	功　能
PREG_SPLIT_NO_EMPTY	只返回非空的元素
PREG_SPLIT_DELIM_CAPTURE	定界符模式中的括号表达式也会被捕获并返回
PREG_SPLIT_OFFSET_CAPTURE	对每个出现的匹配结果同时返回其附属的字符串偏移量

【范例 13-11】将字符串“PHP is a Web language”按单词分隔，将其存放到数组并返回每个单词在原字符串中的位置。其程序如示例代码 13-11 所示。

示例代码 13-11

```
01  <?php                                                    //PHP 开始标记
02      $str = 'PHP is a Web language';                      //变量赋值
03      $chars = preg_split('/ /', $str, -1, PREG_SPLIT_OFFSET_CAPTURE);
                                                             //按空格分隔字符串
04      echo "<h2> 被分隔成单词了：</h2>";                    //输出标题字
05      print_r($chars);                                     //输出数组元素
06  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-11.php，查看其运行结果，即可看到结果如图 13-12 所示。

【代码解析】程序第 03 行将字符串变量$str 以空格为分界分隔成数组，设定 PREG_SPLIT_OFFSET_CAPTURE 标记，返回字符串在原字符串中的位置。第 05 行输出字符串元素。

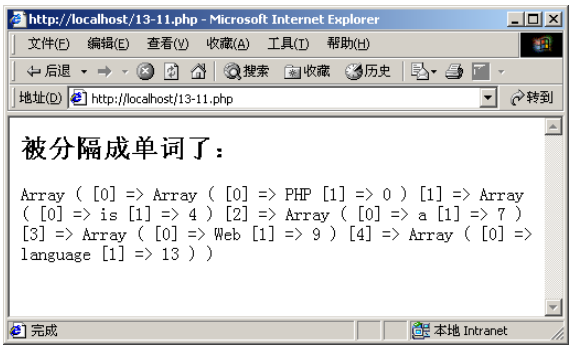

图 13-12　分隔字符串

注意：如果指定了 limit，则最多返回 limit 个子串。如果 limit 是-1，则意味着没有限制，可以用来继续指定可选参数 flags。

13.4　综合练习

1．利用正则表达式匹配“标题: 21 世纪展望 ”，内容按格式输出到浏览器上，再将题目改为“历史回顾”。

提示，利用函数 preg_match_all 匹配进行，可以匹配标记“<>”，替换利用函数 preg_replace 进行替换。其程序如示例代码 13-12 所示。

示例代码 13-12

```
<?php                                                          //PHP 开始标记
    $string="<b>标题: </b><div align=center>21 世纪展望</div>";   //变量赋值
    preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",                  //匹配函数
        $string,                                               //要查找的项
        $out, PREG_PATTERN_ORDER);                             //返回的数组
    print $out[0][0]." ".$out[0][1]."<p>";                     //输出结果
        //print $out[1][0]." ".$out[1][1];
    $patterns= "/21 世纪展望/";                                 //被替换样式
    $replacements = "回顾历史";                                 //变量赋值
    print preg_replace($patterns, $replacements, $string);     //替换
?>                                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-12.php，查看其运行结果，即可看到结果如图 13-13 所示。

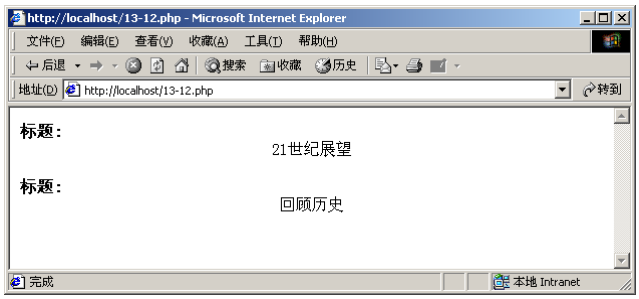

图 13-13　正则表达式匹配

2．通过正则表达式判断网址是否合法，一类判断（.com.cn），一类判断（com|an|me）等。如果正确输出正确的网址，否则输出提示信息。

提示，要求必须以 http 开头，要以.com.cn 结尾，可以利用 ereg 函做判断。其程序如示例代码 13-13 所示。

示例代码 13-13

```
<?php
    $https="http://www.phei.com.cn";
    if (ereg ("(^http)(://)(www\.)([0-9a-zA-Z]+)+(.com)(.cn)", $https, $regs)) {
                                                                    //判段表达式
        echo "$regs[1]$regs[2]$regs[3]$regs[4]$regs[5]$regs[6]";
                                                                    //输出正确的表达式
    echo "<br>";
    }
    else {
        echo "你输入的网址不正确 : $Email";                          //输出错误提示
    echo "<br>";
}
  $https="http://www.163.com";
    if (ereg ("(^http)(://)(www\.)([0-9a-zA-Z]+)(.)+(com|cn|net|me)", $https,
$array)) {                                                          //判段表达式
        print_r($array);                                            //输出数组
echo "<br>";
    }
    else {
        echo "你输入的网址不正确 : $Email";
echo "<br>";
}
?>
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/13-13.php，查看其运行结果，即可看到结果如图 13-14 所示。

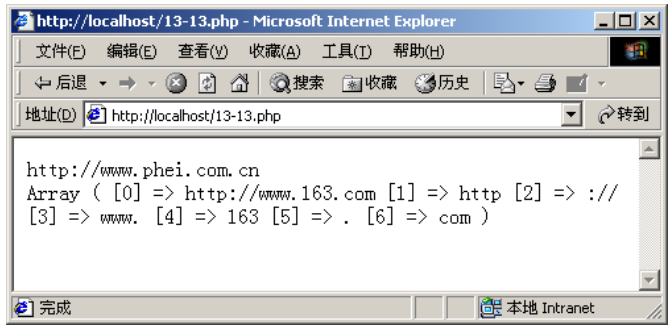

图 13-14　判断网址

13.5　小结

本章主要讲述了正则表达式匹配规则和一些元字符的应用，以及 PHP 中处理它们的函数。正则表达式对检验数据是否合法非常有用，其实现过程也十分简单有效，掌握正则表达后对简化查找、替换、判断等过程很有帮助。正则表达式的功能很强大，本章只是一个处理匹配的基础，想了解更多正则表达式的知识可参阅《精通 PHP 5 应用开发》（秦涛，曾文玉．北京：人

民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民．北京：中国水利水电出版社，2003）。

13.6 习题

一、填空题

1．进行全局正则表达式匹配的函数是_____。

2．PHP 还提供了代表字符类的元符号：符号_____、_____分别代表单个数字和单个非数字，相当于[0-9]和[^0-9]。

3．元字符+用来匹配前面的子表达式_____。

4．正则表达式/^abc/的意义是_____。

5．利用正则表达式分隔字符串的函数是_____。

6．函数 preg_replace_callback()和 preg_replace()作用一样，除了不是提供一个 replacement 参数，而是指定一个_____。

7．字符簇[0-9\.\-]表示的意义是_____。

8．字符簇[^a-z]表示的意义是_____。

二、选择题

1．元字符$代表的意义是（　　）。

A．匹配字符串开始位置　　B．匹配字符串的结尾位置

C．匹配字符串开始和结束位置　　D．匹配字符串任何位置

2．元字符 a{3，}表示的意义是（　　）。

A．匹配 a、{、，　　B．匹配重复 3 次的字母 a

C．匹配少于重复 3 次的字母 a　　D．匹配重复 3 次及以上的字母 a

3．正则表达式/href='(.*)'表示（　　）。

A．合法　　B．含有未知修正符

C．缺少起始定界符　　D．缺少结束定界符

4．正则表达式/abc$/表示（　　）。

A．匹配字符串 abc$　　B．匹配以 abc 开头的字符串

C．匹配以 abc 结尾的字符串　　D．匹配任意的 abc 字符串

5．下列程序的运行结果为（　　）。

```
$result=preg_match("/word/","This string have ten words");
if($result){
    echo "匹配成功<br>";
}
else
    echo "匹配不成功"
```

A．无任何输出　　B．输出“匹配成功”

C．输出“匹配不成功”　　D．程序有误

三、简答题

1．简要说明什么是字符簇、它们的作用是什么。

2．阐述 preg_match_all()函数中各参数的意义。

四、编程题

1．利用正则表达式判断邮件地址是否合法。

2．利用正则表达式匹配“<H3>留言板</H3> 请您留言： ”，内容按格式输出到浏览器上，再将题目改为“历史回顾”。

第 14 章　错误调试与异常处理

编写程序时，难免会出现这样或那样的错误，调试程序是一个漫长的过程，通常令人感到头疼。如果掌握一些常用的错误处理方法，不但能在面对问题时不再感到无从下手，而且能快速地找到错误的根源。

学习本章，可以获得以下知识点：

- 掌握调试策略；
- 了解错误报告；
- 常见的错误类型；
- 处理错误信息的手段及方法。

14.1　基本调试策略

在使用 PHP 编写程序时，难免会出现一些错误，包括复杂的逻辑错误和简单的语法错误，掌握一些必要的调试策略可以节省大量的时间。下面简单介绍调试的策略。

14.1.1　调试步骤

调试应该遵循其一般步骤，首先判断错误最可能出现在哪一环节，然后针对该环节采取一些有效的措施来查找错误，如修改程序，测试结果等。其具体流程图如图 14-1 所示。

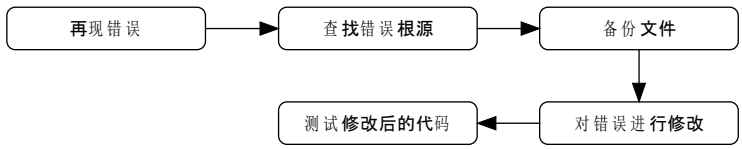

图 14-1　调试步骤

上述这些只是进行错误调试工作的一个简单流程。调试手段主要有增加中间变量或跟踪变量、注释掉部分代码、通过调试器调试等，具体的调试手段还是由程序设计者自己掌握。

提示：程序调试时，可按以上步骤进行一步步调试。

14.1.2　配置指令

处理错误首先要查找错误，查找错误可以根据 PHP 提供的错误指令来进行错误提示，也可以编写代码进行错误的查找。这里首先了解错误报告的配置指令，如表 14-1 列出的是 php.ini

能够显示的错误类型的指令及作用说明。

表 14-1　错误指令

指　令	说　明
E_ALL	所有的错误和警告（不包含 E_STRICT）
E_ERROR	致命的运行时错误

续表

指　令	说　明
E_RECOVERABLE_ERROR	几乎致命的运行时错误
E_WARNING	运行时的警告（非致命错误）
E_PARSE	编译时解析错误
E_NOTICE	运行时的提示，这些提示常常是代码中的 bug 引起的，也许是故意的（如使用一个未初始化的变量，事实上它被自动初始化成一个空字符串）
E_STRICT	运行时提示，能够给予 PHP 建议改变代码，以获得最好的协同性，并完善代码的兼容性
E_CORE_ERROR	PHP 初始化启动过程中的致命错误
E_CORE_WARNING	PHP 初始化启动过程中的非致命错误
E_COMPILE_ERROR	致命的编译错误
E_COMPILE_WARNING	非致命的编译错误
E_USER_ERROR	用户错误信息
E_USER_WARNING	用户警告信息
E_USER_NOTICE	用户提示信息

以下几种是 php.ini 中推荐的几种配置。

在 php.ini 中 error_reporting 控制输出到用户端的消息种类，如图 14-2 所示是 php.ini 默认的输出方式，表示输出所有的错误信息。

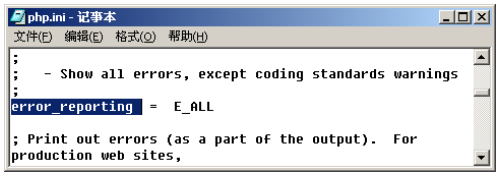

图 14-2　php.ini 配置

另外还有几种可供选择的输出方式。

```
error_reporting = E_ALL
```

表示输出所有的信息。

```
error_reporting= E_COMPILE_ERROR|E_RECOVERABLE_ERROR|E_ERROR|E_CORE_ERROR
```

表示输出所有的 ERROR 信息。

在 php.ini 文件中，display_errors 可以设置是否将以上设置的错误信息输出到用户端，如图 14-3 所示，on 表示要显示，off 表示不显示。

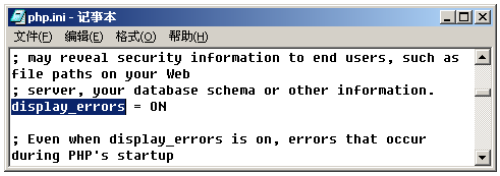

图 14-3　display_errors 设置

提示：常见错误类型有语法错误、运行错误、逻辑错误。下面分别介绍这几种常见的数据类型。

14.2 语法错误

语法错误是指不符合 PHP 语言风格所造成的错误。如 PHP 利用“<?”和“?>”标记开始和结束，由于疏忽，程序结束时忘记了标写“?>”符号，这样就会造成程序运行错误，形如这类错误称为语法错误。在所有错误类型中，语法错误是最常见的。下列就列举一些容易出现的语法错误，以便在编写程序时注意避免此类错误。

14.2.1 缺少分号

缺少分号是 PHP 编程中常见的语法错误。分号意味着一句执行语句的结束，缺少分号就是指当一个语句执行完后，却没有书写分号，程序无法识别语句的结束，造成下一个执行语句错误，从而使程序无法运行。

【范例 14-1】分别给变量 a、b 赋值 1、6，然后分别输出两变量的值，其程序如示例代码 14-1 所示。

示例代码 14-1

```
01  <?php                                    //PHP 开始标记
02      $a=1                                 //缺少分号
03      $b=6;                                //变量赋值
04      echo  $a,$b                          //输出变量值
05  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-1.php，查看其运行结果，即可看到结果如图 14-4 所示。

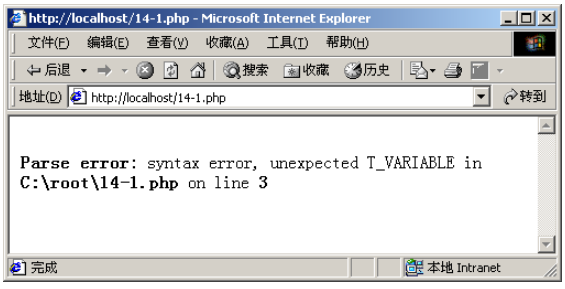

图 14-4 缺少分号

【代码解析】代码第 02 行后面缺少了分号，程序在第 02 行没有遇到分号，所以不认为一个执行语句结束，缺少分号的错误提示都是推后一行。修改上述错误非常简单，只需在“$a=1”后面添加一个分号即可。

提示：如果是程序的结尾位置，如示例代码的 14-1 的 04 行，分号可以省略。

14.2.2 缺少一个引号

缺少引号是另一个常见的语法错误。引号是成对出现的，缺少一个引号就是只有一个单独的引号，即只有开始或结束位置的引号。当只有一个引号时，程序找不到另一个引号的位置，就会提示错误信息。

【范例 14-2】输出字符串“I want to go home”，查看缺少一个引号时的错误提示信息，其程序如示例代码 14-2 所示。

示例代码 14-2

```
01  <?php                                  //PHP 开始标记
02      $a="I want to go home;             //变量赋值
03      $b=6;                              //变量赋值
04      echo  $a,$b;                       //缺少引号
05  ?>                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-2.php，查看其运行结果，即可看到结果如图 14-5 所示。

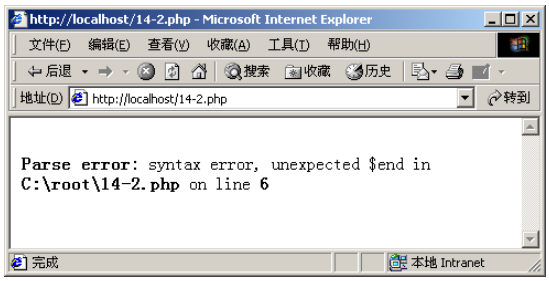

图 14-5　缺少一个引号

【代码解析】代码第 02 行后面缺少了半个引号，由于程序匹配到另一个引号才视为字符串的结束，所以程序执行到结束，提示信息在程序结束的第 06 行。修改上述错误也非常容易，只需在“I want to go home”上补充完整引号。

警告：如果是缺少前半个引号，其提示信息不同于图 14-5，主要原因是其无法寻找引号开始位置。

14.2.3　缺少整个引号

当使用 echo 命令输出字符串不加引号时，程序就无法识别命令执行的内容，将造成程序的错误。

【范例 14-3】输出字符串“I want to go home”，查看缺少引号时的错误提示信息。其程序如示例代码 14-3 所示。

示例代码 14-3

```
01  <?php                                  //PHP 开始标记
02      $a=I want to go home;              //变量赋值
03      $b=6;                              //变量赋值
04      echo  $a,$b;                       //缺少引号
05  ?>                                     //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-3.php，查看其运行结果，即可看到示例代码的运行结果如图 14-6 所示。

【代码解析】代码第 02 行缺少了引号，程序无法识别后面的程序，提示在 02 行出错，程序更正只需将“I want to go home”上加上引号即可。

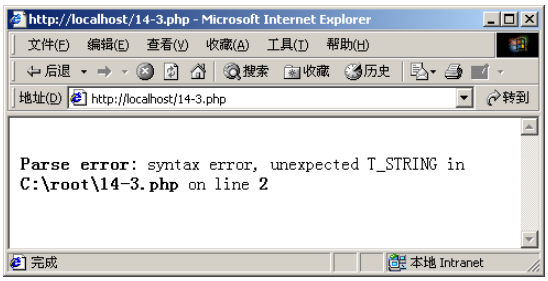

图 14-6　缺少整个引号

提示：其他情况下缺少引号，可以造成与上述情况不同的结果，这根据具体情况而定。

14.2.4　缺少关键字

缺少关键字是指缺少一些命令或函数，程序不能正确识别代码要进行的操作而造成错误，如输出时忘记写 echo 命令等。

【范例 14-4】利用 switch 判断变量 i 的值，分别输出“今天开始学习”、“今天可以休息”、“该发钱了”，其程序如示例代码 14-4 所示。

示例代码 14-4

```
<?php                                    //PHP 开始标记
        $i = 2;                          //变量赋值
    switch($i){                          // switch 循环
    case 2:                              //情况 1
        echo "今天开始学习<br>";          //输出“今天开始学习”
    case 1:                              //情况 2
        echo "今天可以休息<br>";          //输出“今天可以休息”
    case 0:                              //情况 3
        echo "该发钱了<br>";              //输出“该发钱了”
    }
?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-4.php，查看其运行结果，即可看到结果如图 14-7 所示。

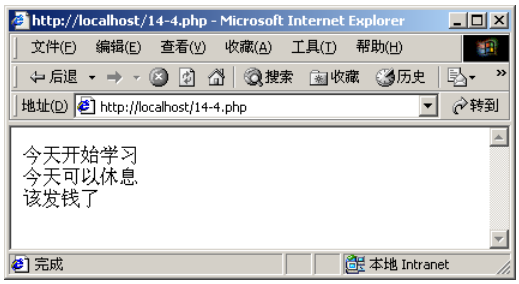

图 14-7　缺少关键字

【代码解析】从输出结果可以看出程序不加判断便输出了结果，这是缺少 break 语句的结果，修改程序只须在每条 case 语句最后加上 break 语句即可。

14.2.5 缺少括号

缺少括号是指应该有括号的没有加上括号，或者括号没有成对出现。这类语法错误可能引起致命错误，所以应加以注意。

【范例 14-5】输出小于 10 不能被 2 整除的数，被 2 整除的数提示“不输出此值”。其程序如示例代码 14-5 所示。

示例代码 14-5

```
01  <?php                                          //PHP 开始标记
02      for($n=1;$n<=10;$n++){                     //循环条件
03          if($n%2==0){                           //判断表达式
04      echo "不输出此值<br>";                     //输出“不输出此值”
05          continue;                              //结束本次循环
06      echo $n."<br>";                            //输出变量值
07      }
08  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-5.php，查看其运行结果，即可看到结果如图 14-8 所示。

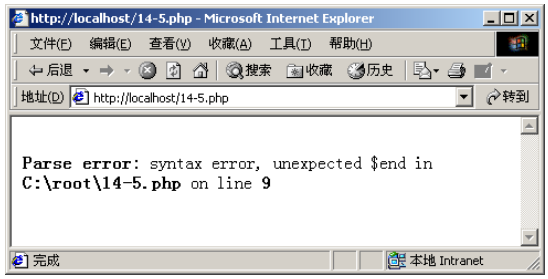

图 14-8　缺少括号

【代码解析】程序不能判断语句的结束，检查发现 if 语句的大括号没有封闭，修改程序时只需在第 05 行补充上大括号。

14.2.6 变量前缺少符号

变量前缺少符号指缺少声明变量的$符号。在 PHP 中，声明变量时在变量名前添加$符号，计算机才能识别其代码性质，如果缺少，将导致错误。

【范例 14-6】分别给变量 a、b 赋值 1、6，然后分别输出两变量的值，变量前缺少$符号。其程序如示例代码 14-6 所示。

示例代码 14-6

```
01  <?php                                          //PHP 开始标记
02      a=1;                                       //变量赋值
03      $b=6;                                      //变量赋值
04      echo  $a,$b;                               //输出变量
05  ?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-6.php，查看其运行结果，即可看到结果如图 14-9 所示。

图 14-9　变量前缺少符号

【代码解析】程序第 02 行声明变量时没有加$符号，程序无法识别“=”号的意义，因此提示错误。要改正代码，只需在第 02 行 a 前加上$符号。

说明：以上介绍了一些常见的语法错误及修改方法。对于语法错误，只要编程人员在书写代码时认真一些并且根据提示查找错误，是比较容易解决的。

14.3　运行错误

运行错误是指程序本身没有语法错误，而程序不能被执行。如调用的文件不存在、越权操作、连接数据密码不正确等都是运行错误，下面简要介绍几种此类错误。

14.3.1　文件操作与权限

编程人员在使用 fopen 函数进行文件操作时，如果没有考虑到被操作文件的权限，则可能导致运行错误。

【范例 14-7】已进行的操作超出了所有的权限，其程序如示例代码 14-7 所示。

示例代码 14-7

```
01  <?php                                    //PHP 开始标记
02      $fp=fopen("foo.txt","r");            //以只读方式打开文件
03      $fw=fwrite($fp,"hello world");       //写入数据
04      if($fw)                              //判断
05      echo "写入成功";                     //输出“写入成功”
06      else                                 //写入失败的情况
07      echo "失败";                        //输出“失败”
08      fclose($fp);                         //关闭文件
09  ?>                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-7.php，查看其运行结果，即可看到结果如图 14-10 所示。

【代码解析】这是由于 foo.txt 文件不存在引起的，它和 6.4.3 节中调用 require()函数所犯的错误一样。如果文件存在，由于 foo.txt 文件以只读文件打开，数据不会被写入文件。

提示：解决这种问题的方法是首先获得该文件的属性，判断是否能进行此项操作，如果超越了权限会导致错误。

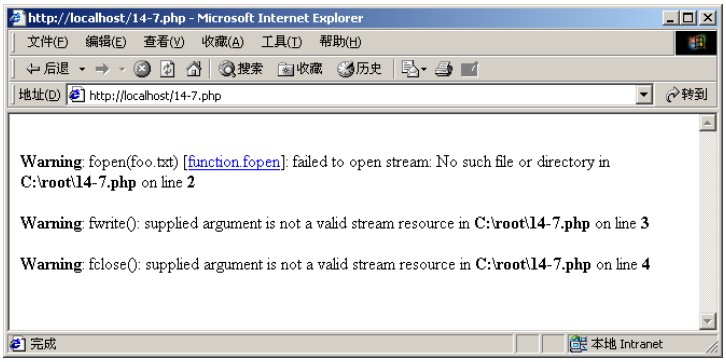

图 14-10　访问受限

14.3.2　连接数据库失败

不能连接数据库的情况很多，可能是用户名、密码不正确，或者服务器工作不正常等原因，如果连接不上或者访问受限都会报告错误。下面通过示例查看数据库的连接。

【范例 14-8】数据库的本地数据主机名 localhost，用户名为 root，密码为 123456，连接本地数据库。其程序如示例代码 14-8 所示。

示例代码 14-8

```
01  <?php                                                    //PHP 开始标记
02      $link=mysql_connect("localhost","root","12345");     //连接数据库
03      if(!$link) echo "失败!";                             //输出“失败!”
04      else echo "成功!";                                   //输出“成功”
05      mysql_close();                                       //关闭
06  ?>                                                       //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-8.php，查看其运行结果，即可看到结果如图 14-11 所示。

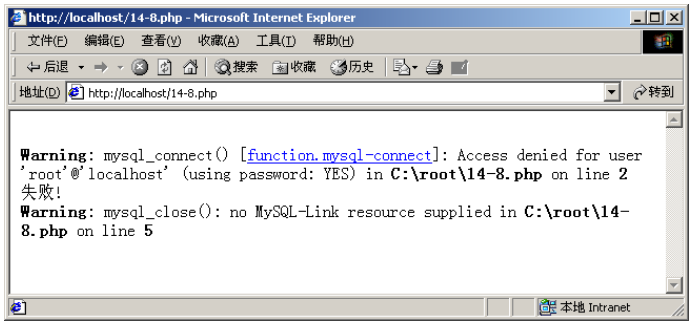

图 14-11　数据库连接

【代码解析】提示用户登录“root@localhost”失败，是否使用密码。提示是，将数据库密码修改为 123456 即可登录成功。

14.4　逻辑错误

逻辑错误是指代码完全正确，而且也是按照正确的程序逻辑执行的，但是结果却不是编程

人员预计的结果。逻辑错误是最难查找的错误类型，下面列举一些较常见的逻辑错误例子，以便以后不再犯类似错误。

14.4.1 计算错误

计算错误如计算结果要保留 float 型的，却整乘整除，从数组读出数据时从 1 开始等，都会带来计算错误，这类错误有时很难查找，要加以注意。

【范例 14-9】利用 for 循环输出 i 值 10 次，其程序如示例代码 14-9 所示。

示例代码 14-9

```
<?php                                          //PHP 开始标记
    $times=10;                                 //变量赋值
    for($i=1;$i<$times;$i++)                   //利用 for 循环
    {
    echo  "循环第".$i."次";                    //输出变量 i 的值
    }
?>                                             //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-9.php，查看其运行结果，即可看到结果如图 14-12 所示。

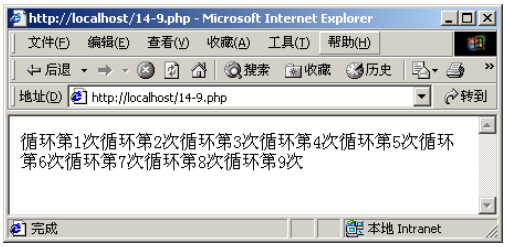

图 14-12　计算错误

【代码解析】上述代码编程人员本想让循环输出 10 次，但是结果只输出 9 次就结束了。原因很简单，变量$i 从 01 到 09 都符合限制条件“$i<times”，而到 10 时则不符合了，就停止了循环，因此总共输出了 9 次。

14.4.2 不测试返回值

在 PHP 编程中，调用有返回值的函数的时候，应尽量先判断其返回值，或者说要提高警惕。逻辑错误很多时候都是由于没有测试返回值造成的。

在 PHP 中，许多最常用的函数都是依赖于最初级的错误报告系统。如果函数工作正常，则会返回一些有用的数据，而如果失败，一般会返回 false。但是有时候正确的返回值也会导致最后结果的错误，下面通过示例查看其情况。

【范例 14-10】分别将 abc、a 赋给两个变量，然后在 abc 中查找是否含有字符 a，分别输出提示信息，其程序如示例代码 14-10 所示。

示例代码 14-10

```
<?php                                          //PHP 开始标记
    $mystring = 'abc';                         //变量赋值
    $findme = 'a';                             //变量赋值
    $pos = strpos($mystring, $findme);         //查找字符串
    if ($pos == false) {                       //判断返回结果
```

```
06        echo " 在 $mystring 中没有找到字符 $findme";                    //输出判断情况
07        }
08        else {                                                          //找到的情况
09            echo " 在 $mystring 中找到了字符 $findme";                  //提示找到信息
10        }
11    ?>                                                                  //PHP 结束标记
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-10.php，查看其运行结果，即可看到结果如图 14-13 所示。

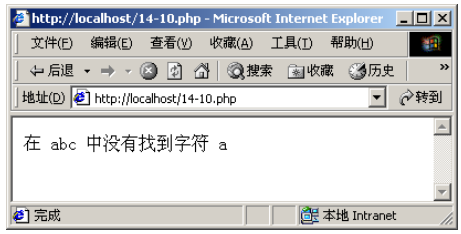

图 14-13　测试返回值错误

【代码解析】代码第 04 行在查找字符串 abc 中查找 a 是否出现，如果找到则返回出现位置，否则返回 flase。第 05 行判断查找结果，从结果可以看出，返回值测试错误，因为 a 的位置为 0，返回 0 则被转换为 flase，改正代码只须将运算符“==”改为“===”。

提示：逻辑错误只有通过适当的调试才能检测到。在调试的时候，首先要消除暗含在脚本中的假设，然后彻底测试每种可能有效或者无效的输入，确认这些输入都得到正确的结果。

14.5　print 语句调试

调试程序是将错误的代码进行查找、更正的过程，调试程序使用 die 和 print 语句是一个较好的选择，特别是对查找逻辑错误。下面通过事例讲解通过 print 语句调试程序的过程。

假设正在处理通过 GET 请求发送过来的表单数据，向浏览器显示信息，但是出于某种原因，数据没有正确地提交，或者不能正确地从 GET 请求中读出。要调试这类问题，重要的是用 print()或 die()语句知道变量的值是什么。

【范例 14-11】使用 get 提交所填写，显示提交的信息和提交的方式，其程序如示例代码如 14-11 所示。

示例代码 14-11

```
01  <?php                                                               //PHP 开始标记
02  $j = "";                                                            //变量赋值
03  print("Lets retrieve all the variables submitted to this.<br> ");
                                                                        //输出字符串
04  print("script via a GET request:<br>");                             //输出
05  foreach($_GET as $key => $i){                                       //遍历
06  print("$key=$j<br>");                                               //输出 j
07  }
08  if($_GET['Submit'] == "Send GET Request")                           //判断
09  $j = "done!<br>";                                                   //变量赋值
10  ?>                                                                  //PHP 结束标记
```

```
11  <form method="GET">                                          //表单开始
12    Name: <input name="name"><br>                          //Name 输入框
13  Email: <input name="email" ><p>                              // E-mail 输入框
14  <input name="Submit" type="submit" value="Send Request">     //提交重置按钮
15  </form>                                                      //表单结束
```

【运行结果】打开 IE 浏览器，在地址栏输入 http://localhost/14-11.php，查看其运行结果，即可看到结果如图 14-14 所示。

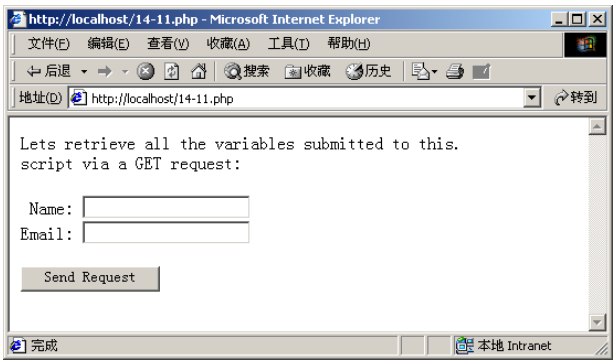

图 14-14　表单

【代码解析】代码提供了一个表单，用 GET 请求向服务器发送变量以进行测试，这个脚本只是提取 GET 请求中的所有变量，如果有，就把它们显示在浏览器上。现在填写用户名为 monkey，E-mail 为 monkey@monkey.com，单击“Send Request”按钮，显示如图 14-15 所示的页面，只有$_GET 请求的键显示在浏览器上，而正确的值都没显示。

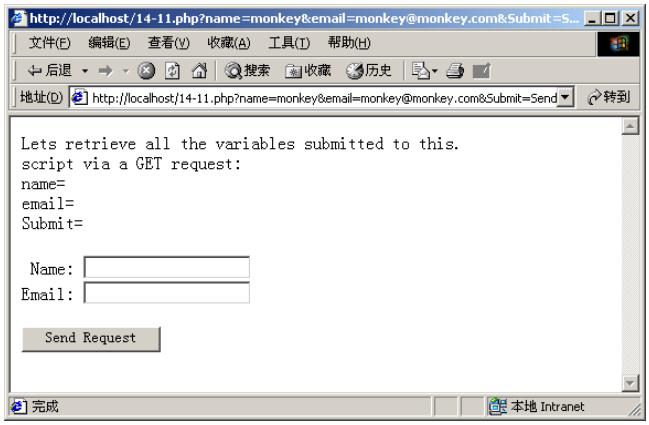

图 14-15　GET 提交

注意：这是一个非常简单的脚本，只是作为使用 print 语句进行调试而展示的一个例子而已。在循环中放一个 print 语句，检验在 foreach 循环中每个元素中是否确实存在数据。

用 print 语句验证代码的功能，在示例代码 14-11 的第 05 行和第 06 行加入代码，保存原来文件。

```
print("Correct data? " . $_GET[$key] ."<br>");
```

加入上述代码的执行结果如图 14-16 所示。

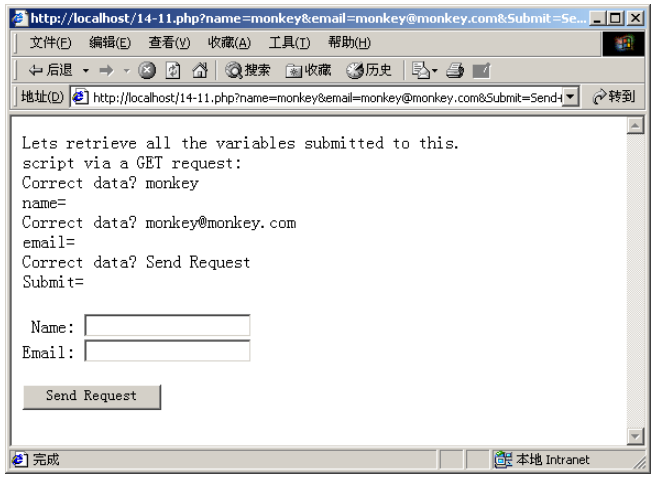

图 14-16　print 语句调试

现在已经知道在 Web 浏览器上显示的$key 值是正确的，但是由于某些原因，值没有正确地显示。应用程序正确地从 GET 请求接收到了变量，那么肯定是在代码中有错误。查看之后注意到，用来显示值的变量$j 是错误的。在 foreach 语句中指定的是$i，所以肯定会有正确的值，但是无意之中输入了$j。所以通过把$j 替换成$i，迅速地修正了错误，重新载入页面之后，就看到了正确的输出。

14.6　综合练习

1．使用调试下列程序，使其输出正确。

```
01  <?php
02          $pi=3.14;
03          for($r=1;$r<=5;$r++){
04          $area=$pi*$r*$r;
05          if($area<50);
06              echo $area."<br>";
07      }
08  ?>
```

调试过程：运行结果发现，输出的数为半径不大于 5 的所有面积，则初步判断 if 没有加以判断。将 IF 语句注释掉，可以看到程序结果无变化，则可断定 if 语句的问题。检查发现，括号后面多一个分号，使其变为独立的语句，同后面 echo 语句无联系，把分号去掉即可得出正确的结论。

2．下列程序无法判断其执行结果，试将其改为能够测试其程序执行的程序。

```
01  <?php
02      $host="localhost";
03      $user="root";
04      $password="123456";
05      mysql_connect($host,$user,$password);
06  ?>
```

提示，程序执行成功与否，做出判断。因为程序无输出，这种情况一般将其执行结果赋给一个变量，然后输出变量的值，另外可通过 die 语句输出其程序错误的情况。将上述代码改为：

```
<?php
    $host="localhost";
    $user="root";
    $password="123456";
    mysql_connect($host,$user,$password)ordie("服务器连接失败");
    echo  "成功";
?>
```

14.7 小结

本章主要介绍了几种常见的错误类型和一些常用的方法。错误调试是编写程序中经常会遇到的，正确的调试方法，会使程序员轻松发现错误。当然这还需要在今后的调试中认真体会，需要大量的实践经验才能很好地掌握。

如果读者对本章内容还有疑问，可参考《精通 PHP 5 应用开发》（秦涛，曾文玉．北京：人民邮电出版社，2007）和《PHP 技术内幕》（作者：Peter Moulding．译者：张帆、贺民．北京：中国水利水电出版社，2003）。

14.8 习题

一、填空题

1．使用_____函数可以定义当前 PHP 页面中错误消息的显示级别。

2．在 php.ini 中，以一个_____表示注释。

3．php.ini 默认的输出方式中，error_reporting = E_ALL 表示_____。

4．常见的错误类型有_____、_____、_____。

5．常用的错误调试输出语句是_____、_____。

6．语句（echo "i love php;）属于_____错误。

7．调用不存在的文件属于_____错误。

8．逻辑错误是代码_____，而且也是按照正确的程序逻辑执行的，但是结果_____。

二、选择题

1．指令 E_ALL 的含义是（　　）。

A．运行时的警告　　B．编译时解析错误

C．所有的错误和警告　　D．用户错误信息

2．如果文件 foo.txt 存在，则程序的执行结果正确的是（　　）。

```
$fp=fopen("foo.txt"," r");
fwrite($fp,"hello world");
fclose($fp);
```

A．程序提示错误　　B．程序无任何输出

C．文字被写入　　D．程序无任何错误

3．下面程序的错误在于（　　）。

```
01  $mystring = 'php';
02  $findme = 'a';
03  $pos = strpos($mystring, $findme);
04  if ($pos == false) {
05  echo " 在 $mystring 中没有找到字符 $findme";
```

```
06  }
07  else {
08       echo " 在 $mystring 中找到了字符 $findme";
09  }
```

A．测试返回值错误　　B．缺少引号

C．缺少分号　　D．缺少关键字

4．选择下面程序的运行结果（　　）。

```
01    for($=1;i<10;$i++){
02     echo "$i";
03   }
```

A．输出 10 次 i 的值　　B．输出 9 次 i 的值

C．不输出　　D．产生死循环

三、简答题

1．简述错误调试的方法。

2．列出常见错误类型。

四、编程题

1．调试下列程序，使其有输出。

```
<?php
    $int=1;
    function fun1(){
        $int2=2;
        echo  "$int1<br>";
        }
    fun1();
    echo "$int2<br>";
?>
```

2．调试下列程序，使其输出正确的结果。

```
01  <?php
02       $score=73;
03            if ($score >=80){
04                 echo  "成绩优秀";
05            }
06            else
07  if ($score>=60){
08                 echo "及格了";
09            }
10
11  if($score>=30){
12            echo "没有通过考试!";
13       }
14       else
15       echo "成绩有误";
16  ?>
```