精品课程配套教材
21 世纪应用型人才培养“十三五”规划教材
“双创”型人才培养优秀教材

PHP 网站开发教程

莫兴福　阚钿玉　主编

西北工业大学出版社
西　安

目　录

第 1 章

PHP 简介与开发工具

【本章内容简介】

PHP 是一门服务器端程序设计语言。除 PHP 外，还有 JSP，ASP，ASP. NET 等重要的服务器端程序设计语言。服务器端程序设计语言主要运行在服务器端，用于处理来自浏览器端的客户请求；服务器端程序在客户请求处理完成之后，将处理结果返回浏览器端，供用户在浏览器端查看或进行下一步交互。

PHP 语言最强大和最重要的特征就是跨平台和面向对象。本章首先介绍 PHP 语言的发展史、功能、特点和发展趋势等，其次介绍 PHP 语言的开发环境配置，接着编写一个简单的 PHP 程序，最后介绍开发过程中遇到错误如何调试和处理。通过本章的学习，读者能够对 PHP 语言有一个整体上的认识，为后期学习 PHP 语言的具体内容打下良好的基础。

【本章学习要求】

通过本章的学习，要求了解 PHP 语言；能够搭建 PHP 语言的开发环境；能尝试编写简单的 PHP 程序，以及遇到一般错误能进行处理。

1.1 PHP 语言简介

PHP（Hypertext Preprocessor，超文本预处理器）是一种被广泛应用的开放源代码的多用途脚本语言，它可嵌入 HTML 中，尤其适合动态网站的开发。PHP 吸收了 C，Java 和 Perl 语言的特点，利于学习，使用广泛，主要适用于 Web 开发领域。

1.1.1 PHP 发展史

PHP 最初是由丹麦的 Rasmus Lerdorf 创建的，刚开始它只是一个简单的、用 Perl 语言编写的程序，用来统计网站的访问量，后来又用 C 语言重新编写，添加了访问数据库的功能。1995 年初，Lerdorf 以 Personal Home Page Tools（PHP Tools）开始对外发布第一个版本，即 PHP1. 0。

1995 年 6 月，PHP2.0 发布，定名为 PHP/FI。PHP/FI 增加了对 MySQL 的支持。到 1996 年底，有 15 000 个网站使用 PHP/FI。1997 年，使用 PHP/FI 的网站数量超过 5 万个。同年，开始了 PHP 第 3 版的开发计划，开发小组加入了 Zeev Suraski 及 Andi Gutmans，1998 年 6 月，PHP3.0 诞生。

2000 年 5 月，PHP4.0 发布。该版本将语言和 Web 服务器之间的层次抽象化，并且加入了线程安全机制，加入了更先进的两阶段解析与执行标签解析系统。这个新的解析程序依然由 Zeev Suraski 和 Andi Gutmans 编写，并且命名为 Zend 引擎。

2004 年 7 月，PHP5.0 发布，标志着一个全新的 PHP 时代到来。PHP5.0 引入了面向对象的全部机制，保留了向下的兼容性，并且引进了类型提示和异常处理机制，能更有效地避免和处理错误的发生。直到 2008 年，PHP5.0 成为 PHP 唯一维护中的稳定版本。

2013 年 6 月，PHP5.5 发布。此版本包含大量的新功能和 bug 修复。开发者需要特别注意的是，该版本不再支持 Windows XP 和 Windows 2003 操作系统。

PHP 进入中国较晚，在很长一段时间内，PHP 在中国的使用率很低。但是最近几年，PHP 以其易学、高效、安全、免费、跨平台等一系列重要优势，在国内迅速发展起来。现在，国内已经出现大量采用 PHP 开发的网站。

1.1.2 PHP 语言特点

PHP 作为一种服务器端的脚本语言，其主要有以下特点。

1. 开放源代码

PHP 属于自由软件，是完全免费的，用户可以从 PHP 官方站点（http://www.php.net）自由下载，而且可以不受限制地获得源码，甚至可以在源码中加进自己需要的代码。

2. 基于服务端

PHP 是运行在服务器上的，充分利用了服务器的性能，PHP 的运行速度只与服务器的速度有关，因此它的运行速度可以非常快；PHP 执行引擎还会将用户经常访问的 PHP 程序驻留在内存中，其他用户再一次访问这个程序时就不需要重新编译了，只要直接执行内存中的代码即可，这也是 PHP 高效性的体现之一。

3. 数据库支持

PHP 能够支持目前绝大多数的数据库，如 DB2，dBase，mSQL，MySQL，Microsoft SQL Server，Sybase，Oracle，Oracle 8，PostgreSQL 等，并完全支持 ODBC（Open Database Connectivity，开放数据库连接），因此可以连接任何支持该标准的数据库。其中，PHP 与 MySQL 是绝佳的组合，它们的组合可以跨平台运行。

4. 跨平台

PHP 可以在目前所有主流的操作系统上运行，包括 Linux，UNIX 的各种变种，Microsoft Windows，Mac OS X，RISC OS 等。另外，PHP 支持大多数的 Web 服务器，包括

Apache，IIS，iPlanet，Personal Web Server（PWS），Oreilly Website Pro Server 等。对于大多数服务器，PHP 均提供有相应的模块。

5. 易于学习

PHP 的语法接近 C，Java 和 Perl，学习起来非常简单，而且拥有很多学习资料。PHP 还提供了数量巨大的系统函数集，用户只要调用一个函数就可以完成很复杂的功能，编程时十分方便。因此用户只需要很少的编程知识就能够使用 PHP 建立一个交互的 Web 站点。

6. 网络应用

PHP 提供了强大的网络应用功能，支持诸如 LDAP，IMAP，SNMP，NNTP，POP3，HTTP，COM（Windows 环境）等协议服务。PHP 可以开放原始端口，使任何其他的协议能够协同工作。利用 PHP 还可以编写发送电子邮件、FTP 上传/下载等网络应用程序。

7. 安全性

由于 PHP 本身的代码开放，所以它的代码由许多工程师进行了检测，同时它与 Apache 编译在一起的方式也让它具有灵活的安全设定。因此到现在为止，PHP 具有公认的安全性。

8. 其他特性

PHP 还提供其他编程语言所能提供的功能，如数字运算、时间处理、文件系统、字符串处理等。除此之外，PHP 还提供了更多的支持，包括高精度计算、公元历转换、图形处理、编码与解码、压缩文件处理以及有效的文本处理功能（如正则表达式、XML 解析等）。

1.1.3　PHP 语言发展趋势

当前，PHP 的发展相当迅速，尤其在国内。

- 全球 5000 万互联网站中，有 60% 以上使用 PHP 技术。
- 国内 80% 以上的动态网站使用 PHP 开发。
- 394 家 AlexaTOP500 中国网站采用 PHP 技术。
- PHP 人才需求比达到 1：30，而供求比达到 1：40。
- PHP 成为中国应用最广的网页编程语言。

PHP 作为非常优秀的、简便的 Web 开发语言，和 Linux，Apache，MySQL 紧密结合，形成 LAMP 的开源黄金组合，不仅降低了使用成本，还提升了开发速度，满足了新的互动式网络开发的应用。

在与其他同类编程语言的比较中，PHP 具有开发速度快、运行效率高、安全性好、可扩展性强、开源自由等特点。

回顾 PHP 近 10 年来的发展过程，展望未来互联网行业的发展趋势，我们可以得出结论：PHP 的发展势头不可阻挡，必将成为未来 Web 开发领域的主流技术体系。

1.2 PHP 开发环境配置

PHP 是一门服务器端程序设计语言，执行 PHP 程序时需要在 Web 服务器上架设一个编译 PHP 网页的引擎。配置 PHP 开发环境的方法很多，但主要工作就是安装和配置 Web 服务器和 PHP 引擎。Apache 是目前比较流行的支持 PHP 运行的 Web 服务器。

1.2.1 开发环境的选择

动态网站服务器平台至少要包括操作系统、Web 服务器、应用程序服务和数据库。好的动态网站服务器是由多个方面因素决定的，如个人喜好、部署费用、安全机制等。目前，3 种主流的 Web 平台分别是 LAMP/WAMP，J2EE 和 ASP. NET，它们的运行环境组合如下：

LAMP：Linux+Apache+MySQL+PHP。

J2EE：UNIX+Tomcat+Oracle+JSP。

ASP. NET：Windows Server+IIS+SQL Server+ASP. NET。

PHP 开发环境涉及操作系统、Web 服务器和数据库。WAMP 是 PHP 开发的一种常用技术环境组合。所谓 WAMP 就是基于 Windows，Apache，MySQL 和 PHP 的运行环境，WAMP 的名字来源于这些软件名称的第一个字母。

1. Apache 服务器

Apache 是一款开放源码的 Web 服务器，其平台无关性使得 Apache 服务器可以在任何操作系统上运行，包括 Windows。强大的安全性和其他优势，使得 Apache 服务器即使运行在 Windows 操作系统上也可以与 IIS 服务器媲美，甚至在某些功能上远远超过了 IIS 服务器。在目前所有的 Web 服务器软件中，Apache 服务器以绝对优势占据了市场份额的 70%，遥遥领先于排名第二位的 Microsoft IIS 服务器。

2. MySQL 数据库

MySQL 是一个开放源码的小型关系数据库管理系统，由于其具有所占空间小、速度快、总体成本低等优点，目前广泛应用于 Internet 的中小型网站中。MySQL 是一个真正的多用户、多线程的 SQL 数据库服务器。由于 MySQL 源代码的开放性和稳定性，并且可与 PHP 完美结合，因此很多站点使用它进行 Web 开发。

3. PHP 脚本语言

PHP 是一种新型的 CGI 程序编写语言，易学易用，运行速度快，可以方便快捷地编写出功能强大，运行速度快，并可同时运行于 Windows，Unix，Linux 平台的 Web 后台程序，内置了对文件上传、密码认证、Cookies 操作、邮件收发、动态 GIF 生成等功能，PHP3. 0 直接为很多数据库提供原本的连接，包括 Oracle，Sybase，Postgres，MySQL，Informix，Dbase，Solid，Access 等，完全支持 ODBC 接口，用户更换平台时，无须变换 PHP3. 0 代码，可即拿即用。PHP3. 0 Server 完全免费。PHP4. 0 提供了更多 Web Server 的支持，例如 Zeus，Netscape Enterprise，Apache Win32 module，并增加了很多性能，修正了很多漏洞。

PHP7. 0 的最大特色是极大地改进了性能，在一些 WordPress 基准测试当中，性能可以达到 PHP5. 6 的 3 倍。

1. 2. 2　Apache 安装与配置

Apache 是世界上使用最广泛的 Web 服务器之一，根据 NetCraft 机构的调查，世界上 50%以上的 Web 服务器是用 Apache 搭建的。

1995 年 4 月，最早的 Apache0. 6. 2 版由 Apache Group 公布发行。Apache Group 是一个非盈利的机构，因此，Apache 版本的更新以及标准发行版中应包含的内容等日常运作都是在互联网上进行的。Apache 是一款开放源代码软件，允许任何人对其进行修改、扩充和更新，正因为如此，它才保持了强大的生命力。

和其他服务器相比，Apache 拥有以下特性：

- 几乎所有的计算机平台都支持 Apache 运行。
- 支持 HTTP 协议的最新版本 HTTP/1. 1。
- 所有的服务器其配置操作都可以通过 httpd. conf 文件进行，操作方便简单。
- 支持通用网关接口（CGI）、快速网关接口（FASTCGI），支持虚拟主机，支持 HTTP 认证。
- 具有对用户会话过程的跟踪能力。
- 支持 Java Serverlets。
- 运行效率高，成本低。

Apache 是免费软件，因此获取 Apache 安装文件的途径很多，但最好从其官方网站上下载。

1. 下载 Apache

Apache 服务器官方网站的地址是 http：//httpd. apache. org/，这里我们以下载稳定版的 httpd 2. 4. 38 为例，在 Apache 服务器官方网站主页点击左侧菜单选项栏的“Download”，如图 1-1 所示。

图 1-1　Apache 的官方网站主页

在弹出来的新页面上，找到“a number of third party vendors”点击链接，如图1-2所示。

图 1-2　Apache 服务产品列表

Apache 本身不提供已编译的安装包，只提供源码，这里我们选择官方推荐的第三方提供编译的网站，如图 1-3 所示。

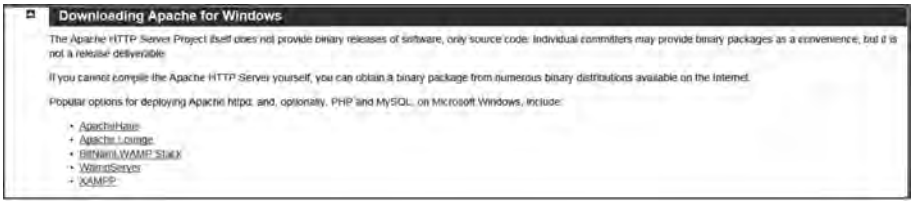


图 1-3　第三方编译网站

其中，后两个是有名的 Wamp 以及 XAMPP 集成环境，如果只想下载 Apache，可以选择前三个网站，这里我们以第一个网站 ApacheHaus 为例。打开 ApacheHaus 之后，会发现这个网站上有各种 Windows 版本，可以根据需要选择要下载的版本。点击红框中的图标即可开始下载，x86 是 32 位的，x64 是 64 位的，用户可根据自己的操作系统选择下载，如图 1-4 所示。

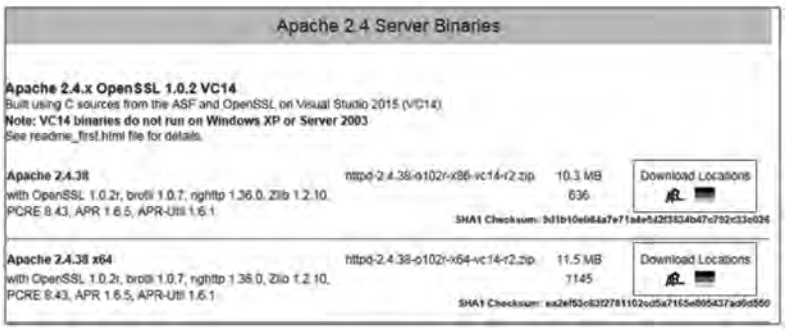


图 1-4　Apache 服务器安装文件的下载页面

选择合适的 Apache 版本下载，如图 1-5 所示。

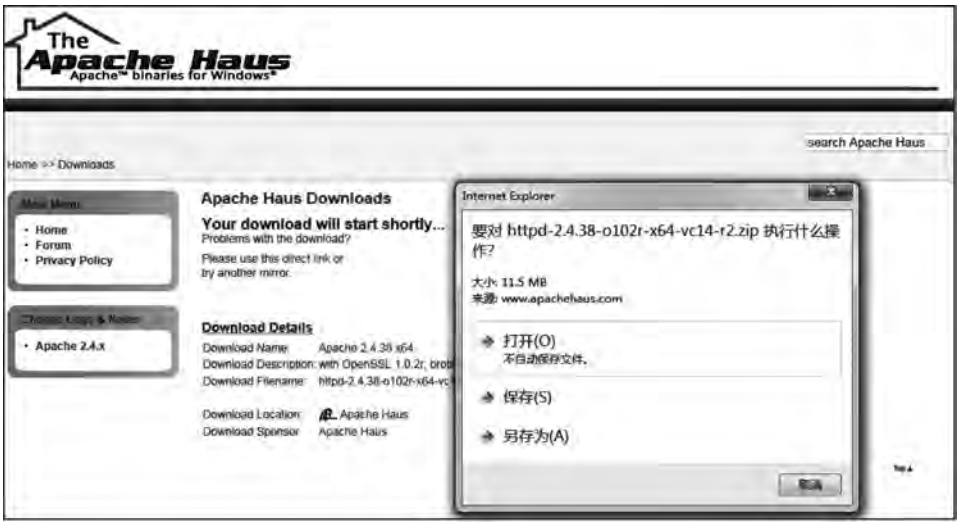


图 1-5　下载 Apache 服务器

下载到本地硬盘后，解压文件到任意盘，这里以解压到 D 盘根目录为例，如图1-6所示。

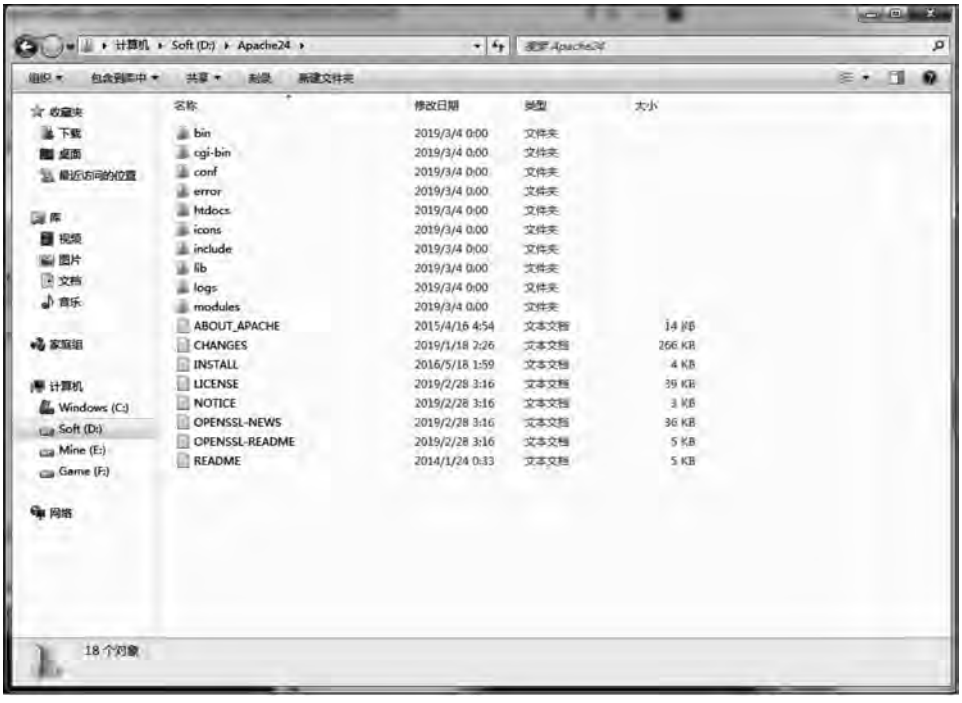


图 1-6　解压 Apache 服务器到 D 盘

2. 安装 Apache

在 Windows“开始”菜单里，找到“命令提示符”，单击右键，在弹出的菜单中选择“以管理员身份运行”，如图 1-7 所示。

图 1-7　运行命令提示符

在命令提示行下输入“D:”，按回车键确认，进入 D 盘根目录，然后输入“CD D:\Apache24\bin”，按回车键确认，进入 Apache 下面的 bin 目录，输入“httpd -k install”，按回车键确认，把 Apache 安装成 Windows 后台服务，如图 1-8 所示。

图 1-8　命令提示符窗口

3. 修改 httpd. conf 文件

进入 D:\Apache24\conf 目录，打开 httpd. conf 文件，用记事本打开该文件，找到第 39 行 ServerRoot，设置如图 1-9 所示。

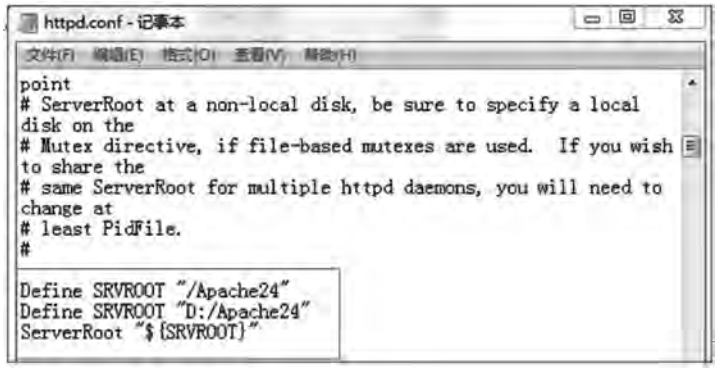


图 1-9　修改 httpd. conf 文件

4. 启动 Apache 服务

在命令提示符下输入“httpd -k start”，按回车键以启动 Apache24 服务，如图 1-10 所示。

图 1-10　启动 Apache 服务

另外，也可以利用 Apache 监控器来启动 Apache 服务。

Apache 监控器是一个用来监控正在运行的 Apache 服务的软件，并且让启动、停止和重启 Apache 服务变得更容易，如图 1-11 所示。只需要双击 \ Apache24 \ bin 目录下的 ApacheMonitor. exe 就可以运行该程序，如果想开机自动启动该软件，把该软件拉到“开始”菜单里的 Startup folder 即可。

图 1-11　利用 Apache 监控器启动 Apache 服务

5. 运行

在浏览器中输入 http：//127.0.0.1 或者 http：//localhost 将显示测试页面，测试 Apache 是否安装成功，如出现如图 1-12 所示的界面，则说明 Apache 安装成功。

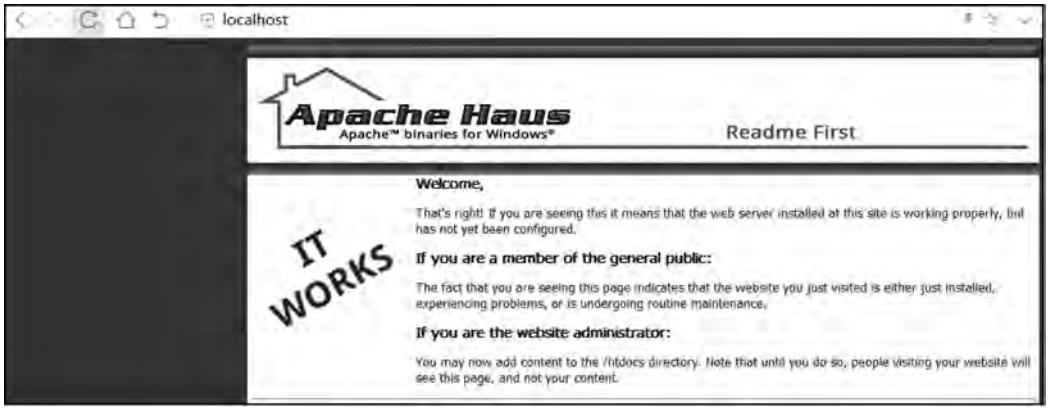


图 1-12　测试 Apache 安装成功界面

1.2.3　PHP 安装与配置

1. 下载 PHP

进入 PHP 服务器官方网站 http：//www.php.net/，这里我们以下载稳定版的 PHP 7.3.2 为例，在页面上找到“windows.php.net/download/”点击链接，如图 1-13 所示。

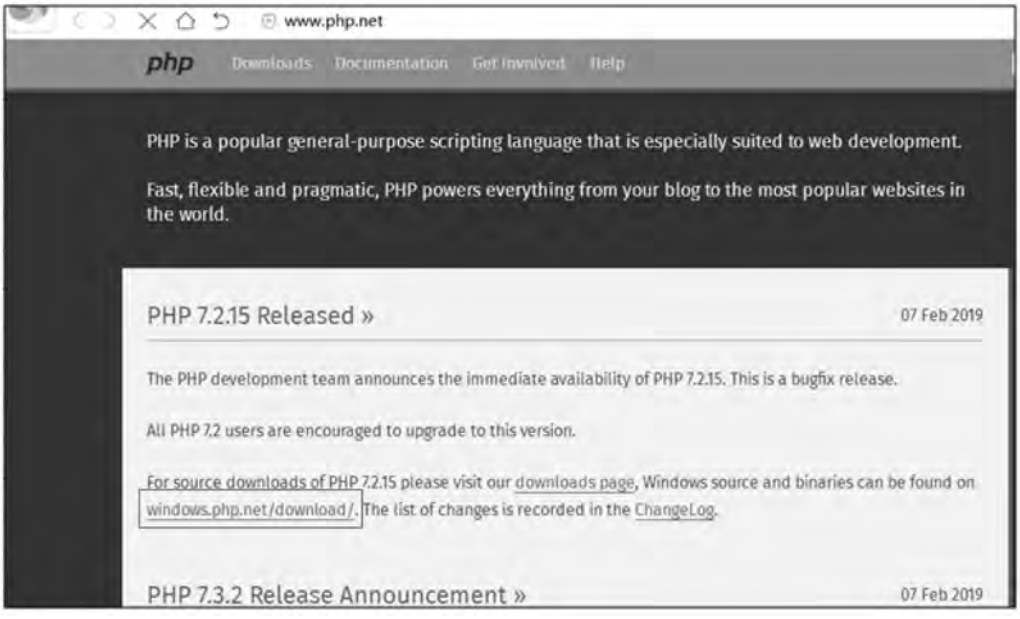


图 1-13　PHP 服务器官方网站

最新的 PHP 有 4 个版本，分别是 32 位的 VC15 x86 Non Thread Safe，VC15 x86 Thread Safe 和 64 位的 VC15 x64 Non Thread Safe，VC15 x64 Thread Safe。Non Thread Safe 版本用于 IIS，这里我们直接选择 Thread Safe 版本。如果操作系统是 64 位的，尽量选择 64 位版本。如图 1－14 所示，点击“Zip”链接，将 64 位版本“php－7. 3. 2－nts－Win32－VC15－x64. zip”下载到硬盘任意分区。

图 1－14　选择适合的 PHP 版本

下载 PHP 安装包之后，将压缩包内所有文件解压到 D 盘根目录，如图 1－15 所示。

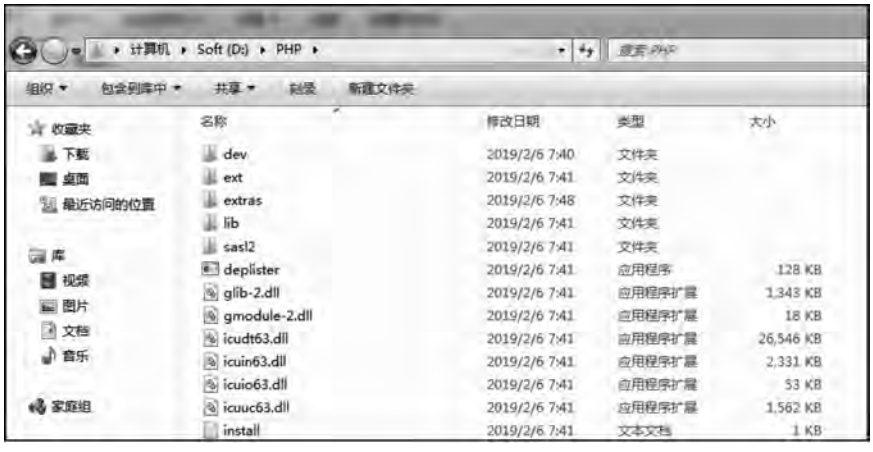


图 1－15　将 PHP 压缩包解压到 D 盘根目录

2. 进行 php. ini 设置

复制一份 D：\ PHP 下的 php. ini－development 文件并重命名为 php. ini 文件，如图 1－16所示。

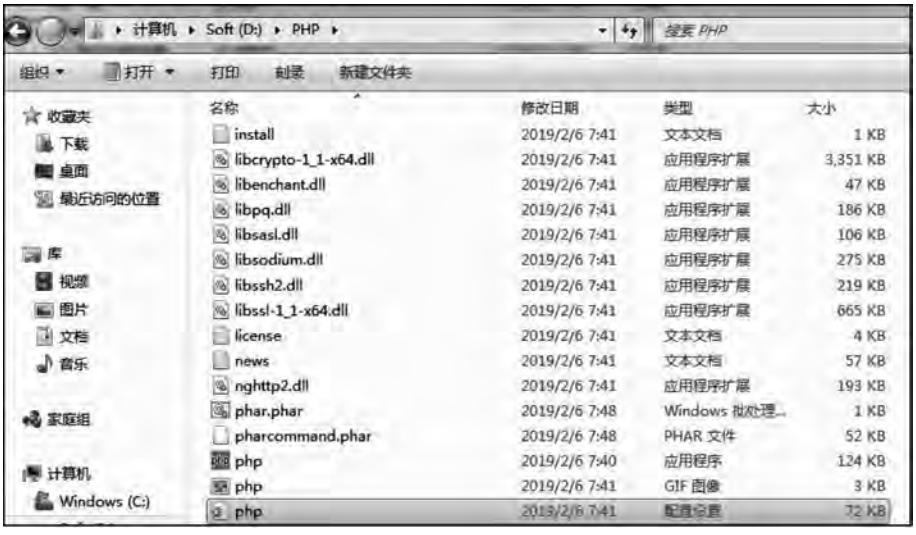


图 1－16　复制 php. ini－development 文件并重命名为 php. ini 文件

用记事本打开 php. ini 文件，查找 extension_dir = "ext"，将其更改为 extension_dir = "D：\ PHP \ ext"，取消此行的注释，并且填写 ext 文件夹的实际路径，否则扩展加载不了，如图 1－17 所示。

```
; Directory in which the loadable extensions (modules) reside.
; http://php.net/extension-dir
;extension_dir = "./"
; On windows:
extension_dir = "D:\PHP\ext"
```

图 1－17　设置 php. ini 文件

设置时区“；date. timezone =”为“date. timezone =PRC”，如图 1－18 所示。

```
;;;;;;;;;;;;;;;;;;;;

[CLI Server]
; Whether the CLI web server uses ANSI color coding in its terminal output.
cli_server.color = On

[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone =PRC
```

图 1－18　继续设置 php. ini 文件

再如图 1-19 所示，去掉文本格式前的“;”符号，开启部分 PHP 的扩展文件。

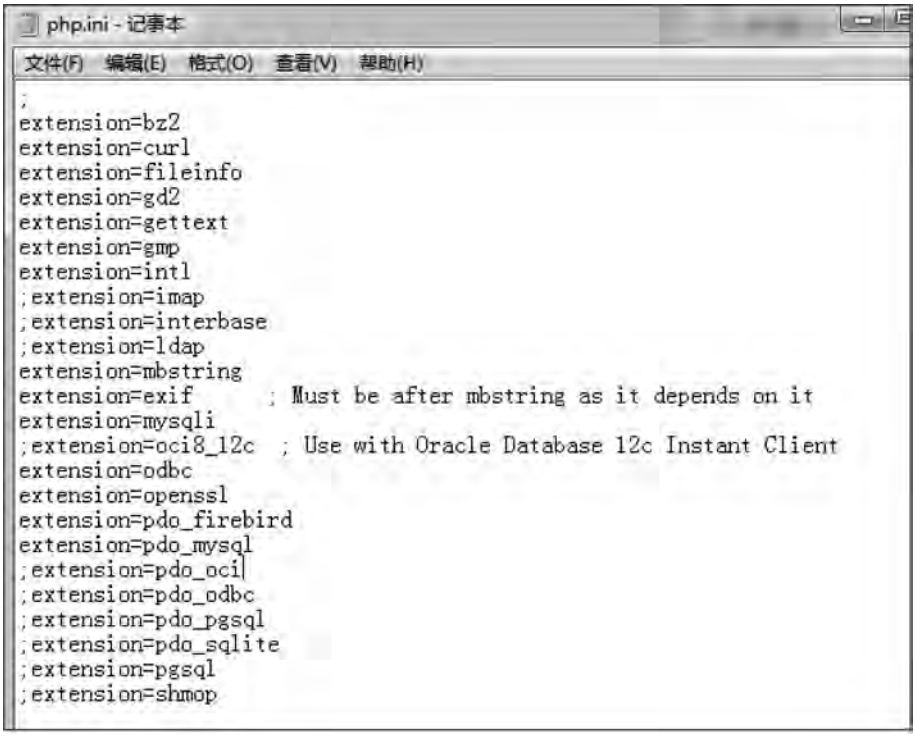


图 1-19　开启部分 PHP 的扩展文件

3. 设置 Apache 的 httpd. conf

用记事本打开 D：\ Apache24 \ conf 下的 httpd. conf 文件，查找#LoadModule xml2enc_module modules/mod_xml2enc. so，在这行下面添加以下内容：

PHPIniDir "D：\ PHP"

LoadModule php7_module "D：\ PHP \ php7apache2_4. dll"

如图 1-20 所示。

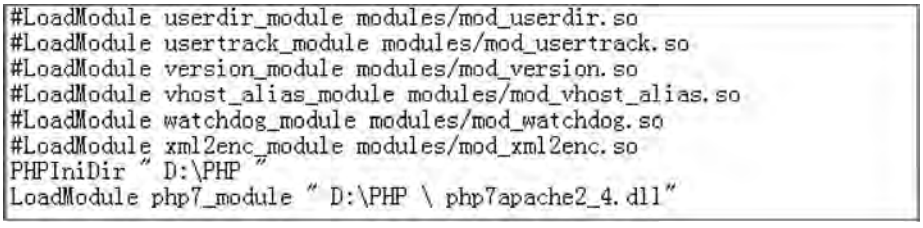


图 1-20　设置 Apache 的 httpd. conf

查找 AddType application/x-gzip . gz . tgz，在这一行下面添加以下内容：

AddType application/x-httpd-php . php

如图 1-21 所示。

```
#
#AddEncoding x-compress .Z
#AddEncoding x-gzip .gz .tgz
#
# If the AddEncoding directives above are commented-out, then you
# probably should define those extensions to indicate media types:
#
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddType application/x-httpd-php .php
```

图 1-21　添加 AddType application/x-httpd-php . php

1. 2. 4　MySQL 的安装与配置

1. 下载 MySQL

进入 MySQL 官方网站下载页面 https：//dev. mysql. com/downloads/mysql/，选择“Windows（x86，64-bit），ZIP Archive”，如图 1-22 所示。点击链接进入下载页面，在弹出来的新页面里点击“No thanks，just start my download”直接下载，并解压至系统 D 盘根目录，如图 1-23 所示。

图 1-22　选择“Windows（x86，64-bit），ZIP Archive”

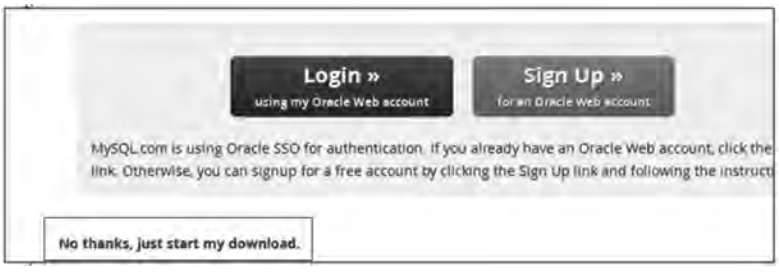


图 1-23　点击“No thanks，just start my download”下载 MySQL

2. 安装配置 MySQL

打开 D：\ mysql 文件夹，在其目录下面新建一个 my. ini 配置文件，输入以下内容：

[mysql]

```
#设置 mysql 客户端默认字符集
default-character-set=utf8
[mysqld]
#设置 3306 端口
port = 3306
#设置 mysql 的安装目录
basedir=D:\mysql
#设置 mysql 数据库的数据的存放目录
datadir=D:\mysql\data
#允许最大连接数
max_connections=200
#服务端使用的字符集默认为 8 比特编码的 latin1 字符集
character-set-server=utf8
#创建新表时将使用的默认存储引擎
default-storage-engine=INNODB
```

在 Windows“开始”菜单里，找到“命令提示符”，单击右键，在弹出的菜单中选择“以管理员身份运行”，使用“CD”命令进入到 mysql 子目录 bin，输入“mysqld --install”进行安装，如图 1-24 所示。

图 1-24　安装 MySQL

输入“mysqld --initialize”创建数据库 data 文件夹，如图 1-25 所示。

然后输入“net start mysql”以启动 MySQL 服务，如图 1-26 所示。

图 1-25　创建数据库 data 文件夹

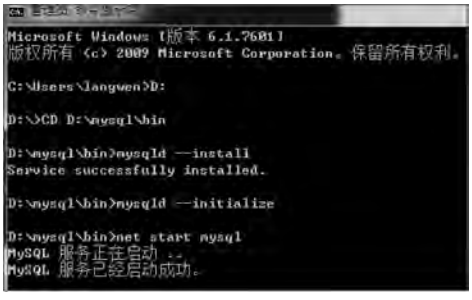


图 1-26　启动 MySQL 服务

3. 创建 MySQL 用户及密码

打开 D：\ mysql \ data 文件夹，用记事本打开后缀为 . err 的文件，查找“root@localhost:”，其后即为系统生成的初始密码，在命令提示符下输入：mysql -u root -p，然后按提示输入 . err 的文件查找到的密码，连接到 MySQL，如图 1-27 所示。

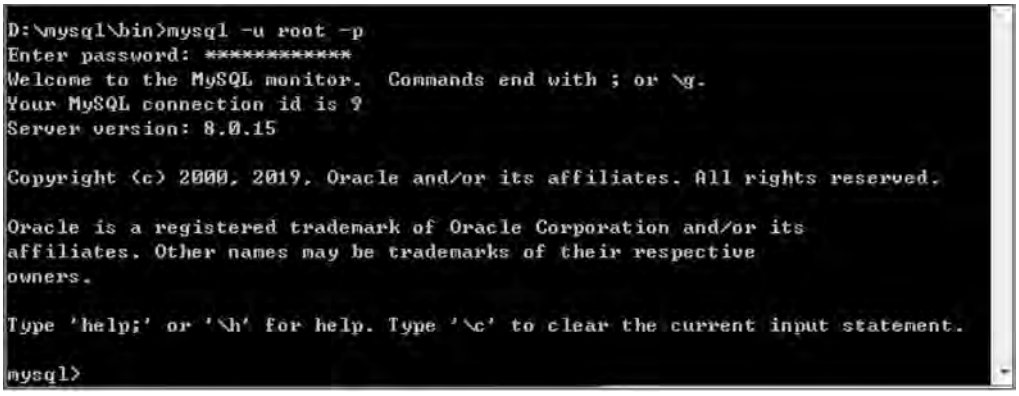


图 1-27　连接 MySQL

另外，输入命令：ALTER USER ' root' @ ' localhost' IDENTIFIED WITH mysql_native_password BY ' 你的密码' ，可以任意修改为要设定的密码。

1.2.5　开发工具的选择

PHP 有多种开发工具，常用的开发工具通常分三类：一类是简单文本编辑器类型，如 EditPlus，Notepad++；一类是专门的 PHP 开发编辑器，如 phpDesigner8，PHP Coder，Zend Studio，PHP Editor；还有一类是在通用开发编辑器上通过嵌入 PHP 插件支持形成的 PHP 开发编辑器，如 NetBeansIDE，Eclipse PDT 等。每种开发工具各有优势，好的开发工具可以提升开发效率，开发人员可以根据需要进行选择。这里不对编辑器做过多介绍，读者可搜索相应的开发工具，查看工具介绍，找一款适合自己需要的开发编辑器。

1.2.6　Eclipse 安装与配置

1. 下载 Eclipse

进入 Eclipse 官方网站下载页面 https：//www. eclipse. org/pdt/（见图 1-28），点击“Windows”链接进入下载页面下载安装文件，并解压至系统 D 盘根目录，无须安装。

图 1-28　Eclipse 官方网站下载页面

2. 下载、安装 Java SE Runtime Environment

由于 Eclipse 需要 Java SE Runtime Environment 组件的支持，因此需要进入其官方网站：https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html，选择同意协议后进入下载页面，选择合适的版本下载，如图 1-29 所示。组件安装十分简单，按提示即可快速完成，这里不再赘述。

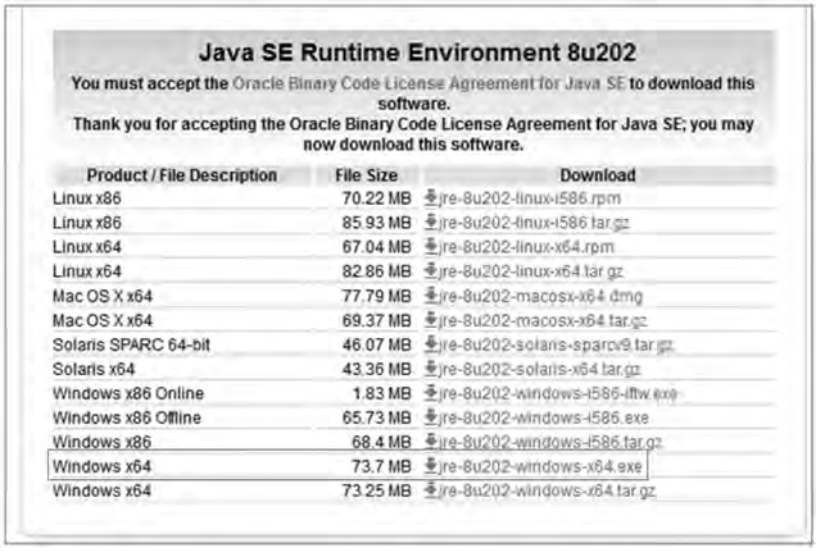


图 1-29　下载 Eclipse

1.2.7 简单 PHP 页面实例

【例 1.2-1】本例的目的是熟悉 PHP 的书写规则和程序的执行。本例将输出一行“Hello World”。

程序代码：

```
<? php
    echo "Hello World";
? >
```

程序运行结果如图 1-30 所示。

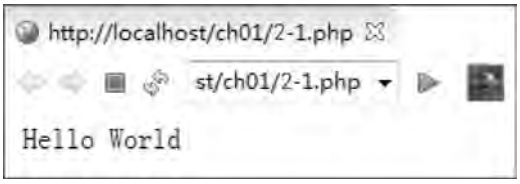


图 1-30 运行结果

程序说明：

“<? php”和“? >”是 PHP 的标记对。在这标记对中，所有的代码都被当成 PHP 代码处理。除了这种方法外，PHP 还可以使用 ASP 风格的“<% … % >”和 SGML 风格的“<? …? >”等。

echo 是 PHP 中的输出语句，与 ASP 中的 response. write，JSP 中的 out. print 含义相同，用于将紧跟其后的字符串或变量值显示在页面中。每行代码都以分号“;”结尾。

www 目录是 WAMP 服务器默认存放网站的位置，在这个示例中，PHP 将会默认到这个位置的 ch01 文件夹中查找 2-1. php 文件，然后进行解析、输出。

1.3 综 合 案 例

1.3.1 基本能力实训项目

1. 实训题目

配置 PHP 动态网站开发所需的环境。

2. 实训目的

掌握 PHP 动态网站开发所需的软件安装和环境配置。

3. 实训内容

（1）下载合适版本的 PHP 安装程序（选择 Windows 安装版），然后进行关联。

（2）通过对 Apache 和 PHP 进行相关的配置，实现 PHP 与 Apache 的关联。

（3）在 HTML 标记语言的 BODY 标记部分嵌入代码：

```
<? php
    echo phpinfo( );
? >
```

以测试 PHP 程序开发环境是否组建成功并查看相关配置信息。

1.3.2　拓展能力实训项目

1. 实训题目

编写一个简单的 PHP 程序。

2. 实训目的

熟悉 PHP 开发环境。

3. 实训内容

利用 PHP 开发环境，输出“Hello World”。

本 章 小 结

在网站开发过程中，客户端需要用到 HTML，CSS，JavaScript 语言，而服务器端程序语言一般采用 PHP，JSP，ASP 等。PHP 以简单、高效著称。

PHP 是一种服务器端、跨平台、HTML 嵌入式的脚本语言，其独特的语法混合了 C，Java 和 Perl 语言的特点，是一种被广泛应用的、开源的多用途脚本语言，尤其适合 Web 开发。PHP 语言的优点有：安全性高、跨平台、易学、执行速度快、免费、模块化、支持面向对象与过程、内嵌 Zend 加速引擎、性能稳定。

PHP 适合用来开发任何应用类型的服务器端程序，可应用于各行各业。在编写 PHP 程序时，如果程序不多，简单的记事本即可满足编辑要求；如果是大型项目，也有许多开源免费的 IDE 平台可供选择。另外，在学习 PHP 的过程中，有大量的 PHP 图书、网站、技术论坛等辅助学习渠道。

本章最后还介绍了 PHP 开发环境配置，以及如何编写一个简单的 PHP 程序。

习　　题

1. 常用的 PHP 开发工具有哪些？
2. 尝试在 Windows 上部署 PHP 运行环境。
3. 将 Apache 服务器的主目录设置为 D：\ wgzx，并运行一个该目录中的 PHP 文件。
4. 编写一个简单的 PHP 程序，并在浏览器中执行。

第 2 章

PHP 基本语法

【本章内容简介】

语法是学习一门语言的基础，PHP 语法与 C 语言语法相似，如果读者有 C 语言的编程经验，那么学习 PHP 语法就很容易。本章主要介绍 PHP 语言基础、数据类型、变量与常量、运算符与表达式等内容。

【本章学习要求】

通过本章的学习，要求掌握 PHP 语言标记，以及语法和注释的相关知识，掌握 PHP 的几种数据类型，掌握 PHP 变量和常量的相关知识，掌握 PHP 运算符和表达式的相关知识和应用。

2.1 PHP 入门

PHP（Hypertext Preprocessor，超文本预处理器）是一种通用开源脚本语言，利于学习，使用广泛，主要适用于 Web 开发领域。PHP 代码嵌入在 HTML 代码中，通过一定的标记来区分 HTML 代码、客户端和服务器端代码。

2.1.1 第一个 PHP 脚本程序

下面用 Phpstudy+Eclipse 的开发环境来写第一个 PHP 脚本程序。打开 Phpstudy，启动 Apache 和 MySQL 服务器，如图 2-1 所示；打开 Eclipse，创建项目，名称为“chapter2”，在项目中新建 PHP 文件 first.php，编写以下代码：

```
<! DOCTYPE html>
<html>
<head>
    <title>PHP 程序</title>
</head>
<body>
    <? php
```

```
        echo "<h1>你好,这是第一个 PHP 脚本程序! </h1>";
    ? >
</body>
</html>
```

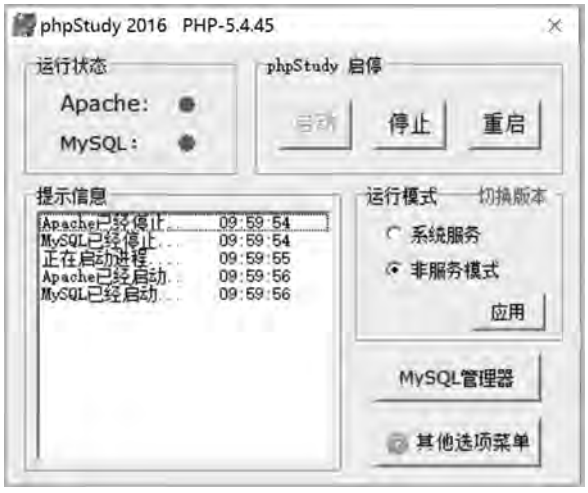


图 2-1　Phpstudy 启动界面

说明：程序中“<? php”和“? >”分别是 PHP 语言的开始符和结束符，在这两个符号之间的代码为 PHP 代码，其余的是 HTML 代码，这里是将 PHP 嵌入 HTML 代码中。而在 PHP 代码中“<h1>”和“</h1>”标记是 PHP 代码中加入 HTML 标记，这里采用最新的 HTML5。

代码编写完成后单击面板上的图标进行保存，下方的“PHP Browser”窗口将会出现运行结果，如图 2-2 所示。也可以选择直接在浏览器中运行程序。打开 IE，在地址栏中输入 http：//localhost/chapter2/first. php，单击【Enter】键，出现如图 2-3 所示的运行结果。

图 2-2　Eclipse 中运行结果

图 2-3　浏览器运行结果

2.1.2　PHP 语言标记风格

在 PHP 程序中出现“<? php”和“? >”标识符就是 PHP 标记。PHP 标记告诉 Web 服务器 PHP 代码何时开始和结束。这两个标记之间的代码将被解释成 PHP 代码，PHP 标记用来隔离 PHP 和 HTML 代码。

PHP 的标记风格如下。

1. 标准风格

这种风格是以“<? php”开始，以“? >”结束（<? 和 php 之间不能有空格）。

```
<? php
    //PHP 代码
? >
```

这是本书使用的标记风格，也是最常见的一种风格。它在所有服务器环境上都能使用，而在 XML（可扩展标记语言）代码中嵌入 PHP 代码时就必须使用这种标记以适应 XML 的标准，所以推荐用户使用这种标记风格。

2. 简短风格

简短风格是以“<?”开始，以“? >”结束。

```
<?
    //PHP 代码
? >
```

这种风格是最简单的标记风格。要使用简短风格，必须保证 php. ini 文件中的 short_open_tag = On（Phpstudy 中默认是禁止的）。

3. 脚本 script 标记风格

```
<script language = "php"
    //PHP 代码
</script>
```

这种风格是将 PHP 代码写在<script>标记中，类似 JavaScript 的编写方式。

4. ASP 风格

这种风格以“<%”开始，以“%>”结束。

```
<%
    //PHP 代码
%>
```

与第 2 种风格一样，默认是禁止的，通过修改 php. ini 文件的 asp_tags 选项可以支持这种使用风格。当 PHP 代码与 ASP 源代码混在一起时建议不要使用这种风格。

将 PHP 语句放置在开始和结束标记之间，可以告诉 PHP 解释器进行何种操作。而每一条 PHP 语句都是以分号结束的。在 PHP 中，分号用于分隔 PHP 语句，丢失分号是最容易出现的语法错误，读者在编写时一定要注意。

2.1.3　PHP 注释

在 PHP 程序中，间隔字符，如换行（回车）、空格和 tab（制表符），都被认为是空格。在 HTML 中空格字符将被忽略，在 PHP 中同样如此。多个空格显示时只显示一个空格，例如：

```
<? php
    echo "Hello          ";
    echo "world";
? >
```

上面代码输出的结果为“Hello world”。

注释可以理解为代码中的解释和说明，是程序中不可缺少的重要元素。PHP 注释和 PHP 代码相同，必须位于 PHP 的开始标记与结束标记之间，PHP 解释器将忽略注释中的所有文本，就像跳过多余的空格一样。

PHP 支持以下三种注释风格。

1. C++风格的单行注释（//）

```
<? php
    //这是 C++注释风格
? >
```

C++风格的单行注释以“//”开始，到该行结束的内容都是注释。

2. Shell 风格的单行注释（#）

```
<? php
    #这是 shell 风格的注释
? >
```

shell 风格的单行注释以“#”开始，到该行结束的内容都是注释。

3. C 风格的多行注释（/ * … * /）

```
<? php
  / *
    第一行注释
```

```
    第二行注释
  */
echo "这是 C 语言的多行注释";
? >
```

C 风格的多行注释以“/*”开始，以“*/”结束。与 C 语言相同，多行注释不允许嵌套。

2.1.4 PHP 页面输出

PHP 代码作为服务器脚本在后台运行，运行得出的数据通过 PHP 自带的显示函数输出到浏览器页面中，一般使用 echo（）函数和 print（）函数。

1. echo（）函数

echo（）函数是 PHP 中最常用的函数，其功能是将一个或多个表达式的值输出到页面上。语法格式为：

```
echo("string arg1,...,string [argn]...")
```

echo（）函数会将传入的 arg1，arg2，…，argn 等参数的值输出到网页上。例如，在网页上输出字符串“你好！PHP”，代码如下：

```
<? php
    echo "你好！PHP";
? >
```

echo（）函数可以一次输出多个值，各个值之间要以逗号分隔。例如：

```
<? php
    echo "计算结果:",1+2+3+4;
? >
```

浏览器上显示的结果为：

计算结果：10

2. print（）函数

print（）函数的功能是将字符串输出到浏览器或打印机等输出设备。语法格式为：

```
int print( "string arg")
```

该函数执行成功返回 1，失败返回 0。

print（）函数一次只能输出一个字符串。

例如，在浏览器上输出“你好！PHP”，代码如下：

```
<? php
    print("你好！PHP");
? >
```

显示函数在输出字符串时一般使用双引号将字符串括起来，单引号也可以。echo（）和 print（）函数其实都不是真正的函数，而是一种语言结构，调用时不必加括号，在实际编程中也推荐使用这种方法：

```
<? php
```

```
    echo("hello");              //使用带括号的 echo()函数
    echo "world";               //使用不带括号的 echo()函数
    print("hello");             //使用带括号的 print()函数
    print "world";              //使用不带括号的 print()函数
?>
```

2.1.5　在 HTML 代码中嵌入 PHP

在 HTML 代码中嵌入 PHP 代码相对来说比较简单，下面是一个在 HTML 代码中嵌入 PHP 代码的例子：

```
<! DOCTYPE html>
<html>
<head>
    <title>HTML 中嵌入 PHP</title>
</head>
<body>
    HTML 文本框
    <input type="text" value="<? php echo '这是 PHP 的输出内容'? >">
</body>
</html>
```

说明：服务器在解析 PHP 文件时，如果遇到“<? php”和“? >”符号，就把这两个符号内的代码作为 PHP 代码进行解析。在 HTML 代码中插入 PHP 代码正是使用这种方法来完成的。上面代码的运行结果如图 2-4 所示。

图 2-4　运行结果

网络服务器解析 HTML 代码的速度要比解析 PHP 代码快得多，所以在实际应用中为了节省服务器资源，构建 PHP 程序时如果有大量的 HTML 代码，应该在 HTML 代码中嵌入 PHP 程序来显示。

2.1.6　在 PHP 代码中嵌入 JavaScript

在 PHP 代码中嵌入 JavaScript 能够与客户端建立起良好的用户交互界面，强化 PHP 的功能，其应用十分广泛。在 PHP 中生成 JavaScript 脚本的方法与普通页面输出的方法一样，可以使用显示函数。例如：

```
<? php
    echo "<script>";
    echo "alert('我是 JavaScript! ');";
    echo "</script>";
```

```
? >
```

说明：alert（）函数生成一个弹出对话框，如图 2-5 所示，其内容就是函数的参数。有关 JavaScript 的内容在后面还会涉及。

图 2-5　JavaScript 对话框

【例 2.1-1】综合之前学习的内容，制作一个 PHP 和 HTML，JavaScript 结合的网页。新建 EX2.1-1.php 文件，输入代码：

```
<! DOCTYPE html>
<html>
<head>
    <title>标记应用</title>
    <style type="text/css">
    p{
        text-align:center;
        font-family:"黑体";
        font-size:24px;
    }
    </style>
</head>
<body>
    <p>HTML5 页面</p>
    <? php
        $str1="PHP 变量 1";              //在弹出框中显示
        $str2="PHP 变量 2";              //在文本框中显示
        echo "<script>";
        echo "alert('". $str1."');";     //在 JavaScript 中使用 $str1 变量
        echo "</script>";
    ? >
    <input type="text" name="tx" size=20><br/>
    <input type="button" name="bt" value="单击" onclick="tx.value='<? php echo $str2;? >'">
</body>
```

```
</html>
```

保存后运行该文件，页面弹出如图 2-6 所示的对话框，单击【确定】按钮后，页面中出现一个文本框和一个名为“单击”的按钮，单击【单击】按钮，文本框中会显示“PHP 变量 2”，如图 2-7 所示。

图 2-6　弹框效果

图 2-7　页面文本框显示 $str2 变量

说明：语句“echo "alert ('". $str1. "');";”在输出 JavaScript 脚本中使用了连接符“.”将 JavaScript 代码和 PHP 代码串联起来，从而使 JavaScript 能够输出 PHP 变量。

2.1.7　PHP 编码规范

养成良好的编程习惯，能够提高代码的质量和编写的效率。不良的编程习惯经常会造成代码缺陷，使得程序难以维护，甚至在维护时又可能引入新的缺陷。因此书写 PHP 代码时需要遵循一些基本的编码规范。

1. 书写规范

（1）缩进。使用 Tab 键（制表符）缩进，缩进单位为 4 个空格。

（2）大括号 { }。大括号经常与选择语句（if…else）、循环语句（for、while）配合使用，构成语句块。

（3）关键字、小括号、函数、运算符。尽量将小括号与关键字用空格隔开。例如：

```
if ( $a<10)
{
    …
}
```

小括号与函数名不要隔开，以便区分开关键字和函数。例如：

```
empty( $string)
```

使用 return 返回语句时，不要使用小括号。例如：

```
return10;
```

2. 命名规范

命名遵循的基本原则是：以英文单词为蓝本，尽量不要使用拼音或拼音和英文混杂的命名方式。命名要尽量简短，含义一目了然。

（1）类。类名的首字母使用大写形式，其他的尽量使用小写形式。尽量不要使用下画线“_”。例如：ObjectName、MyDatabase。

（2）常量。常量名的所有字母均要大写，单词间以下画线“_”分隔。例如：PI、TRUE、MY_NAME 等。

（3）变量。所有字母都建议使用小写形式。使用下划线“_”作为每个单词的分隔符。同一个软件系统，变量的命名规则必须统一。

2.2 数 据 类 型

PHP 的数据类型是指变量的数据类型，其变量不需要声明就可以直接赋值，所以值的类型即为变量的类型。

PHP 支持 8 种基本的数据类型：integer（整型）、float（浮点型）、string（字符串型）、boolean（布尔型）、array（数组）、object（对象）、NULL（空）、resource（资源）。除此之外，为了提高代码的可读性，PHP 还支持一些伪类型变量。

2.2.1 整型

整型变量的值是整数，表示范围是-2 147 483 648~2 147 483 647。整型值可以用十进制数、八进制数、十六进制数或二进制数（PHP 5.4.0）的标志符号指定，前面再加上可选符号（-或+）。八进制数符号指定，数字前必须加 0；十六进制数符号指定，数字前必须加 0x；二进制数符号指定，数字前必须加 0b。例如：

```
$n1=656;                //十进制数
$n2=0;                  //零
$n3=-42;                //负数
$n4=0123;               //八进制数(等于十进制数的 83)
$n5=0x1B;               //十六进制数(等于十进制数的 27)
$n6=0b100101            //二进制数(等于十进制数的 37)
```

2.2.2 浮点型

浮点类型也称浮点数、双精度数或实数。浮点数的字长与平台相关，最大值是 1.8e308，并具有 14 位十进制数的精度。例如：

```
$pi=3.1415926;
$length=1.3e4;
$volume=7e-10;
```

2.2.3 字符串型

字符串是连续的字符序列，字符串的组成字符包含以下几种类型。

字母类型：A，B，a，b 等。

数字类型：1，2 等。

特殊字符：#，＊，+，^等。

不可见字符：\ n（换行符），\ r（回车符），\ t（Tab 字符）等。

其中不可见字符是用来控制字符串格式化输出的一种特殊字符。它在浏览器页面上不可见，只能看到其控制的字符串的输出结果。

字符串可以用 3 种方法来定义：单引号、双引号、heredoc 结构。

1. 单引号

定义字符串最简单的方法是用单引号“' ”括起来。如果要在字符串中表示单引号，则需要用转义符“\ ”将单引号转义之后才能输出。和其他语言一样，如果想在单引号之前或字符串结尾处出现一个反斜线“\ ”，就要使用两个反斜线来表示。例如：

```
<? php
    echo '输出\'单引号';                    //输出:输出'单引号
    echo '输出反斜线\\';                    //输出:反斜线\
? >
```

2. 双引号

使用双引号“"”将字符串括起来同样可以定义字符串。如果要在定义的字符串中表示双引号，则同样需要用转义符转义。另外，还有一些特殊字符的转义序列，见表 2-1。

表 2-1　特殊字符转义序列表

序　列	含　义
\ n	换行（LF 或 ASCII 字符 0x0A（10））
\ r	回车（CR 或 ASCII 字符 0x0D（13））
\ t	水平制表符（HT 或 ASCII 字符 0x09（9））
\ v	垂直制表符（VT 或 ASCII 字符 0x0B（11））（PHP 5.2.5）
\ e	Escape（ESC 或 ASCII 字符 0x1B（27））（PHP 5.4.0）
\ f	换页（FF 或 ASCII 字符 0x0C（12））（PHP 5.2.5）
\ \	反斜线
\ $	美元符号
\ "	双引号
\ [0-7]{1,3}	此正则表达式序列匹配一个用八进制符号表示的字符
\ x[0-9A-Fa-f]{1,2}	此正则表达式序列匹配一个用十六进制符号表示的字符

使用双引号和单引号的主要区别是，单引号定义的字符串中出现的变量和转义序列不会被变量的值替代，而双引号中使用的变量名在显示时会显示变量的值。例如：

```
<? php
    $str="加油";
    echo '中国 $str!';                    //输出:中国 $str!
    echo "中国 $str!";                    //输出:中国加油!
```

```
? >
```

字符串的连接：使用字符串连接符“.”可以将几个文本连接成一个字符串。通常使用 echo 命令向浏览器输出内容时使用这个连接符可以避免编写多个 echo 命令。例如：

```
<? php
    $str= "PHP 变量";
    echo "连接成"."字符串";                    //字符串与字符串连接
    echo  $str."连接字符串";                   //变量和字符串连接
? >
```

3. heredoc 结构

第三种定义字符串的方法是用 heredoc 结构“<<<”。使用时，应该在“<<<”之后提供一个标志符，然后是字符串，最后用同样的标志符结束字符串。结束标志符必须从行的第一列开始，标志符必须遵循 PHP 中标记的命名规则：只能包含字母、数字、下划线，而且必须以下划线或字母开始。例如：

```
<! DOCTYPE html>
<html>
<head>
    <title>定界符</title>
    <style type="text/css">
    p{font-size:24px;}
    </style>
</head>
<body>
<? php
 $name="李明";
echo <<<EOT
    My name is  $name
EOT;
? >
<p>
<? php
echo <<<EOD
    <br/>My name is Tom.<br/>
    <br/>How are you.<br/>
EOD;
? >
</p>
</body>
</html>
```

运行效果如图 2-8 所示。

图 2-8　使用 heredoc 结构输出字符串

说明：代码中的“EOT”和“EOD”都是标志符，可以使用别的来代替，但开始标志符和结束标志符必须一致，此界定符在需要输出大量的 HTML 文本时比较适用。

注意：开始标志符所在行的标志符之后不能包含任何字符，包括空格、制表符。同样结束标志符所在行也不能包含任何其他字符，除了一个分号“;”之外。如果破坏了这条规则，结束标志符就不会被视为结束标志符，PHP 将继续向下寻找结束符，这会导致在脚本最后一行出现一个语法错误。

2.2.4　布尔型

布尔型是最简单的一种数据类型，其值可以是 TRUE（真）或 FALSE（假），这两个关键字不区分大小写。要想定义布尔变量，只需将其值指定为 TRUE 或 FALSE。布尔型变量通常用于流程控制，例如：

```
<? php
    $a=TRUE;                    //设置变量值为 TRUE
    $b=FALSE;                   //设置变量值为 FALSE
    $username="李明";
    //使用字符串进行逻辑控制
    if( $username==="李明")
    {
        echo "Hello,李明!";
    }
    //使用布尔值进行逻辑控制
    if( $a==TRUE)
```

```
    {
        echo "a 为真";
    }
    //单独使用布尔值进行逻辑控制
    if( $b)
    {
        echo "b 为真";
    }
? >
```

说明：定义布尔变量后使用 echo 命令输出 TRUE 值的结果为 1，输出 FALSE 值的结果为 0。

2.2.5 数组和对象

数组是一组由相同数据类型元素组成的一个有序映射。在 PHP 中，映射是一种把 values（值）映射到 keys（键名）的类型。数组通过 array（）函数定义（自 PHP 5.4 起可以使用短数组定义语法，用［ ］替代 array（）），其值使用“key=>value”的方式设置，多个值通过逗号分隔。当然也可以不使用键名，默认是 1，2，3，…。例如：

```
<? php
    $ar1=array(1,2,3,4,5,6,7,8,9);                          //直接给数组赋值
    $ar2=array("animal "=>"tiger", "color"=>"red","numer"=>"12");//为数组指定键名和值
                                                             // PHP 5.4 起支持
    $ar2=["animal "=>"tiger", "color"=>"red","numer"=>"12"];
? >
```

在 PHP 中，是通过关键字“new”把对象实例化到一个变量中的，例如：

```
<? php
    //定义一个类
    class test
    {
        var  $items=0;
        function users()
        {
            $this->items=100;
        }
    }
    $newtest=new test();                    //初始化对象
    echo  $newtest->items;                  //访问对象的属性
    $newtest->users();                      //访问对象的方法
    echo  $newtest->items;
? >
```

说明：这里只简单了解数组和对象，具体内容将在后面的章节中介绍。

2.2.6　NULL 类型

特殊的 NULL 值表示一个变量没有值。NULL 类型唯一可能的值就是 NULL（不区分大小写）。在下列情况下一个变量被认为是 NULL。

- 被直接赋值为 NULL。
- 尚未被赋值。
- 被 unset（）函数销毁。

例如：

```
<? php
 $var1 = NULL;                    //直接赋值为 NULL
 $var2;                           //未赋值
 $var3 = " value" ;
unset( $var3) ;                   //被销毁
var_dump( $var1) ;                //输出 NULL
var_dump( $var2) ;                //输出 NULL
var_dump( $var3) ;                //输出 NULL
? >
```

2.2.7　资源类型

资源类型（resource）是一种特殊变量，它保存了到外部资源的一个引用。资源是通过专门的函数来建立和使用的。特定的内置函数（如数据库函数）将返回 resource 类型的变量，它们都代表外部资源，如文件、数据库链接等。在操作资源时可以使用 get_resource_type（）函数获得资源的类型信息。

2.2.8　伪类型

伪类型并不是 PHP 语言中的基本数据类型。因为 PHP 是弱类型语言，在一些函数中，一个参数可以接受多种类型的数据，还可以接受别的函数作为回调函数使用，故引入伪类型来确保程序代码的可读性，常用的伪类型有如下几种。

- mixed：说明一个参数可以接受多种不同的（但不一定是所有的）类型。例如，gettype（）可以接受所有的 PHP 类型，str_replace（）可以接受字符串和数组。
- number：说明一个参数可以是 integer 或者 float。
- void：作为返回类型意味着函数的返回值是无用的，作为参数列表意味着函数不接受任何参数。
- callback：自 PHP 5.4 起可用 callback 类型指定回调类型。一些函数如 call_user_func（）或 usort（）可以接受用户自定义的回调函数作为参数，回调函数不仅可以是简单函数，也可以是对象的方法，包括静态类方法。一个 PHP 的函数以 string 类型传递其名称，可以使用任何内置的或者用户自定义函数，但除了语言结构 array（），echo（），

empty（），eval（），exit（），isset（），list（），print（）或 unset（）等。

在函数原型中，“$…”表示“等等”的意思。当一个函数可以接受任意个参数时使用此变量名。

2.2.9 类型转换

PHP 在定义变量时不需要明确的类型定义，变量类型是根据赋给变量的值来决定的。也就是说，把一个字符串赋给变量 $var，则 $var 的类型就是 string 型，如果又把一个整型赋给 $var，则 $var 的类型变为整型，这也是 PHP 的特色之一。

PHP 自动类型转换的另一个例子是加号“+”。如果一个数是浮点数，则使用加号后其他的所有数都被当作浮点数，结果也是浮点数。否则，参与“+”运算的运算数都将被解释成整数，结果也是一个整数。例如：

```
<? php
    echo  $str1 = "1";                    // $str1 为字符串型
    echo  $str2 = "ab";                   // $str2 为字符串型
    echo  $num1 = $str1 + $str2;          // $num1 的结果是整型(1)
    echo  $num2 = $str1+5;                // $num2 结果是整型(6)
    echo  $num3 = $str1+2.56;             // $num3 结果是浮点型(3.56)
    echo  $num4 = 5+"6kb";                // $num4 结果为整型(11)
? >
```

读者可以运行比照 echo 输出的结果加以理解。

PHP 还可以使用强制类型转换，它将一个变量或值转换为另一种类型，这种转换与 C 语言类型的转换是相同的：在要转换的变量前面加上用括号括起来的目标类型。PHP 允许的强制转换如下：

(int)，(integer)：转换成整型。

(string)：转换成字符串型。

(float)，(double)，(real)：转换成浮点型。

(bool)，(boolean)：转换成布尔型。

(array)：转换成数组。

(object)：转换成对象。

(unset)：转换为 NULL（PHP 5）。

(binary)，b 前缀：转换为二进制字符串（PHP 5.2.1 新增）。例如：

```
<? php
    echo  $var = (int)"hello";            //变量为整型(值为 0)
    echo  $var = (int)TRUE;               //变量为整型(值为 1)
    echo  $var = (int)12.56;              //变量为整型(值为 12)
    echo  $var = (string)10.5;            //变量为字符串型(值为"10.5")
    echo  $var = (bool)1;                 //变量为布尔型(值为 TRUE)
    echo  $var = (boolean)0;              //变量为布尔型(值为 FALSE)
```

```
    echo  $var=(boolean)"0";              //变量为布尔型(值为 FALSE)
     $string="binary string";
    echo  $var=(binary) $string;          //变量为二进制字符串(值为"binary string")
    echo  $var=b"binary string";          //变量为二进制字符串(值为"binary string")
? >
```

说明：

- 强制转换成整型还可以使用 intval（）函数，转换成字符串型还可以使用strval（）函数。例如：

```
 $var=intval("12ab3c");                   //变量为整型(值为 12)
 $var=strval(2.3e5);                      //变量为字符串型(值为 230 000)
```

- 在将变量强制转换为布尔类型时，当被强制转换的值为整型值 0、浮点型 0.0、空白字符或字符串"0"、没有特殊成员变量的数组、特殊类型 NULL 时都被认为是 FALSE，其他的值都被认为是 TRUE。
- 如果要获得变量或表达式的信息，如类型、值等，可以使用 var_dump（）函数。例如：

```
<? php
    $var1=var_dump(123);
    $var2=var_dump((int)FALSE);
    $var3=var_dump((bool)NULL);
   echo  $var1;                    //输出结果:int(123)
   echo  $var2;                    //输出结果:int(0)
   echo  $var3;                    //输出结果:boolean(false)
? >
```

结果中，前面是变量的数据类型，后面是变量的值。

2.3　变量与常量

程序运行时，存储在内存的数据有两种形态，即常量和变量。在程序运行中值不改变的量称为常量，根据条件值发生变化的量称为变量。

2.3.1　自定义变量

由前面的内容可知，PHP 变量是由“ $”标志的变量名来表示的。自定义变量可以根据用户的需求自己定义，且变量名是区分大小写的。

1. 变量名的定义

在定义变量时，变量名与 PHP 中其他标记一样遵循相同的规则：

- 变量名必须以美元符号 $开头。
- 变量名的第一个字符必须是字母或下划线，其后可以是字母、数字和下划线的

组合。

- PHP 中的变量名区分大小写。

例如：

```
<? php
    //合法变量名
     $a = 1;
     $A = 2;
     $a123 = 1;
     $abc = 1;
    //非法变量名
     $123 = 1;
     $12Ab = 1;
     $a.r = 1;
     $好 = 1;
     $ * a = 1;
? >
```

2. 变量的初始化

PHP 变量的类型有布尔型、整型、浮点型、字符串型、数组、对象、资源和 NULL。数据类型在前面已经做过介绍。变量在初始化时，使用“=”给变量赋值，变量的类型会根据其赋值自动改变。例如：

```
$var = "abc";                    // $var 为字符串型
$var = TRUE;                     // $var 为布尔型
$var = 123;                      // $var 为整型
```

PHP 也可以将一个变量的值赋给另外一个变量。例如：

```
<? php
    $height = 100;
    $width = $height;                    // $width 的值为 100
? >
```

3. 变量的引用

PHP 提供了另外一种给变量赋值的方式——引用赋值，即新变量引用原始变量，改动新变量的值将影响原始变量，反之亦然。使用引用赋值的方法是，在将要赋值的原始变量前加一个“&”符号。例如：

```
<? php
    $var = "hello";                  // $var 赋值为 hello
    $bar = & $var;                   //变量 $bar 引用 $var 的地址
    echo $bar;                       //输出结果:hello
    $bar = "world";                  //给变量 $bar 赋新值
    echo $bar;                       //输出结果:world
? >
```

注意：只有已经命名过的变量才可以引用赋值，下面的用法是错误的：

```
$bar=&(25 * 5);
```

4. 变量的作用域

变量的作用域是指变量的作用范围。一个变量初始化之后，其作用范围就确定了。大多数的 PHP 变量只有一个作用域。按作用域的大小，变量一般分为局部变量和全局变量。

（1）局部变量。局部变量只是局部有效，它的作用域分为两种：

- 在当前文件主程序中定义的变量，其作用域限于当前文件的主程序，不能在其他文件或当前文件的局部函数中起作用。
- 在局部函数或方法中定义的变量仅限于局部函数或方法，在当前文件的主程序、其他函数、其他文件中无法引用。例如：

```
<? php
    $my_var="test";                                    // $my_var 的作用域仅限于当前主程序
    function my_func()
    {
        $local_var=123;                                // $local_var 的作用域仅限于当前函数
        echo '  $local_var=' . $local_var ."<br/>";    //调用该函数时输出结果值为 123
        echo '  $my_var=' . $my_var . "<br/>";         //调用该函数时输出结果值为空
    }
    my_func();                                         //调用 my_func()函数
    echo ' $my_var=' . $my_var . "<br/>";              //输出结果值为"test"
    echo ' $local_var=' . $local_var . "<br/>";        //输出结果值为空
? >
```

在函数的局部变量里还有一个特殊的例子——静态变量。它也属于函数中的局部变量，只不过一般变量在程序执行时，离开作用域后其值就会消失或改变，而静态变量在程序执行时，离开作用域后其值不会消失。静态变量使用“static”关键字来声明。例如：

```
<? php
    //创建函数 vars()
    function vars()
    {
        $var=0;                          //局部变量 $var 初始化为 0
        echo  $var. "<br/>";
        $var++;                          //加 1 操作
    }
    vars();                              //第 1 次调用 vars()函数,输出结果为 0
    vars();                              //第 2 次调用 vars()函数,输出结果仍为 0
    //创建函数 static_var()
    function static_var()
    {
        static  $var=0;                  //声明静态变量 $var
        echo  $var. "<br/>";
        $var++;
```

```
    }
    static_var( );                          //第 1 次调用 static_var( )函数,输出结果为 0
    static_var( );                          //第 2 次调用 static_var( )函数,输出结果为 1
? >
```

(2) 全局变量。PHP 的全局变量和 C 语言稍有不同。C 语言中，全局变量在函数中自动生效，除非被局部变量覆盖。而在 PHP 中，函数中使用全局变量必须使用“global”关键字先声明为全局变量，否则视为局部变量，且全局变量的个数没有限制。例如：

```
<? php
    $my_global = 1;                                          //定义变量 $my_global
    function my_func1( )                                     //函数 my_func1( )
    {
        global  $my_global;                                  //声明 $my_global 为全局变量
        global  $two_global;                                 //声明 $two_global 为全局变量
        echo ' $my_global =' . $my_global . "<br/>";         //调用该函数时输出结果值为 1
        $two_global = 2;                                     //将全局变量 $two_global 赋值为 2
    }
    function my_func2( )                                     //函数 my_func2( )
    {
        global  $two_global;                                 //声明 $two_global 为全局变量
        echo ' $two_global = ' . $two_global . "<br/>";      //调用该函数时输出结果值为 2
        $two_global = 3;
    }
    my_func1( );                                             //调用 my_func1( )函数,输出 1
    my_func2( );                                             //调用 my_func2( )函数,输出 2
    echo  $two_global;                                       //输出结果值为 3
? >
```

说明：函数 my_func1 ( ) 中声明的 $my_global 为全局变量后， $my_global 被初始化为 1，若在函数中将其值改为其他值，则在程序中 $my_global 的值也变为其他值；另外，在函数中声明全局变量时不能同时为其赋值，例如，“global $two_golbal = 3;”是错误的，必须分开两行来写。

注意：函数中定义的全局变量只有在函数被调用后才有效。

5. 检查变量是否存在

要检测一个变量是否存在，可使用 isset ( ) 函数，它的语法格式如下：

```
bool isset ( mixed  $var [ , mixed  $var [ ,  $... ] ] )
```

若变量 $var 已经存在，则函数将返回 TRUE，否则返回 FALSE。例如：

```
<? php
    $var2 = 123;
    var_dump(isset( $var1));                    //返回 boolean false
    var_dump(isset( $var2));                    //返回 boolean true
? >
```

另外，unset（）函数释放一个变量。empty（）函数检查一个变量是否为空或零值，如果变量值是非空或非零值，则 empty（）返回 FALSE，否则返回 TRUE。换句话说，""、0、"0"、NULL、FALSE、array（）、$var，以及没有任何属性的对象都将被认为是空的。例如：

```
<? php
    $var = 0;
    if( empty( $var) )
        echo "变量为空";                    //输出"变量为空"
? >
```

2.3.2　可变变量

在不确定一个变量的名称时，可以使用可变变量。所谓可变变量就是一个变量的变量名可以动态地设置和使用。一个普通变量通过声明来设置，而一个可变变量通过获取一个普通变量的值作为它的变量名。可变变量通过两个“ $”来设置。例如：

```
<? php
    $name = "Tom";
    $ $name = 20;
    echo  $ $name;                    //输出 20
    echo " ${ $name}";                //输出 20
    echo  $Tom;                       //输出 20
    echo  ${"Tom"};                   //输出 20
    $name = 123;                      //改变 $name 的值
? >
```

说明：程序中使用 $ $name 定义可变变量后，它的变量名就是它所引用的普通变量 $ name 的值"Tom"。在使用该可变变量时，$ $name、${ $name}、$Tom 和 ${"Tom"} 都可以表示该可变变量。其中当 $ $name 用双引号包含时，需要用“ {} ”表示成 ${ $ name} 再使用。但是当 $name 变量的值改变时，$ $name 变量名就不再适用，变量名为 $ Tom 和 ${"Tom"} 的变量值仍为 20。

2.3.3　预定义变量

预定义变量是由 PHP 预设的一组数组，通过这些预定义变量可以获取用户会话、客户机操作系统的环境和服务器操作系统的环境信息。因其作用范围全局有效，所以又称超全局变量或自动全局变量。超全局变量不可用作可变变量。常用预定义变量见表 2-2。

表 2-2　常用预定义变量

变量名	说　明
$GLOBALS	引用全局作用域中可用的全部变量，组成数组。变量名就是该数组的索引，它可以称得上是所有超级变量的超级集合

续表

变量名	说　明
$_GET	包含通过 GET 方法传递的参数的相关信息。主要用于获取通过 GET 方法提交的数据
$_POST	包含通过 POST 方法传递的参数的相关信息。主要用于获取通过 POST 方法提交的数据
$_COOKIE	通过 HTTP Cookies 传递到脚本的信息
$_SESSION	主要用于会话控制和页面间值的传递，包含与所有会话变量有关的信息
$_FILES	它是由 HTTP post 方法传递的已上传文件项目组成的数组
$_REQUEST	它是所有用户输入的变量数组，包括 $_GET、$_POST、$COOKIE 所包含的输入内容
$_SERVER ['SERVER_ADDR']	当前运行脚本所在服务器中的 IP 地址
$_SERVER ['SERVER_NAME']	当前运行脚本所在服务器的主机名称，如果该脚本运行在一个虚拟主机上，则该名称由虚拟主机所设置的值决定
$_SERVER ['SERVER_PORT']	服务器所使用的端口，默认值为 80
$_SERVER ['SERVER_SIGNATURE']	包含服务器版本和虚拟主机名的字符串
$_SERVER ['REMOTE_ADDR']	正在浏览当前页面用户的 IP 地址
$_SERVER ['REMOTE_HOST']	正在浏览当前页面用户的主机名
$_SERVER ['REMOTE_PORT']	用户连接到服务器所使用的端口
$_SERVER ['REQUEST_METHOD']	访问页面时的请求方法，如 GET，POST，PUT 和 HEAD 等
$_SERVER ['DOCUMENT_ROOT']	当前运行脚本所在的文档根目录
$_SERVER ['SCRIPT_FILENAME']	当前执行脚本的绝对路径名

PHP 还可以直接使用数组的参数名来定义超全局变量，例如“$_SERVER ["PHP_SELF "]”可以直接使用 $PHP_SELF 变量来代替，但该功能默认是关闭的，打开它的方法是，修改 php. ini 配置文件中“register_globals = Off”所在行，将“Off”改为“On”。但是全局系统变量的数量非常多，这样做可能导致自定义变量与超全局变量重名，从而发生混乱，所以不建议开启这项功能。例如：

```
<? php
    echo $_SERVER["SERVER_PORT"];          //输出 80
    echo $_SERVER["SERVER_NAME"];          //输出 localhost
    echo $_SERVER["DOCUMENT_ROOT"];        //输出 C:/wamp/www
? >
```

2.3.4　外部变量

在程序中定义或自动产生的变量叫内部变量，而由 HTML 表单、URL 或外部程序产生的变量叫外部变量。外部变量可以通过预定义变量 $_GET，$_POST，$_REQUEST 来获

得。表单可以产生两种外部变量：POST 变量和 GET 变量。POST 变量用于提交大量的数据，$_POST 变量从表单中接收 POST 变量，接收方式为“$_POST['表单变量名']”；GET 变量主要用于小数据量的传递，$_GET 变量从提交表单后的 URL 中接收 GET 变量，接收方式为“$_GET['表单变量名']”。$_REQUEST 变量可以取得包括 POST、GET 和 Cookie 在内的外部变量。

【例 2.3－1】分别用 POST 和 GET 方法提交表单，使用 $_GET，$_POST，$_REQUEST变量接收来自表单的外部变量。

新建 EX2.3-1.php 文件，输入代码：

```
<! DOCTYPE html>
<html>
<head>
    <title>外部变量演示</title>
</head>
<body>
<! -- 产生 POST 外部变量的 HTML5 表单 form1 -->
<form action="" method="post">
    学号:<input type="text" name="XH"><br/>
    姓名:<input type="text" name="XM"><br/>
    <input type="submit" name="postmethod" value="POST 方法提交">
</form>
<! -- 产生 GET 外部变量的 HTML5 表单 form2 -->
<form action="" method="get">
    性别:<input name="SEX" type="radio" value="男">男
        <input name="SEX" type="radio" value="女">女<br/>
    专业:<select name="ZY">
            <option>计算机应用技术</option>
            <option>软件技术</option>
            <option>数字媒体开发</option>
        </select><br/>
    <input type="submit" name="getmethod" value="GET 方法提交">
</form>
</body>
</html>
<? php
    //使用 isset( )函数判断是否是 POST 方法提交
    if( isset( $_POST['postmethod'] ) )
    {
        $XH= $_POST['XH'];                    //获取学号值
        $XM= $_POST['XM'];                    //获取姓名值
        echo "接收 POST 变量:<br/>";
```

```
        echo "学号:". $XH."<br/>";
        echo "姓名:". $XM."<br/>";
    }
    //使用 isset( ) 函数判断是否是 GET 方法提交
    if( isset( $_GET[ 'getmethod' ] ) )
    {
        $SEX= $_GET[ 'SEX' ];                    //GET 方法获取性别值
        $ZY= $_GET[ 'ZY' ];                      //GET 方法获取专业值
        echo "<br/>接收 GET 变量:<br/>";
        echo "性别:". $SEX."<br/>";
      echo "专业:". $ZY."<br/>";
    }
    echo "<br/>接收 REQUEST 变量:<br/>";          //将 REQUEST 方法获取的变量列在最后
    echo "学号:".@ $_REQUEST[ 'XH' ]."<br/>";     //使用 REQUEST 方法获取学号
    echo "姓名:".@ $_REQUEST[ 'XM' ]."<br/>";     //使用 REQUEST 方法获取姓名
    echo "性别:".@ $_REQUEST[ 'SEX' ]."<br/>";    //使用 REQUEST 方法获取性别
    echo "专业:".@ $_REQUEST[ 'ZY' ]."<br/>";     //使用 REQUEST 方法获取专业
? >
```

保存后运行，在学号文本框中输入“1716020112”，在姓名文本框中输入“李明”，单击 POST方法提交 按钮，运行结果如图 2-9 所示。接着在性别单选按钮中选择“男”，在专业下拉列表中选择“软件技术”，单击 GET方法提交 按钮，运行结果如图 2-10 所示。

图 2-9　POST 外部变量

图 2-10　GET 外部变量

说明：注意图 2-9 和图 2-10 地址栏的变化，GET 方法提交的表单变量会包含在 URL 中传送到指定文件。代码中，输出 $_REQUEST 变量时变量前加了一个“@”符号，这个符号是错误控制运算符，作用是可以忽略一些错误信息。由于在本例中提交按钮没有单击时，表单未提交，执行 $_REQUEST 变量的这部分代码时可能会出错，所以要加一个“@”运算符，错误控制运算符将在 2.4.5 小节介绍。

2.3.5　常量

在 PHP 中，常量是一个简单值的标志符，在脚本执行期间常量的值是不变的。常量

区分大小写，按照惯例，常量标志符一般都是大写。常量名和其他任何 PHP 标记都遵循相同的命名规则，合法的常量名以字母或下画线开始，后面跟任何字母、数字或下画线。常量的范围是全局的，常量一经定义，就可以在程序的任何地方访问它。

常量分为自定义常量和预定义常量。

1. 自定义常量

自定义常量使用 define（）函数来定义，语法格式如下：

```
define("常量名","常量值");
```

在 PHP 5.3.0 以后，可以使用 const 关键字在类定义之外定义常量：

```
const 常量名=常量值;
```

常量一旦定义，就不能再改变或取消定义，而且值只能是标量，数据类型只能是 boolean、integer、float 或 string。和变量不同，常量定义时不需要加“ $”。例如：

```
<? php
    define("_KO","156");
    const CONSTANT='Hello World! ';
    if(defined("CONSTANT"))
    {
        echo CONSTANT;                    //输出"Hello World!"
    }
    echo constant("_KO");                 //输出"156"
? >
```

说明：defined（）函数检查常量是否存在，constant（）函数读取常量的值。和使用 define（）来定义常量相反的是，使用 const 关键字定义常量必须处于最顶端的作用区域，因为用此方法是在编译时定义的。这就意味着不能在函数内、循环内以及 if 语句之内用 const 来定义常量。

2. 预定义常量

预定义常量也称魔术常量。PHP 提供了大量的预定义常量，但是很多预定义常量是由不同的扩展库定义的，只有加载这些扩展库后才能使用。预定义常量使用方法和常量相同，但是它的值会根据情况的不同而不同，经常使用的预定义常量有 8 个，这些特殊的常量是不区分大小写的，见表 2-3。

表 2-3　PHP 的预定义常量

名　称	说　明
__LINE__	常量所在的文件中的当前行号
__FILE__	常量所在的文件的完整路径和文件名
__DIR__	文件所在的目录（PHP 5.3.0 中新增）
__FUNCTION__	常量所在的函数名称，自 PHP 5 起本常量返回该函数被定义时的名称（区分大小写）
__CLASS__	常量所在的类的名称，自 PHP 5 起本常量返回该类被定义时的名称（区分大小写）。自 PHP 5.4 起对 trait 也起作用。当用在 trait 方法中时，__CLASS__是调用 trait 方法的类的名称

续表

名　称	说　明
__TRAIT__	Trait 的名称（PHP 5.4.0 新增），自 PHP 5.4 起此常量返回 trait 被定义时的名称（区分大小写）
__METHOD__	常量所在的类的方法名
__NAMESPACE__	当前命名空间的名称（区分大小写）。此常量是在编译时定义的（PHP 5.3.0 新增）

2.4　运算符与表达式

运算符是一种符号，指定要在一个或多个表达式中执行的操作。PHP 运算符包括算术运算符、赋值运算符、位运算符、比较运算符、错误控制运算符、条件运算符、递增/递减运算符、逻辑运算符、字符串运算符、数组运算符、类型运算符等。

2.4.1　算术运算符

PHP 有 6 种最基本的算术运算符：加（+）、减（-）、乘（*）、除（/）、取模（%）、取反（-）。例如：

```
<? php
    $a=10;
    $b=3;
    echo  $num= $a+ $b;                    //加法，$num 值为 13
    echo  $num= $a- $b;                    //减法，$num 值为 7
    echo  $num= $a * $b;                   //乘法，$num 值为 30
    echo  $num= $a/ $b;                    //除法，$num 值为 3.333333…
    echo  $num= $a% $b;                    //取模，$num 值为 1
    echo  $num=- $a;                       //取反，$num 值为-10
? >
```

读者可自己运行程序对照查看 echo 输出结果。

注意：除号“/”总是返回浮点数，即使两个运算数是整数（或由字符串转换成的整数）也是这样。取模“$a% $b”时，结果的正负号取决于%左端数据的正负号。

2.4.2　赋值运算符

赋值运算符的作用是将右边的值赋给左边的变量，最基本的赋值运算符是“=”。如“$a=3”表示将 3 赋给变量 $a，变量 $a 的值为 3。由“=”组合的其他赋值运算符还有“+=”“-=”“*=”“/=”“.=”等。例如：

```
<? php
    $a=10;
```

```
    $b=3;
    echo  $num= $a+ $b;          //将 $a+ $b 的结果值赋给 $num, $num 值为 13
    echo  $num=( $c=6)+4;        // $c 的值为 6, $num 的值为 10
    echo  $a+=6;                 //等同于 $a= $a+6, $a 赋值为 16
    echo  $b-=2;                 //等同于 $b= $b-2, $b 赋值为 1
    echo  $a * =2;               //等同于 $a= $a * 2, $a 赋值为 32
    echo  $b/=0.5;               //等同于 $b= $b/0.5, $b 赋值为 2
    $string="连接";
    echo  $string.="字符串";     //等同于 $string= $string."字符串", $string 赋值为"连接字符串"
? >
```

PHP 支持引用赋值，可以使用“ $var=& $othervar;”语法。引用赋值意味着两个变量指向了同一个数据，没有复制任何东西。例如：

```
<? php
    $a=3;
    $b=& $a;                     // $b 是 $a 的引用
    print " $a\n";               // 输出 3
    print " $b\n";               // 输出 3
    $a=4;                        // 修改 $a
    print " $a\n";               // 输出 4
    print " $b\n";               // 也输出 4,因为 $b 是 $a 的引用,因此也被改变
? >
```

自 PHP 5 起，new 运算符会自动返回一个引用，因此，再对 new 的结果进行引用赋值在 PHP 5.3 及以后版本中会发出一条 E_DEPRECATED 错误信息，而之前的版本则会发出一条 E_STRICT 错误信息。

例如，以下代码将产生警告信息：

```
<? php
    class C {}
    /* 输出错误信息:Deprecated: Assigning the return value of new by reference is deprecated
in... */
    $o=&new C;
? >
```

2.4.3　位运算符

位运算符可以操作整型和字符串型两种类型数据。它操作整型数的指定位置位，如果左、右参数都是字符串，则位运算符将操作字符的 ASCII 值。所有的位运算符及其说明见表 2-4。

表 2-4　PHP 的位运算符及其说明

位运算符	名　称	例　子	结　果
&	按位与	$a & $b	将 $a 和 $b 中都为 1 的位设为 1
|	按位或	$a | $b	将 $a 或 $b 中为 1 的位设为 1
^	按位异或	$a ^ $b	将 $a 和 $b 中不同的位设为 1
~	按位非	~ $a	将 $a 中为 0 的位设为 1，反之亦然
<<	左移	$a << $b	将 $a 中的位向左移动 $b 次（每一次移动都表示“乘以 2”）
>>	右移	$a >> $b	将 $a 中的位向右移动 $b 次（每一次移动都表示“除以 2”）

2.4.4　比较运算符

比较运算符用于对两个值进行比较，不同类型的值也可以进行比较，如果比较的结果为真则返回 TRUE，否则返回 FALSE。所有的比较运算符及其说明见表 2-5。

表 2-5　PHP 的比较运算符及其说明

比较运算符	名　称	例　子	结　果
==	等于	$a == $b	TRUE，如果 $a 等于 $b
===	全等	$a === $b	TRUE，如果 $a 等于 $b，并且它们的类型也相同
! =	不等	$a ! = $b	TRUE，如果 $a 不等于 $b
<>	不等	$a <> $b	TRUE，如果 $a 不等于 $b
! ==	非全等	$a ! == $b	TRUE，如果 $a 不等于 $b，或者它们的类型不同
<	小与	$a < $b	TRUE，如果 $a 严格小于 $b
>	大于	$a > $b	TRUE，如果 $a 严格大于 $b
<=	小于等于	$a <= $b	TRUE，如果 $a 小于或等于 $b
>=	大于等于	$a >= $b	TRUE，如果 $a 大于或等于 $b

说明：要注意，如果整数和字符串进行比较，字符串会被转换成整数；如果比较两个数字字符串，则作为整数比较。

2.4.5　逻辑运算符

逻辑运算符可以操作布尔型数据，PHP 中的逻辑运算符有 6 种，所有的逻辑运算符及其说明见表 2-6。

表 2-6　PHP 的逻辑运算符

逻辑运算符	名　称	例　子	结　果
and	逻辑与	$a and $b	TRUE，如果 $a 与 $b 都为 TRUE
or	逻辑或	$a or $b	TRUE，如果 $a 或 $b 任意一个为 TRUE
xor	逻辑异或	$a xor $b	TRUE，如果 $a 或 $b 任意一个为 TRUE，但不同时是
!	逻辑非	! $a	TRUE，如果 $a 不为 TRUE
&&	逻辑与	$a && $b	TRUE，如果 $a 与 $b 都为 TRUE
||	逻辑或	$a || $b	TRUE，如果 $a 或 $b 中任意一个为 TRUE

例如：

```
<? php
    $m=10;
    $n=6;
    if( $m>5&& $n<=8)              //判断 $m>5 和 $n<=8 是否都是 TRUE
    {
        echo "YES!";                //输出'YES! '
    }
? >
```

2.4.6　错误控制运算符

PHP 支持错误控制运算符@，将其放置在 PHP 表达式之前，该表达式可能产生的任何错误信息都将被忽略。例如：

```
<? php
    echo @ $a;          //变量 $a 未定义,不加@ 会显示 NOTICE 信息
    $a="Hello!";
    echo $a;            //输出"Hello!"
    $b=@ test( );       //忽略调用 test( ) 函数时产生的错误信息
    $con=@ mysql_conncet( "localhost","username","pwd" );
    //忽略连接 mysql 数据库出错产生的错误信息
? >
```

程序产生错误时，PHP 会将错误信息输出到页面上，使用错误控制运算符后就不再显示这些错误信息了。由于错误控制运算符只对表达式有效，因此可以将它放在变量、函数、常量等之前，而不能将它放在函数的定义或类的定义之前，也不能用于条件结构，如 if 和 foreach 等。

注意：错误控制运算符@ 甚至能够使导致脚本终止这样的严重错误的错误报告也失效。这意味着，如果在某个不存在或类型错误的函数调用前用“@”来抑制错误信息，则脚本不会显示程序在哪里出错的任何信息，这将给程序的调试造成很大的麻烦，所以使用

“@”符号时要特别慎重。

2.4.7 条件运算符

条件运算符是一个三元运算符，其语法格式如下：

```
表达式 1? 表达式 2:表达式 3
```

条件表达式的运算过程为：如果表达式 1 的值为 TRUE，则这个条件表达式的值为表达式 2 的值；如果表达式 1 的值为 FALSE，则这个表达式的值为表达式 3 的值。例如：

```
<? php
    $a=3;
    $b=4;
    $c=10;
    $d=15;
    echo $a> $b? $c: $d;
? >
```

程序运行结果为：

15

2.4.8 递增/递减运算符

PHP 支持 C 语言风格的递增与递减运算符。PHP 的递增/递减运算符主要是对整型数据进行操作，同时对字符也有效。这些运算符是前加、后加、前减和后减。前加是在变量前有两个“+”号，如“++ $a”，表示 $a 的值先加 1，然后返回 $a。后加的“+”在变量后面，如“$a++”，表示先返回 $a，然后 $a 的值加 1。前减和后减与加法类似。例如：

```
<? php
    $a=5;                    // $a 赋值为 5
    echo ++ $a;              //输出 6
    echo $a;                 //输出 6
    $a=5;
    echo $a++;               //输出 5
    echo $a;                 //输出 6
    $a=5;
    echo -- $a;              //输出 4
    echo $a;                 //输出 4
    $a=5;
    echo $a--;               //输出 5
    echo $a;                 //输出 4
? >
```

PHP 还可对字符进行递增运算，不过与 C 语言不同的是，对“'Z'”进行递增将得到

“'AA'”，而在 C 中，“'Z'+1”将得到“'［'”（“Z”的 ASCII 码为 90，“［”的 ASCII 码为 91）。在 PHP 中，字符变量只能递增，递减没有效果。例如：

```
<? php
    $a='a';
    echo ++$a;                    //输出'b'
    $a='z';
    echo ++$a;                    //输出'aa'
    $a='b';
    echo --$a;                    //递减无效,输出'b'
? >
```

注意：

对字符直接进行算术运算只能得到整型数据，如 $a='a'+1，$a 的值为 1。

递增/递减运算符不影响布尔值。递减 NULL 值没有效果，但是递增 NULL 的结果是 1。

2.4.9　字符串运算符

字符串运算符主要用于处理与字符串相关的操作。字符串运算符只有一个点“.”，该运算符用于将两个字符串连接成一个新字符串，它还可以与赋值运算符组合使用。例如：

```
<? php
    $a="Hello ";
    $b="World";
    echo $a.$b;                   //输出'Hello World'
    $a.="World";
    echo $a;                      //输出'Hello World'
? >
```

2.4.10　其他运算符

PHP 还提供了数组运算符，用来对两个数组进行比较，所有的数组运算符及其说明见表 2-7。PHP 还有一个类型运算符 instanceof，用来测定一个给定的对象是否来自指定的对象类。

表 2-7　PHP 的数组运算符

数组运算符	名　称	例　子	结　果
+	联合	$a + $b	$a 和 $b 的联合
==	相等	$a == $b	如果 $a 和 $b 具有相同的键/值对则为 TRUE
===	全等	$a === $b	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
! =	不等	$a ! = $b	如果 $a 不等于 $b 则为 TRUE

续表

数组运算符	名　称	例　子	结　果
<>	不等	$a <> $b	如果 $a 不等于 $b 则为 TRUE
! ==	不全等	$a ! == $b	如果 $a 不全等于 $b 则为 TRUE

2.4.11　运算符优先级和结合性

一般来说，运算符具有一组优先级，也就是它们的执行顺序。运算符还有结合性，也就是同一优先级的运算符的执行顺序，这种顺序通常是从左到右（左结合）、从右到左（右结合）或者非结合。表 2-8 从高到低列出了 PHP 运算符的优先级，同一行中的运算符具有相同优先级，此时它们的结合性决定了求值顺序。

表 2-8　PHP 运算符优先级和结合性

结合方向	运算符	附加信息
非结合	new	new
左	[	array（）
非结合	++，--	递增/递减运算符
非结合	!，~，-，(int)，(float)，(string)，(array)，(object)，@	类型
左	* / %	算数运算符
左	+，-，.	算数运算符和字符串运算符
左	<<，>>	位运算符
非结合	<，<=，>，>=	比较运算符
非结合	==,! =，===,! ==	比较运算符
左	&	位运算符和引用
左	^	位运算符
左	|	位运算符
左	&&	逻辑运算符
左	| |	逻辑运算符
左	? :	三元运算符
右	=，+=，-=，*=，/=，.=,%=，&=，|=，^=，<<=，>>=	赋值运算符
左	and	逻辑运算符
左	xor	逻辑运算符
左	or	逻辑运算符
左	,	多处用到

说明：表中未包括优先级最高的运算符——圆括号。它提供圆括号内部的运算符的优先级，这样可以在需要时避开运算符优先级法则。例如：

```
$a= $b * （2+ $c）;
```

如果写成：

```
$a = $b * 2+ $c;
```

就会得到错误的结果。

2.4.12　表达式

表达式是 PHP 最重要的基石。在 PHP 中，几乎所写的任何东西都是一个表达式。简单却最精确的定义表达式就是“任何有值的东西”。最基本的表达式就是常量和变量；一般的表达式大部分都是由变量和运算符组成的，如 $a=5；再复杂一点的表达式就是函数。例如：

```
<? php
    $a=10;
    $b= $a++;
    $c=test( );
    $a>1? $c+10: $c-10;
    function test( )
    {
        return 20;
    }
? >
```

2.5　综 合 案 例

本节通过一个综合案例来说明 PHP 变量、常量、运算符及表达式的使用。

【例 2.5-1】设计一个计算圆形和梯形面积的网页，要求分别用 POST 方法和 GET 方法接收圆的半径和梯形的上底、下底、高。

新建 EX2.5-1.php 文件，输入以下代码：

```
<! DOCTYPE html>
<html>
<head>
    <title>面积计算</title>
</head>
<body>
<! -- 产生 POST 外部变量的 HTML5 表单 form1 -->
<form action="" method="post">
    请输入圆的半径:<input type="text" name="R"><br/>
    <input type="submit" name="postmethod" value="POST 方法提交">
</form>
<! -- 产生 GET 外部变量的 HTML5 表单 form2 -->
<form action="" method="get">
```

```
        请输入梯形的上底:<input name="A" type="text"><br/>
        请输入梯形的下底:<input name="B" type="text"><br/>
        请输入梯形的高:<input name="H" type="text"><br/>
        <input type="submit" name="getmethod" value="GET 方法提交">
    </form>
    </body>
    </html>
    <? php
        define("PI",3.1415926);
        //使用 isset()函数判断是否是 POST 方法提交
        if(isset( $_POST['postmethod']))
        {
            $r= $_POST['R'];                          //获取半径值
            $C_area=PI * $r * $r;
            echo "接收 POST 变量:<br/>";
            echo "圆的面积为:". $C_area."<br/>";
        }
        //使用 isset()函数判断是否是 GET 方法提交
        if(isset( $_GET['getmethod']))
        {
            $a= $_GET['A'];                           //GET 方法获取梯形上底值
            $b= $_GET['B'];                           //GET 方法获取梯形下底值
            $h= $_GET['H'];                           //GET 方法获取梯形高值
            $T_area=( $a+ $b) * $h/2.0;
            echo "<br/>接收 GET 变量:<br/>";
            echo "梯形面积为:". $T_area."<br/>";
        }
    ? >
```

输入圆的半径 10，单击“POST 方法提交”按钮，运行结果如图 2-11 所示；输入梯形的上底 20，下底 30，高 10，单击“GET 方法提交”按钮，运行结果如图 2-12 所示。

图 2-11　POST 方法接收

图 2-12　GET 方法接收

本章小结

本章主要介绍了 PHP 的基本语法。在学完本章内容后，读者应重点掌握以下知识。

- PHP 语言标记风格、语法和注释的相关知识。
- PHP 中变量数据类型的定义是通过为变量赋值（初始化），由系统自动设定的。PHP 支持 8 种原始类型（type），其中有 4 种标量类型，2 种复合类型和 2 种特殊类型。标量数据类型包括 boolean（布尔型）、string（字符串）、integer（整型）和 float（浮点型，也称作 double）；复合数据类型包括 array（数组）和 object（对象）；特殊数据类型包括 resource（资源）和 NULL（NULL）。
- PHP 中的类型转换可以通过“显式转换”和“隐式转换”两种方式来实现。显式转换也叫强制类型转换，是在需要转换类型的变量前加上用“（）”括起来的数据类型名称或使用 settype（）函数来实现；隐式转换就是指自动类型转换。
- 变量就是一个保存了一小块数据的“对象”，从变量的字面意思可以理解为该数据块中的值是可以改变的，即在不同时段内代表不同的实体。在 PHP 中，变量采用美元符号（$）加变量名的方式来定义。
- 常量可以理解为值不变的量。常量在使用前必须先定义，并且只能是标量值（布尔值、整数、浮点数、字符串）。
- 运算符是一个特殊符号，它对一个值或一组值执行一个指定的操作，来产生另一个值。PHP 中包含算术运算符、比较运算符、赋值运算符、逻辑运算符、位运算符、字符串运算符等。

习　题

一、填空题

1. PHP 语言标记风格有四种，分别是_____、_____、_____、_____。

2. 标量数据类型包括 boolean（布尔型）、_____、_____和 float（浮点型）；复合数据类型包括 array（数组）和_____；特殊数据类型包括 resource（资源）和_____。

3. 按照变量作用域的不同，可以将 PHP 中的变量分为_____和_____。

4. PHP 中只有一个字符串运算符_____。其作用是将两个字符串或字符串与任何标量数据连接起来，组成一个新的字符串。

二、简答题

1. 在 PHP 中，变量的命名需要满足哪些条件？

2. 简述静态变量的含义。

3. 简述 PHP 可支持哪些数据类型，有哪些运算符。

第 3 章

PHP 流程控制结构

【本章内容简介】

本章主要学习程序控制结构语句的使用，主要包括分支结构、循环结构、流程控制符等知识。

【本章学习要求】

通过本章的学习，要求掌握选择结构、循环结构语句的使用。

所谓流程控制，就是控制程序的执行方向，它通过流程控制语句来实现。流程控制语句在编程语言中占有重要的地位，大部分程序段需要依靠它来完成。PHP 的程序流程控制语句主要包括条件控制语句、循环控制语句等，而这些控制语句大多可以嵌套使用。

在进行流程控制时，可能会使用到多条语句，这时就需要将多条语句封装成一个代码段。在将代码段分隔时，一般都使用“｛”作为代码段的开头，“｝”作为结尾，与 C 语言类似。PHP 语法在每个语句后面都要加分号，而代码段结束符号“｝”后面不加分号。

3.1 分 支 结 构

主要的分支结构有单向条件分支（if）、双向条件分支（else 从句）、多向条件分支（elseif 子句）、switch 语句等，我们需要掌握这些分支结构的用法，下面进行详细的介绍。

3.1.1 单向条件分支（if）

if 结构是单向条件分支结构，if 语句的基本格式是对一个表达式进行计算，根据计算结果决定是否执行后面的语句。if 后面小括号中的“表达式”就是执行的条件，条件返回的结果只能是布尔型值。其通常是由比较运算符或者逻辑运算符组成的表达式所计算的结果值，或是一些返回布尔型的函数等。如果是传入其他类型的值，也会自动转换为布尔型的 TRUE 或 FALSE。如果表达式为 TRUE，则执行代码块，否则不执行。

【例 3.1-1】

```
<? php
    $x=10;                    //定义一个整型变量 $x,值为 10
```

```
    $y=20;                          //定义一个整型变量 $y,值为 20
    if( $x< $y) {                   // $x 是小于 $y 的,所以执行下面语句块
        $t = $x ;                   //先将 $x 的值放到临时的变量 $t 中
        $x = $y ;                   //再将变量 $y 的值赋给变量 $x
        $y = $t ;                   //再将临时变量 $t 中的值赋给变量 $y
    }                               //语句块结束的花括号
    var_dump( $x > $y );            //两个变量的值已经交换,输出 true
    echo $x."". $y;
? >
```

3.1.2　双向条件分支（else 从句）

双向条件分支好比 if 语句中也可以包含 else 子句，其意思是需要在满足某个条件时执行一条语句，而在不满足该条件时执行其他语句，这正是 else 子句的功能。else 延伸了 if 语句，可以在 if 语句中的表达式的值为 FALSE 时执行语句。值得注意的是，else 语句是 if 语句的从句，必须和 if 一起使用，不能单独存在。

【例 3.1-2】

```
<? php
    $x=30;
    $y=20;
    if( $x> $y) {
        echo "变量\ $x 大于变量\ $y <br>";          //判断的条件成立,此句会执行
    } else {
        echo "变量\ $x 小于变量\ $y <br>";          //判断的条件不成立,此句会被执行
    }
    echo "变量\ $x 和变量\ $y,比较完毕。";
? >
```

3.1.3　多向条件分支（elseif 子句）

elseif 子句是典型的多向条件分支，它延伸了 if 语句，elseif 子句会根据不同的表达式值确定执行哪个语句块。在 PHP 中也可以将 elseif 分开成两个关键字 else if 来使用。其执行顺序是如果表达式 1 为 TRUE，则执行代码块 1 语句；如果判断表达式 2 为 TRUE，则执行代码块 2 语句；以此类推，判断第 n 个表达式为 TRUE，则执行代码块 n 语句；如果表达式的条件都不为 TRUE，则执行 else 子语中的代码块 n+1 语句，当然最后的 else 语句也可以省略。

【例 3.1-3】

```
<? php
    $week=@ date("D"); //获取当前的星期值,如 Mon、Tue、Wed 等,@ 为忽略警告等信息。

    if ( $week=="Mon") {
```

```
        echo "星期一";
    } elseif( $week == "Tue") {
        echo "星期二";
    } elseif( $week == "Wed") {
        echo "星期三";
    } elseif( $week == "Thu") {
        echo "星期四";
    } elseif( $week == "Fri") {
        echo "星期五";
    } elseif( $week == "Sat") {
        echo "星期六";
    } elseif( $week == "Sun") {
        echo "星期日";
    }
? >
```

3.1.4 switch 语句

switch 语句和 elseif 相似，也是一种多向条件分支结构，if 和 elseif 语句使用布尔表达式或布尔值作为分支条件来进行分支控制，而 switch 语句则用于测试一个表达式的值，并根据测试结果选择执行相应的分支程序，从而实现分支控制。switch 语句由一个选择表达式和多个 case 标签组成，case 标签后紧接着一段代码块。在使用 switch 语句时应该注意以下几点：

（1）switch 语句后面的选择表达式的数据类型只能是整型或字符串型，不能是 boolean 型。通常这个控制表达式是一个变量名称。

（2）switch 语句后面的花括号是必须有的。

（3）case 语句的个数没有规定，可以无限增加。但 case 标签和其后面的值之间应有一个空格，值后面必须有一个冒号，这是语法的一部分。

（4）switch 匹配完成以后，将依次逐条执行匹配的分支模块中的语句，直到 switch 结构结束或者遇到了 break 语句才停止执行。

（5）switch 语句中 default 标签直接在后面加上一个冒号，其意思是表达式的值不能与前面任何一个 case 标签后的值相等，这时才执行 default 分支中的语句。default 标签可以省略。

【例 3.1-4】

```
<? php
    $week = @ date( "D");
    switch( $week) {
        case "Mon": echo "星期一"; break;
        case "Tue": echo "星期二"; break;
        case "Wed": echo "星期三"; break;
        case "Thu": echo "星期四"; break;
        case "Fri": echo "星期五"; break;
```

```
        case "Sat": echo "星期六"; break;
        case "Sun": echo "星期日"; break;
    }
? >
```

3.2　循 环 结 构

计算机程序最擅长的功能之一就是按规定的条件，重复执行某些操作。循环结构可以减少源程序重复书写的工作量，即在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体，在 PHP 中提供 while 循环、do-while 循环和 for 循环三种。

3.2.1　while 循环

while 循环需要设定一个布尔型条件，当条件为真时，它不断地执行一个语句块，直到条件为假为止。程序执行到 while 语句后，会进行如下操作：

计算表达式的值，确定是 TRUE 还是 FALSE。

如果表达式为 FALSE，while 语句将结束，然后执行 while 语句之后的语句。

如果表达式为 TRUE，则执行 while 语句花括号中的代码块，然后返回到第 1 步执行。

【例 3.2-1】

```
<html>
    <head><title>使用 while 循环嵌套输出表格</title></head>
    <body>
        <table style="align: center;width:600px" border="1">
            <? php
                $out = 0;
                while( $out < 10 ) {
                    $bgcolor = $out%2 == 0? "#FFFFFF": "#DDDDDD";
                    echo "<tr style='background-color:". $bgcolor."'>"; //指定行的背景颜色
                    $in = 0;
                    while( $in < 10 ) {
                        echo "<td>".( $out * 10+ $in)."</td>";//执行一次,输出一个单元格
                        $in++;
                    }
                    echo "</tr>";
                    $out++;
                }
            ? >
        </table>
```

```
    </body>
</html>
```

3.2.2 do-while 循环

do-while 循环和 while 循环非常相似，区别在于表达式的值是在每次循环结束时检查。do-while 的循环语句必定会执行一次，因为表达式的结果值在每次循环结束后检查。

【例 3.2-2】

```
<? php
 $count = 0;
do {
    echo $count;
     $count++;
}
while ( $count < 10);
? >
```

3.2.3 for 循环

for 循环语句适用于明确知道重复执行次数的情况，即 for 语句将循环次数的变量在 for 语句中预先定义好。for 语句由分号分隔成三个部分，即循环变量初始化、条件表达式和循环变量自增长或者自减少。初始化是一个赋值语句，它用来给循环控制变量赋初值；条件表达式是一个关系表达式，它决定什么时候退出循环；增量定义循环控制变量，每循环一次后按什么方式变化。程序执行到 for 语句时，会进行如下操作：

第一次进入 for 循环时，对循环控制变量赋初值。

根据判断条件的结果决定是否要继续执行循环，如果判断条件为真，则继续执行循环；如条件为假，则结束循环执行下面的语句。

执行完循环体内的语句后，系统会根据循环控制变量增减方式，更改循环控制变量的值，再回到步骤 2 重新判断是否继续执行循环。例如：

```
<? php
    for( $i= 1; $i<= 10; $i++ )
        echo "这是第<b> $i </b>次循环执行输出的结果<br>";
? >
```

3.3 流程控制符

在上述介绍的循环结构中，都是通过循环语句本身提供的条件表达式来指定循环次数执行代码块直到停止循环。但如果想在循环体执行过程中中止循环，或是跳过一些循环继续执行其他代码块，我们就需要一些特殊的流程控制符来控制循环。

3.3.1　break 语句

break 可以结束当前 for、foreach、while、do-while 或者 switch 循环结构的执行。使用 break 语句可以将深埋在嵌套循环中的语句退出到指定层数，break 可以接受一个可选的数字参数来决定跳出几重语句。

【例 3.3-1】

```
<? php
    $i = 0;
    while ( ++ $i ) {                    //外层使用一个 while 语句的循环
        switch ( $i) {                   //内层使用一个 switch 语句
            case 5:
                echo "变量为 5 时,只退出 switch 语句<br>";
                break 1;                 //使用 break 1 退出 1 层
            case 10:
                echo "当变量为 10 时,不仅退出 switch,还退出 while 循环<br>";
                break 2;                 //使用 break 2 退出 2 层
            }
        }
? >
```

3.3.2　continue 语句

continue 语句只能使用在循环语句内部，功能是跳过该次循环，继续执行下一次循环结构。在 while 和 do-while 语句中 continue 语句跳转到循环条件开始处继续执行，对于 for 循环控制变量更新，然后继续执行代码块。

【例 3.3-2】

```
<? php
    $sum = 0;
    for ( $i=1; $i <= 10; $i++ ) {
        if ( $i%3 ! = 0)                 //不能被 3 整除,跳出本次循环
            continue;
        $sum += $i;
    }
    echo "结果为: $sum";                 //输出结果为 18
? >
```

3.3.3　exit 语句

当前的脚本中只要执行到 exit 语句，不管它在哪个结构中，都会直接退出当前脚本。exit（）是一个函数，die（）函数是 exit（）函数的别名，可以带有一个参数输出一条消

息，并退出当前脚本。

【例 3.3-3】

```
<? php
//如果连接 MySQL 数据库失败则使用 exit( )函数输出错误消息,并退出当前脚本
    $conn = mysql_connect( "localhost", "root", "root" ) or exit( "连接数据库失败!" );
? >
<? php
echo "连接失败,此脚本也不执行";
? >
```

3.3.4 return 语句

如果在一个函数中调用 return 语句，将立即结束此函数的执行并将它的参数作为函数的值返回。如果在全局范围中调用，则当前脚本文件中止运行。在 PHP 的脚本或者脚本的循环体内使用 return，将结束当前脚本的运行。

【例 3.3-4】

```
<? php
 $n=5;
for( $i=1; $i<10; $i++)
{
    if( $i> $n)
    {
        return;                    //当 $i>5 时结束脚本运行
        echo "大于 5";             //此处不输出任何内容
    }
    echo  $i ."";                  //输出 1 2 3 4 5
}
? >
```

3.3.5 exit 控制符

exit 控制符也可结束脚本的运行，用法和 return 控制符类似。

【例 3.3-5】

```
<? php
 $a=5;
 $b=6;
if( $a< $b)
    exit                       //如果 $a< $b 则结束脚本
echo  $a."小于". $b;           //不输出
? >
```

但 exit 和 return 的不同之处在于，当前文件包含另外的文件时，若在被包含文件中执

行 exit 语句，则在当前文件中被包含文件的下面的代码都将停止运行。而 return 控制符则不会停止当前文件中代码的运行。

3.4　包含文件操作

包含文件操作是指在当前文件中使用语句包含其他文件中的代码，并在当前文件中运行。在现实中，程序员编写程序的时候，不喜欢干同样的事情，也不喜欢把同样的代码（比如一些公用的函数）写几次，于是就把需要公用的代码写在一个单独的文件里面，比如 share. php，而后在其他文件进行包含调用。

在 PHP 中常用的包含文件操作语句包括 include（）、require（）、include_once（）、require_once（）等。语句中指定的文件，将被包含到当前文件中并运行。

这 4 种语句的用法类似，不同之处在于：

include（）包含文件发生错误时，如包含的文件不存在，脚本将发出一个警告，但脚本会继续运行。

require（）包含文件发生错误时，会产生一个致命错误并停止脚本的运行。

include_once（）使用方法和 include（）相同，但如果在同一个文件中使用 include_once（）函数包含了一次指定文件，那么此文件将不被再次包含。

require_once（）使用方法和 require（）相同，但如果在同一个文件中使用 require_once（）函数包含了一次指定文件，那么此文件将不被再次包含。

在包含文件时，函数中要指定正确的文件路径和文件名。如果不指定路径或者路径为“./”，则在当前运行脚本所在目录下寻找该文件，如 include（'1. php'）或 include（'./1. php'）。如果指定文件的路径为“../”，则在 Apache 的根目录下寻找该文件，如 include（'../1. php'）。如果要指定根目录下不是当前脚本所在目录下的文件，可以指定其具体位置，如 include（'../studyPHP/1. php'）。

当一个文件被包含时，从被包含的所在行开始，被包含文件中可用的任何变量在当前脚本中都可以直接使用，所有在包含文件中定义的函数和类都具有全局作用域。这样不但可以减少文件中代码的数量，还可以把重复的函数做成包含文件，供需要的文件调用。例如，假设我们有一个名为“footer. php”的标准的页脚文件：

```
<? php
echo "<p>Copyright © 2018-". date("Y") . "广东南方职业学院</p>";
? >
```

如需在另一张页面中引用这个页脚文件，可以使用 include 语句：

```
<html>
<body>
<h1>欢迎访问我们的首页！</h1>
<p>一段文本。</p>
<p>一段文本。</p>
```

```
        <? php
        include ' footer.php' ;
        ? >
    </body>
</html>
```

3.5 综 合 案 例

【例 3.5-1】 综合上述所学的流程控制的内容，编写回答多项选择题的 PHP 程序。题目要求如下：以下属于 Web 开发语言的有哪几种？供选择答案：C 语言，PHP，FLASH，ASP，JSP。正确答案：PHP，ASP，JSP。

新建 EX3.5-1.php 文件，输入以下代码：

```
<form action="" method="post">
    以下属于 Web 开发语言的有哪几种? <br>
    <input type="checkbox" name="answer[ ]" value="C 语言">C 语言<br>
    <input type="checkbox" name="answer[ ]" value="PHP">PHP<br>
    <input type="checkbox" name="answer[ ]" value="FLASH">FLASH<br>
    <input type="checkbox" name="answer[ ]" value="ASP">ASP<br>
    <input type="checkbox" name="answer[ ]" value="JSP">JSP<br>
    <input type="submit" name=bt_answer value="提交"/>
    </form>
    <? php
    if( isset( $_POST[' bt_answer' ] ) )
    {
        $answer=@ $_POST[' answer' ];                    // $answer 是数组
        if( ! $answer)
        echo "<script>alert(' 请选择答案' )</script>";
        $num = count( $answer);          //使用 count 函数取得 $answer 数组中值的个数
        $anw="";                                         //初始化 $anw 为空
        for( $i=0; $i< $num; $i++)                       //使用 for 循环
        {
            $anw= $anw. $answer[ $i];                    //将 $answer 中的值连接起来
        }
        if( $anw=="PHPASPJSP")                           //判断是否是正确答案
            echo "<script>alert(' 回答正确! ' )</script>";    //弹出提示框提示正确
        else
            echo "<script>alert(' 回答错误! ' )</script>";    //弹出提示框提示错误
        }
? >
```

运行结果如图 3.1 所示。

图 3.1　多项选择题运行结果

注意：在程序中，复选框的名称都为“answer［］”，当 PHP 从表单中“answer”取值，取到的是一个数组，复选框被选中时的值就存入这个数组中。count（）函数返回一个数组中值的个数。

本章小结

本章主要学习了 PHP 流程控制结构，特别介绍了分支结构、循环结构、流程控制结构、包含语句等。分支结构包含 if、elseif、switch 等语句，循环结构包括 while 循环、do-while 循环和 for 循环等，包含语句包括 include（），require（），include_once（），require_once（）等语句。

习　　题

一、简答题

1. 从执行方式上看，语句的控制结构分为哪几种?

2. PHP 中 有哪三种控制语句用以实现分支结构与循环结构? 请写出具体的语句。

二、编程题

1. 编程求 1~100 之间奇数之和。

2. 编程画出以下图形：

```
*
* * *
* * * * *
* * * * * * *
* * * * * * * * *
```

3. 编程找出 100~1000 之间的所有素数。

4. 编程找出所有的水仙花数。水仙花就是一个三位数的整数，其个位、十位、百位的三次方和等于原来的数，比如 $153=1^3+5^3+3^3$。

第 4 章

PHP 函数的应用

【本章内容简介】

函数是一个能独立完成某种功能的可重复调用的程序模块，其中封装了程序代码和数据，实现了更高级的抽象和数据隐藏，使得用户只关心函数的功能和使用方法，而不必更多地关心函数功能的具体实现细节。函数有两种：一种是系统内部提供的函数，你只要知道这个函数是干什么用的，自己会用就行了，不用管里面究竟是怎么实现的；另一种函数是自己定义的，用来实现自己独特的需求。本章重点讲解 PHP 中函数的定义和使用方法，并通过示例进行分析说明。

【本章学习要求】

掌握 PHP 函数的定义和调用。

掌握 PHP 函数参数的传递方法。

掌握 PHP 变量的作用域。

4.1 函数的定义

像数学中的函数一样，在数学中，$y=f(x)$ 是基本的函数表达形式，x 可看作是参数，y 可看作是返回值，所以函数定义就是一个被命名的、独立的代码段，它执行特定的任务，并可能给调用它的程序返回一个值。该定义中的各部分如下：

函数是被命名的：每个函数都有唯一的名称，在程序的其他部分使用该名称，可以执行函数中的语句，称为调用函数。

函数是独立的：无须程序其他部分的干预，函数便能够单独执行其任务。

函数执行特定的任务：任务是程序运行时所执行的具体工作，如将一行文本输出到浏览器、对数组进行排序、计算三次方根等。

函数可以将一个返回值返回给调用它的程序：程序调用函数时，将执行该函数中的语句，而这些语句可以将信息返回给调用它们的程序。

PHP 的模块化程序结构都是通过函数或对象来实现的，将复杂的 PHP 程序分为若干个功能模块，每个模块都编写成一个 PHP 函数，然后通过在脚本中调用函数，以及在函

数中调用函数来实现一些大型问题的 PHP 脚本编写。使用函数的优越性如下：

（1）提高程序的重用性。

（2）提高软件的可维护性。

（3）提高软件的开发效率。

（4）提高软件的可靠性。

（5）控制程序设计的复杂性。

函数是程序开发中非常重要的内容，因此，对函数的定义、调用和值的返回等，要尤其注重理解和应用，并通过上机调试加以巩固。

4.2　自定义函数

函数可以分为用户自定义函数和系统函数，用户自定义函数是由用户自行编写实现的，功能也是由用户所编写的代码来决定。PHP 系统函数允许和文件进行交互、使用数据库、创建图形，还可以连接其他的服务器。

编写函数的时候首先要明白希望函数做什么，系统函数是在 PHP 中提供的可以直接使用的函数，每一个系统函数都是一个完整的可以完成指定任务的代码段，多学会一个系统函数，就会多一个 PHP 的功能。使用系统函数的执行效率比自定义函数高，两种函数在程序中的调用方式没有区别。

4.2.1　函数的定义

PHP 为用户提供了自定义函数的功能，定义函数的格式如下：

```
function function_name([ $parameter[, ...]])
{
    函数体;
    return 返回值;          //可以从函数中返回一个值,可选。
}
```

定义函数的关键字为 function。function_name 是用户自定义的函数名，通常这个函数名可以是以字母或下划线开头后面跟 0 个或多个字母、下划线和数字的字符串，且不区分大小写。每个函数都有唯一的名称，但需要注意的是，在 PHP 中不能使用函数的重载，所以不能定义重名的函数，也包括不能与系统函数同名。给函数指定一个描述其功能的名称是个不错的主意。

定义函数时函数名后面的括号也是必须有的，在括号中的 $parameter 为函数的参数，函数一般可以有 0 个或多个参数，参数间用逗号分开。当参数个数为 0 时，函数可以被直接调用，当参数个数不为 0 时，用户在调用的时候就需要提供有效的参数。

使用关键字 return 可以从函数中返回一个值，在 return 后面加上一个表达式，程序执行到 return 语句时，该表达式将被计算，然后返回到调用的程序处继续执行，函数的返回

值为该表达式的值。

在函数定义时，花括号内的代码就是在调用函数时将会执行的代码，这段代码可以包括变量、表达式、流程控制语句，甚至是其他的函数或类定义。

【例 4.2-1】

```
<? php
function func( $a, $b)
{
    if( $a = = $b)
        echo "a=b";
    else if( $a> $b)
        echo "a>b";
    else
        echo "a<b";
}
? >
```

上面这段程序定义了一个简单的函数 func（），该函数的作用是比较两个数的大小，根据不同的情况输出不同的内容。

4.2.2 函数的调用

函数在定义后就可以被调用了。不管是自定义的函数还是系统函数，如果函数不被调用，就不会执行。只要在需要使用函数的位置，使用函数名称和参数列表进行调用。函数被调用后开始执行函数体中的代码，执行完毕返回到调用的位置继续向下执行。所以在函数调用时函数名称有以下三个作用。

（1）通过函数名称去调用函数，并让函数体的代码运行，调用几次函数体就会执行几次。

（2）如果函数有参数列表，还可以通过函数名后面的小括号传入对应的值给参数，在函数体中使用参数来改变函数内部代码的执行行为。

（3）如果函数有返回值，当函数执行完毕时就会将 return 后面的值返回到调用函数的位置处，这样就可以把函数名称当作函数返回的值使用。

只要定义的函数在脚本中可见，就可以通过函数名在脚本的任意位置调用，在 PHP 中可以在函数定义之后调用，也可以在函数定义之前调用，还可以在函数中调用函数。

【例 4.2-2】

```
<html>
<head>
<title>实现货币转换</title>
</head>
<body>
<? php
```

```
function example( $price, $tax)
{
    function examples( $yuan, $taxs=7.5)
    {
        return  $yuan * $taxs;
    }
 $total= $price+( $price * $tax);
echo "价格是: $total 美元<br>";
echo "价格是:".examples( $total)."元<br>";
}
example(15.00,0.75);
? >
</body>
</html>
```

运行结果：

价格是：26.25 美元

价格是：196.875 元

运行说明：

（1）以上程序是最常见的函数调用方式，也是嵌套函数，就是在函数中定义并调用其他函数。

（2）嵌套函数可以将一个复杂的功能分解成多个子函数，再通过调用的方式结合起来，有利于提高函数的可读性。

另外，函数如果没有返回值，调用时使用函数名即可。如果函数具有返回值，则可以将函数的返回值赋给一个变量。

【例 4.2-3】

```
<? php
//对一个数组进行升序排序的函数 my_sort()
function my_sort( $array)
{
    for( $i=0; $i<count( $array); $i++)
    {
        for( $j= $i+1; $j<count( $array); $j++)
        {
            if( $array[ $i]> $array[ $j])
            {
                $tmp= $array[ $j];
                $array[ $j]= $array[ $i];
                $array[ $i]= $tmp;
            }
        }
```

```
        }
        return $array;
    }
    $arr=array(6,4,7,5,9,2);                    //未排序的数组
    $sort_arr=my_sort( $arr);                   //将排序后的数组赋给 $sort_arr
    foreach( $sort_arr as $num)
        echo $num;                              //输出 245679
? >
```

4.2.3 参数的传递

参数列表是由零个、一个或多个参数组成的，各参数之间用逗号分隔。通过参数列表可以传递信息到函数。参数是从左向右求值的。一般情况下，参数是通过值来传递（默认），例如前面定义的 func（）函数就是通过变量 $a 和 $b 的值传递的。通过值传递参数不会因为函数内部参数的变化而改变函数外部的值。

另外，PHP 函数也可以通过引用传递参数，还支持可变长度参数列表。如果希望函数修改外部传来的参数值，可以使用引用参数传递，只要在定义函数时在参数前面加上“&”。

【例 4.2-4】

```
<? php
function fruit (& $fru)                         //定义函数 fruit ()
{
    $fru="apple";
}
$pear="pear";
fruit( $pear);                                  //调用函数 fruit (),参数使用变量 $pear
echo $pear;                                     //输出"apple "
? >
```

函数还可以定义 C++ 风格的标量参数默认值。在定义函数时给参数赋予默认值，参数的默认值必须是常量表达式，不能是变量、类成员或函数调用。

【例 4.2-5】

```
<? php
function makecoffee( $type = "cappuccino")
{
    return "Making a cup of $type. <br>";
}
echo makecoffee();
echo makecoffee(null);
echo makecoffee("espresso");
? >
```

注意：默认值必须是常量表达式，不能是诸如变量、类成员或者函数调用等表达式。当使用默认参数时，默认参数必须放在非默认参数的右侧；否则，函数将不会按照预期的情况工作。请思考下面函数默认参数不正确的用法代码片断：

【例 4.2-6】

```
<? php
function makeyogurt( $type = "acidophilus", $flavour)
{
    return "Making a bowl of  $type  $flavour. ";
}

echo makeyogurt("raspberry");          //不能正确显示
? >
```

函数默认参数正确的用法如下：

```
<? php
function makeyogurt( $flavour, $type = "acidophilus")
{
    return "Making a bowl of  $type  $flavour. ";
}
echo makeyogurt("raspberry"); // 正确显示
? >
```

这个例子的输出是：

Making a bowl of acidophilus raspberry.

4.2.4　递归函数

PHP 支持递归函数，递归函数即自调用函数，可以实现循环的作用。在函数体内部直接或者间接地自己调用自己，即函数的嵌套调用是函数本身。通常在此类函数之中会附加一个条件判断叙述，以判断是否需要执行递归调用，并且在特定的条件下终止函数的递归调用动作，把目前流程的主控权交回到上一层函数来执行。此外，如果当某个执行递归调用的函数没有附加条件判断叙述时，可能会造成无限循环的错误情形。

函数递归调用最大的好处在于可以精简程序中的复杂重复调用程序，并且能以这种特性来执行一些较为复杂的运算动作。例如，列表、动态树形菜单及遍历目录等操作。相应的非递归函数虽然效率高，但却比较难编程，而且相对来说可读性差。现代程序设计的目标主要是可读性好。随着计算机硬件性能的不断提高，程序在更多的场合优先考虑可读而不是高效，所以，鼓励用递归函数实现程序思想。

【例 4.2-7】

```
<? php
//声明一个函数,用于测试递归
function test( $n){
```

```
        echo  $n." ";          //在函数开始输出参数的值
        if( $n>0)                   //判断参数是否大于 0
        {
            test( $n-1);            //如果参数大于 0 则调用自己,并将参数减 1 后再次传入
        }
        else
        {                           //判断参数是不大于 0
            echo "<--------> ";
        }
        echo  $n." ";
    }
    test(10);                       //调用 test( ) 函数将整数 10 传给参数
? >
```

运行结果：

10 9 8 7 6 5 4 3 2 1 0 <--------> 0 1 2 3 4 5 6 7 8 9 10

说明：在上面的实例中声明了一个 test（）函数，该函数需要一个整型的参数。在函数外面通过传递整数 10 作为参数调用 test（）函数。在 test（）函数体中，第一条代码输出参数的值和一个空格。然后判断条件是否成立，成立则调用自己并将参数值减 1 再次传入。开始调用时，是外层调内层，内层调更内一层，直到最内层由于条件不允许必须结束。最内层结束了，输出 <--------> 作为分界符，执行调用之后的代码输出参数的值和空格，程序就会回到稍外一层继续执行。稍外一层在结束时，退回到再稍外一层继续执行，层层退出，直到最外层结束。执行完成以后的结果就是我们上面看到的结果。

4.2.5 变量函数

在 PHP 中是支持变量函数的，首先定义一些函数，再声明一个变量，使用变量来调用不同的函数，然后不断地重新给变量赋值。下面我们将通过一个实例来介绍变量函数的具体使用方法。在这个例子中首先定义 3 个函数，接着声明一个变量并赋值，通过变量函数来访问不同的 3 个函数，需要重新不断地给变量赋值。

【例 4.2-8】

```
<? php
function come( ){
    echo "谁进门了? <br>";
}
function go( $name="tom"){
    echo " $name 出门了<br>";
}
function back( $str){
    echo " $str 重新进门 <br>";
}
```

```
    $func = "come";              //声明一个变量并赋值
    $func();                     //使用变量函数来调用函数 come()
    $func = "go";                //重新给变量赋值
    $func("jane");               //使用变量函数来调用函数 go()
    $func = "back";              //重新给变量赋值
    $func("jack");               //使用变量函数来调用函数 back()
? >
```

运行结果：

```
谁进门了?
jane 出门了
jack 重新进门
```

说明：可以看到，变量函数的调用是通过改变变量名来实现的，通过在变量名后面加上一对小括号，PHP 将自动寻找与变量名相同的函数，并执行它。如果找不到对应的函数，系统将会报错。为了防止这类错误，可以在调用变量函数之前使用 PHP 的 function_exists（）函数来判断该变量函数是否存在。

【例 4.2-9】

```
<? php
 $action="showstr";
function showstr()
{
    echo "显示字符串";
}
if(function_exists( $action()))          //判断函数是否存在
     $action();                          //实际调用了 showstr()函数
? >
```

注意：变量函数不能用于语言结构，如 echo（）、print（）、unset（）、include（）、require（）、isset（）及类似的语句。

4.3　PHP 变量的作用域

变量的作用域也就是它的生效范围。大部分的 PHP 变量只有一个单独的使用范围，也包含了 include 和 require 引入的文件。当一个变量执行赋值动作后，会随着声明区域位置的差异，而有不同的使用范围。大致上来说变量会依据声明的位置分为局部变量和全局变量两种。

4.3.1　局部变量

局部变量也称为内部变量，是在函数内部声明的变量，局部变量只是局部有效，它的作用域分为两种：

（1）在当前文件主程序中定义的变量，其作用域限于当前文件的主程序，不能在其他文件或当前文件的局部函数中起作用。

（2）在局部函数或方法中定义的变量，其作用域仅限于局部函数或方法，在当前文件主程序、其他函数、其他文件中无法引用。

【例 4.3-1】

```
<? php
 $my_var="test";                                    // $my_var 的作用域仅限于当前主程序
function my_func()
{
     $local_var=123;                                // $local_var 的作用域仅限于当前函数
    echo ' $local_var=' . $local_var ."<br>";       //调用该函数时输出结果值为 123
    echo ' $my_var = ' . $my_var . "<br>";          //调用该函数时输出结果值为空
}
my_func();                                          //调用 my_func()函数
echo ' $my_var=' . $my_var . "<br>";                //输出结果值为"test"
echo ' $local_var=' . $local_var . "<br>";          //输出结果值为空
? >
```

4.3.2 全局变量

全局变量也称为外部变量，是在函数的外部定义的，它的作用域为从变量定义处开始，到本程序文件的末尾。和其他编程语言不同，在 PHP 中，全局变量不是自动设置为可用的。在 PHP 中，由于函数可以视为单独的程序片段，所以局部变量会覆盖全局变量的能见度，因此在函数中并无法直接调用全局变量，如下所示：

```
<? php
 $a=20;       //在函数外部声明一个全局变量 $a 值为 20
 $b=10;       //在函数外部声明一个全局变量 $b 值为 10
function demo(){     //测试在函数内部不能直接使用全局变量 $a 和 $b
echo "运算结果:".( $a+ $b)."<br>";     //相当于在函数内部声明并且没赋值的两个变量
}
demo();
? >
```

运行结果为：

0

说明：两个变量没有赋初始值为 NULL，执行两个空值相加后结果为 0。

注意：在所有函数外部定义的变量，拥有全局作用域 global；在函数内部定义的变量拥有局部作用域 local。全局变量可以被脚本的任何位置访问，但在函数内部访问全局变量要使用 global 关键字。而局部变量只能在函数内部进行访问。

【例 4.3-2】

```
<? php
 $x = 2;
function test( ){
      $y = 3;
     echo  $x;
     echo "<br>";
     echo  $y;
     echo "<br>";
}
test( );
echo  $x;
echo "<br>";
echo  $y;
? >
```

运行结果为：

3

2

说明：上面的代码中，当调用 test（）函数时，会输出两个变量的值，函数会输出局部变量 $y 的值，但是不能输出全局变量 $x 的值。而在函数外输出的两个变量的值中，会输出变量 $x 的值，但是不能输出 $y 的值，因为 $y 是局部变量，不能在函数外部访问。

在函数内部访问全局变量，要使用 global 关键字。

【例 4.3-3】

```
<? php
 $x = 1;
function test( ){
     global  $x;
      $y = 2;
     echo  $x+ $y;
}
test( );              //输出 3
? >
```

4.3.3　静态变量

静态变量的特点：

（1）不会随着函数的调用和退出而发生变化，不过，尽管该变量还继续存在，但不能使用它。当再次调用定义它的函数时，它又可继续使用，而且保存了前次被调用后留下的值。

（2）静态局部变量只会初始化一次。

（3）静态属性只能被初始化为一个字符值或一个常量，不能使用表达式。即使局部静态变量定义时没有赋初值，系统也会自动为其赋初值 0（对数值型变量）或空字符（对字符变量）。

（4）当多次调用一个函数且要求在调用之间保留某些变量的值时，可考虑采用静态局部变量。虽然用全局变量也可以达到上述目的，但全局变量有时会造成意外的副作用，因此仍以采用局部静态变量为宜。

另外，局部变量从存储方式上可以分为动态存储类型和静态存储类型。函数中的局部变量，如果不专门声明为 static 存储类别，默认都是动态地分配存储空间的。其中的内部动态变量在函数调用结束后自动释放。如果希望在函数执行后，其内部变量依然保存在内存中，应使用静态变量。在函数执行完毕后，静态变量并不会消失，在所有对该函数的调用之间共享，即在函数再次执行时，静态变量将继续前次的结果进行运算。并且静态变量仅在脚本的执行期间函数第一次被调用时被初始化。要声明函数变量为静态的，用关键字 static。比较以下两个程序：

程序 1

```
<? php
function test( ){
    $x = 0;          //没有加 static 关键字时
    echo $x;
    $x++;

}
test( ); //输出 0
test( ); //输出 0
test( ); //输出 0
? >
```

在程序 1 中，函数 test（）中的局部变量 $x，没有使用 static 关键字时，它是一个动态变量，当程序执行 $x++时，它的结果会被存放在内存中，但程序结束后它的结果会被释放，变回原来的初始状态。

程序 2

```
<? php
function test( ){
    static $x = 0; //加 static 关键字时
    echo $x;
    $x++;

}
test( ); //输出 0
test( ); //输出 1
test( ); //输出 2
```

```
? >
```

在程序 2 中，函数 test（）中使用 static 关键字定义局部变量 $x 为静态变量，并赋初值为 0。函数第一次执行时，静态变量 $x 经运算后，从初值 0 变为 1。函数第一次执行完毕后，静态变量 $x 并没有被释放，而函数将结果保存在静态内存中。函数第二次执行时，$x 从内存中获取上一次计算的结果 1，继续运算，并将结果 2 存于静态内存中。以后每次函数执行时，静态变量将从自己的内存空间中获取前一次的存储结果并以此为初值进行计算。

4.4　综 合 案 例

【例 4.4-1】 设计一个计算机器程序，实现简单的加、减、乘、除运算。

新建 EX4.4-1.php 文件，输入以下代码：

```
<html>
<body>
<form method=post>
<table align=center>
<tr><td>简单的计算器</td></tr>
<tr><td><input type="text" size="4" name="number1">
    <select name="caculate">
    <option value="+">+
    <option value="-">-
    <option value="*">*
    <option value="/">/
    </select>
    <input type="text" size="4" name="number2">
    <input type="submit" name="ok" value="计算">
    </td>
</tr>
</table>
</form>
</body>
</html>
<? php
function cac( $a, $b, $caculate)
{
    if( $caculate==="+")
        return $a+$b;
    if( $caculate==="-")
        return $a-$b;
```

```
        if( $caculate==" * " )
            return $a * $b;
        if( $caculate=="/" )
        {
            if( $b=="0" )
                echo "除数不能等于 0";
            else
                return $a/ $b;
        }
    }
    if(isset( $_POST['ok'] ))
    {
        $number1= $_POST['number1'];
        $number2= $_POST['number2'];
        $caculate= $_POST['caculate'];
        if(is_numeric( $number1)&&is_numeric( $number2))
        {
            $answer=cac( $number1, $number2, $caculate);
            echo "<script>alert('". $number1. $caculate. $number2."=". $answer."')</script>";
        }
        else
            echo "<script>alert('输入的不是数字!')</script>";
    }
?>
```

本章小结

本章主要学习了函数的定义以及函数的调用方法，自定义函数的传递方式，并且对递归函数、变量函数进行了讲解，对变量的作用域以及变量的生存周期进行了介绍。

习　题

一、写出相关 PHP 代码的输出结果

(1)

```
function m( $val){
    ++ $val;
}
$val = 10;
```

```
m( $val);
echo $val;
```

(2)

```
function get_arr( $arr){
    unset( $arr[0]);
}
 $arr1 = array(1, 2);
get_arr( $arr1);
echo count( $arr1);//count 个数
```

(3)

```
function sum(... $numbers) {
     $acc = 0;
    foreach ( $numbers as $n) {
        $acc += $n;
    }
    return $acc;
}
echo sum(1, 2, 3, 4, 5);
```

二、编程题

编写用户定义函数 is_leap ( ), 用于测试参数 ($year) 指定的年份是否是闰年, 并返回一个布尔型的值。参数 ($year) 的默认值是 2000。

第 5 章

PHP 数组

【本章内容简介】

数组是 PHP 中重要的数据类型之一。利用数组可以对大量性质相同的数据进行存储、排序、插入及删除等操作，从而提高程序开发效率，并改善程序代码的编写方式。PHP 凭借其代码开源、升级速度快等特点，对数组的操作能力更为强大，尤其是提供了大量方便、易懂的数组操作函数。本章主要介绍 PHP 数组的相关知识。

【本章学习要求】

通过本章的学习，要求了解数组的分类和处理，掌握数组的输出和遍历方法，掌握与数组常用操作相关的函数的应用以及了解 PHP 预定义数组。

5.1 数 组 概 述

变量是一个只能存放单个数值的容器。然而，PHP 有几类变量，可以在一个变量中同时存储多个值，比如数组。

5.1.1 数组的基本概念

数组是一组数据的集合，它将数据按照一定规则排列起来，形成一个可操作的整体。数组中的每一个数据称为“元素”，元素之间相互独立，每个元素相当于一个变量，元素依靠“键”（C 语言称之为“下标”）来识别。

5.1.2 数组的分类

PHP 将数组分为一维数组、二维数组和多维数组，但不管是一维数组还是多维数组，都可以将其分为数字索引数组（indexed array）和关联数组（associative array）。

1. 数字索引数组

数字索引数组的键名（下标）由数字组成，默认从 0 开始，每个数字对应数组元素在数组中的位置。

2. 关联数组

关联数组的键名（下标）可以由数值和字符串混合组成。

5.2　数组的处理

PHP 中创建数组的方法很多，本节主要介绍数组的创建与初始化、键名与值的操作。

5.2.1　数组的创建与初始化

PHP 中，若数组元素只是普通值如数字、字母时，这个数组就是一维数组。数组元素是另外的类型时，这个数组就是多维数组。每个数组元素都有一个相关的索引（也称为关键字），使用索引就可以访问数组元素。创建数组一般有以下几种方法。

1. 使用 array（）函数创建数组

PHP 中的数组可以是一维数组，也可以是多维数组。创建数组可以使用 array（）函数，语法格式如下：

```
array array([ $keys=>] $values,...)
```

“ $keys=> $Values”，定义了关键字的键名和值，自定义键名可以是字符串或数字，如果省略了键名，会自动产生从 0 开始的整数作为键名。如果对某个给出的值没有指定键名，则取该值前面最大的整数键名加 1 后的值。例如：

```
<? php
 $array1=array(1,2,3,4);                                    //定义不带键名的数组
 $array2=array("color"=>"blue","name"=>"picture","number"=>"01");
                                                            //定义带键名的数组
 $array3=array(1=>2,2=>4,5=>6,8,10);                        //定义省略某些键名的数组
? >
```

说明：数组 $array1 的键名是自动产生的从 0 开始的整数，键名分别是 0、1、2、3。$array2 的键名是“color”、“name”、“number”。$array3 中值“8”和“10”的键名省略，数组会自动给其分配键名，值“8”的键名是该值前面最大的整数键名加 1，即 6，值“10”的键名为 7。

为了更好地理解数组的键名和值，这里介绍一个打印函数 print_r（ ）。这个函数用于打印一个变量的信息。如果给出的是字符串型、整型或浮点型的变量，将打印变量值本身。如果给出的是数组类型的变量，将会按照一定格式显示键名和值，对象类型与数组类似。

print_r（ ）函数的语法格式如下：

```
bool print_r(mixed expression [, bool return])
```

如果想捕捉 print_r（）的输出，可使用 return 参数。若此参数设为 TRUE，print_r（ ）将不打印结果，而是返回其输出（此为默认动作）。例如：

```
<? php
```

```
    $array=array("a"=>5, "b"=>10, 20);
    print_r( $array);
?>
```

输出结果为：

Array ([a] => 5 [b] => 10 [0] => 20)

注意：

如果定义了两个完全一样的键名，则后面一个会覆盖前面一个。例如：

```
<? php
    $array=array(1, 1, 1, 1, 8 => 1, 4 => 1, 19, 3 => 13);
    print_r( $array);
? >
```

输出结果为：

Array ([0] => 1 [1] => 1 [2] => 1 [3] => 13 [4] => 1 [8] => 1 [9] => 19)

说明：键名 3 被定义了两次，保留了最后的值 13。

数组创建完后，要使用数组中某个值，可以使用 $array ["键名"] 的形式。如果数组的键名是自动分配的，则默认情况下 0 元素是数组的第一个元素。例如：

```
<? php
    $array1=array("黄色","蓝色","黑色");
    echo  $array1[1];                    //输出"蓝色"
    $array2=array("a"=>5,"b"=>10,"c"=>15);
    echo  $array2["b"];                  //输出 10
? >
```

另外，通过对 array ( ) 函数的嵌套使用，还可以创建多维数组。例如：

```
<? php
    $array=array(
        "color"=>array("红色","蓝色","白色"),
        "number"=>array(1,2,3,4,5,6)
    );                                          //定义二维数组 $array
    echo  $array["color"][2];                   //输出数组元素,输出结果为"白色"
    print_r( $array);                           //打印二维数组
? >
```

输出结果为：

Array ([color] => Array ([0] => 红色 [1] => 蓝色 [2] => 白色)
[number] => Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] => 6))

数组创建之后，可以使用 count ( ) 和 sizeof ( ) 函数获得数组元素的个数，参数是要进行计数的数组。例如：

```
<? php
```

```
$array=array(1,2,3,6=>7,8,9,5,10);
echo count( $array);                    //输出 8
echo sizeof( $array);                   //输出 8
? >
```

2. 使用变量建立数组

通过使用 compact（）函数，可以把一个或多个变量，甚至数组，建立成数组元素，这些数组元素的键名就是变量的变量名，值是变量的值。语法格式如下：

```
array compact(mixed  $varname [ , mixed ...])
```

每个参数 $varname 可以是一个包括变量名的字符串或者是一个包含变量名的数组。对每个参数，compact（）在当前的符号表中查找该变量名并将它添加到输出的数组中，变量名成为键名而变量的内容成为该键的值。任何没有变量名与之对应的字符串都被略过。例如：

```
<? php
    $num=10;
    $str="string";
    $array=array(1,2,3);
    $newarray=compact("num","str","array");          //使用变量名创建数组
    print_r( $newarray);
? >
```

输出结果为：

Array（［num］=> 10　［str］=> string　［array］=> Array（［0］=> 1［1］=> 2［2］=> 3））

与 compact（）函数相对应的是 extract（）函数，其作用是将数组中的单元转化为变量，例如：

```
<? php
    $array=array("key1" =>1, "key2" =>2, "key3" =>3);
    extract( $array);
    echo " $key1  $key2  $key3";                    //输出 1 2 3
? >
```

3. 使用两个数组创建一个数组

使用 array_combine（）函数可以使用两个数组创建另外一个数组，语法格式如下：

```
array array_combine(array  $keys, array  $values)
```

array_combine（）函数用来自 $keys 数组的值作为键名，用来自 $values 数组的值作为相应的值，最后返回一个新的数组。例如：

```
<? php
    $a=array('green','red','yellow');
    $b=array('avocado','apple','banana');
    $c=array_combine( $a, $b);
    print_r( $c);
```

```
//输出:Array ( [green] => avocado [red] => apple [yellow] => banana )
? >
```

4. 建立指定范围的数组

使用 range（）函数可以自动建立一个值在指定范围的数组，语法格式如下：

```
array range(mixed $low, mixed $high [, number $step])
```

$low 为数组开始元素的值，$high 为数组结束元素的值。如果 $low> $high，则序列将从 $high 到 $low。$step 是单元之间的步进值，$step 应该为正值，如果未指定则默认为 1。range（）函数将返回一个数组，数组元素的值就是从 $low 到 $high 之间的值。例如：

```
<? php
    $array1 = range(1,5);
    $array2 = range(2,10,2);
    $array3 = range("a","e");
    print_r( $array1);
    //输出:Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )
    print_r( $array2);
    //输出:Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 [4] => 10 )
    print_r( $array3);
    //输出:Array ( [0] => a [1] => b [2] => c [3] => d [4] => e )
? >
```

5. 自动建立数组

数组还可以不用预先初始化或创建，在第一次使用它的时候，数组就已经创建，例如：

```
<? php
    $arr[0] = "a";
    $arr[1] = "b";
    $arr[2] = "c";
    print_r( $arr);              //输出:Array ( [0] => a [1] => b [2] => c )
? >
```

说明：以上代码创建了 $arr 的数组，在第一句代码运行时，如果 $arr 数组不存在，则自动创建一个只有一个元素的 $arr 数组，后续的代码将在这个数组中添加值。

5.2.2 键名和值的操作

1. 存在性检查

检查数组中是否存在某个键名和值可以使用 array_key_exists（）和 in_array（）函数，array_key_exists（）和 in_array（）函数都为布尔型，存在则返回 TRUE，不存在则返回 FASLE。例如：

```
<? php
    $array = array(1,2,3,5=>4,7=>5);
    if( in_array(5, $array))                    //判断是否存在值 5
```

```
        echo "数组中存在值:5";                    //输出"数组中存在值:5"
    if(! array_key_exists(3, $array))             //判断是否不存在键名 3
        echo "数组中不存在键名:3";                //输出"数组中不存在键名:3"
? >
```

说明：isset（）函数也可以用来检查数组中的键名是否存在，但如果检查的键名对应的值为 NULL 时，isset（）函数返回 FALSE，而 array_key_exisit（）函数则返回 TRUE。

array_search（）函数也可以用于检查数组中的值是否存在，与 in_array（）函数不同的是，in_array（）函数返回的是 TRUE 或 FALSE，而 array_search（）函数当值存在时返回这个值的键名，若值不存在则返回 NULL。例如：

```
<? php
    $array=array(1, 2, 3, "a", 5, "b");
    $key=array_search("a", $array);           //查找"a"是否在数组 $array 中
    if( $key==NULL)                           //如果返回结果为 NULL 则不存在
    {
        echo "数组中不存在这个值";            //不输出
    }
    else
        echo  $key;                           //输出 3
? >
```

2. 获取和输出

使用 array_keys（）和 array_values（）函数可以取得数组中所有的键名和值，并保存到一个新的数组中。例如：

```
<? php
    $arr=array("red"=>"红色","blue"=>"蓝色","white"=>"白色");
    $newarr1=array_keys( $arr);               //取得数组中的所有键名
    $newarr2=array_values( $arr);             //取得数组中的所有值
    print_r( $newarr1);
    //输出结果:Array ( [0] => red [1] => blue [2] => white )
    print_r( $newarr2);
    //输出结果:Array ( [0] => 红色 [1] => 蓝色 [2] => 白色 )
? >
```

使用 key（）函数可以取得数组当前单元的键名，例如：

```
<? php
    $array=array("a"=>1, "b"=>2, "c"=>3, "d"=>4);
    echo key( $array);                        //输出"a"
    next( $array);                            //将数组中的内部指针向前移动一位
    echo key( $array);                        //输出"b"
? >
```

说明：next（）函数返回数组下一个单元的值，并将数组中的内部指针向前移动一位，如果已经到了数组末端，则返回 FALSE。prev（）函数的作用与 next（）函数相反，

是将数组中的内部指针向后移动一位。

另外，“end（$array）;”表示将数组中的内部指针指向最后一个单元；“reset（$array）;”表示将数组指针指向第一个单元，即重置数组的指针；“each（$array）”表示返回当前的键名和值，并将数组指针向下移动一位，这个函数非常适合在数组遍历时使用。

使用 list（）函数可以将数组中的值赋给指定的变量。这样就可以将数组中的值输出显示出来了，这个函数在数组遍历的时候将非常有用。例如：

```
<? php
    $arr=array("红色","蓝色","白色");
    list( $red, $blue, $white)= $arr;          //将数组 $arr 中的值赋给 3 个变量
    echo $red;                                 //输出"红色"
    echo $blue;                                //输出"蓝色"
    echo $white;                               //输出"白色"
? >
```

5.3 数组的输出和遍历

数组主要是对多个相互关联的数据进行批量处理，一般是对整个数组进行统一管理，比如数组的输出和遍历。

5.3.1 数组的输出

在 PHP 中，要对数组某一元素进行输出，通常使用 echo 和 print 语句；要将数组结构输出，可以通过 print_r（）函数来实现。其语法格式如下：

```
bool print_r(mixed $expression)
```

如果该函数的参数 expression 为普通的字符型、整型或实型变量，则直接输出该变量；如果该参数为数组，则按键值和元素的顺序输出数组中的所有元素。

【例 5.3-1】 使用 print_r（）输出二维数组。

新建 EX5.3-1.php 文件，输入以下代码：

```
<? php
    $str = array (
        "办公应用" =>array ("Word","Excel","Powerpoint"),
        "平面设计" =>array ("m" =>"Photoshop","n" =>
        "CorelDRAW","o" =>"Illustrator"),
        "Web 开发" =>array ("PHP",8=>"ASP.NET","JSP") );     //定义二维数组
    print_r ( $str) ;                                        //输出数组
? >
```

运行结果如图 5-1 所示。

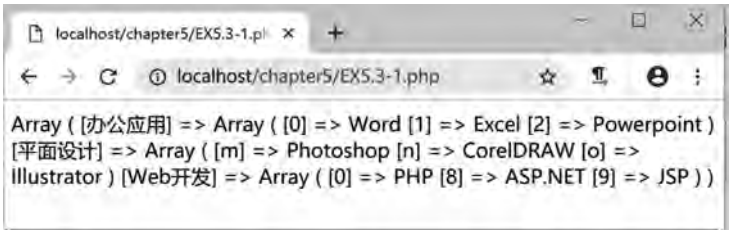


图 5-1　使用 print_r（ ）输出二维数组

说明：当数组的元素是变量时，则称此数组是一维数组；当数组的元素是一维数组时，则称此数组是二维数组。

5.3.2　数组的遍历

在实际应用中，遍历数组是最常用的访问数组的方法。进行数组的遍历要使用到循环控制语句，如 while、for、foreach 等。

1. 使用 while 循环遍历数组

while 循环、list（） 和 each（） 函数结合使用就可以实现对数组的遍历。list（） 函数的作用是将数组中的值赋给变量，each（） 函数的作用是返回当前的键名和值，并将数组指针向下移动一位。例如：

```
<? php
    $arr=array(1,2,3,4,5,6);
    while(list( $key, $value)= each( $arr))
    //直到数组指针到数组尾部时停止循环
    {
        echo  $value;                    //输出 123456
    }
? >
```

如果数组是多维数组（假设为二维数组），则在 while 循环中多次使用 list（） 函数。

【例 5.3-2】 使用 while 循环遍历二维数组。

新建 EX5.3-2.php 文件，输入以下代码：

```
<? php
    $t_array=array(
        array("081101","王林","计算机"),
        array("081102","程明","计算机"),
        array("081210","李红庆","通信工程")
    );
    //以表格形式输出数组的值
    echo "<table border=1><tr><td>学号</td><td>姓名</td><td>专业</td></tr>";
    while(list( $key, $value)= each( $t_array))
```

```
    {
        //将二维数组中的单个数组中的值用变量替换
        list( $XH, $XM, $ZY)= $value;
        //输出变量的值
        echo "<tr><td> $XH</td><td> $XM</td><td> $ZY</td></tr>";
    }
    echo "</table>";                    //输出表格结尾
? >
```

运行结果如图 5-2 所示。

图 5-2　输出表格

2. 使用 for 循环遍历数组

for（）循环是通过数组的下标来访问数组中的元素，并且必须保证下标是连续的数字索引。在 PHP 中，数组不仅可用非连续数字作为下标，还可用字符串作为下标，此时就不能用 for（）循环来遍历数组了。

【例 5.3-3】 使用 for 循环遍历数组。

```
<? php
    $arr = array('ccy','yyy','wyq','dyl');          //定义数组变量
    for ( $i = 0; $i< count( $arr); $i++){
        $str= $arr[ $i];                            //将数组元素赋值给变量 $str
        echo "Her name is $str.<br>";               //循环显示“Her name is $str.”
    }
? >
```

运行结果如图 5-3 所示。

图 5-3　使用 for 循环遍历数组

3. 使用 foreach 循环访问数组

foreach 循环是一个专门用于遍历数组的循环，语法格式如下：

```
foreach (array_expression as $value)
    //代码段
foreach (array_expression as $key => $value)
    //代码段
```

第一种格式遍历给定的 array_expression 数组，每次循环中，当前单元的值被赋给变量 $value 并且数组内部的指针向前移一步（因此下一次循环将会得到下一个单元）。第二种格式做同样的事，只是当前单元的键名也会在每次循环中被赋给变量 $key。例如：

```
<? php
    $color=array("a"=>"red","blue","white");
    foreach( $color as $value)
    {
        echo $value."<br/>";                    //输出数组的值
    }
    foreach( $color as $key=>$value)
    {
        echo $key."=>". $value."<br/>";         //输出数组的键名和值
    }
? >
```

输出结果为：

```
red
blue
white
a=>red
0=>blue
1=>white
```

【例 5.3-4】在页面上生成 5 个文本框，用户输入学生成绩，提交表单后输出其中分数小于 60 分的值，并计算平均成绩后输出。

新建 EX5.3-4.php 文件，输入以下代码：

```
<? php
    echo "<form method=post>";                  //新建表单
    for( $i=1; $i<6; $i++)                      //循环生成文本框
    {
        //文本框的名字是数组名
        echo "学生".$i."的成绩:<input type=text name='stu[]' ><br/>";
    }
    echo "<input type=submit name=bt value='提交'>";  //提交按钮
    echo "</form>";
    if(isset( $_POST['bt']))                    //检查提交按钮是否被按下
```

```
{
    $sum=0;                                    //总成绩初始化为 0
    $k=0;
    $stu= $_POST['stu'];                       //取得所有文本框的值并赋予数组 $stu
    $num=count( $stu);                         //计算数组 $stu 元素个数
    echo "您输入的成绩有:<br/>";
    foreach( $stu as  $score)                  //使用 foreach 循环遍历数组 $stu
    {
        echo  $score."<br/>";                  //输出接收的值
        $sum= $sum+ $score;                    //计算总成绩
        if( $score<60)                         //判断分数小于 60 的情况
        {
            $sco[ $k]= $score;                 //将分数小于 60 的值赋给数组 $sco
            $k++;                              //数组 $sco 的键名索引加 1
        }
    }
    echo "<br/>低于 60 分的成绩有:<br/>";
    for( $k=0; $k<count( $sco); $k++)          //使用 for 循环输出 $sco 数组
        echo  $sco[ $k]."<br/>";
    $average= $sum/ $num;                      //计算平均成绩
    echo "<br/>平均分为: $average";            //输出平均成绩
}
```

运行 EX5.3-4.php 文件，在文本框中依次输入成绩：42，64，78，95，55，单击【提交】按钮，运行结果如图 5-4 所示。

图 5-4　处理学生成绩

说明：代码中求分数低于 60 分的成绩时，从总的数组中取得低于 60 的值保存在一个新数组中，并输出该数组。

5.4　数组的排序

数组中的值经常是混乱地排放的，所以需要使用排序函数来排序，使其易于访问和操作。PHP 提供了很多数组排序函数，特别是对一维数组的排序非常简单。

5.4.1　升序排序

1. sort（）函数

使用 sort（）函数可以对已经定义的数组进行排序，使得数组单元按照元素值从低到高重新索引。语法格式如下：

```
bool sort(array  $array [ , int  $sort_flags])
```

说明：sort（）函数如果排序成功返回 TRUE，失败则返回 FALSE。两个参数中 $array 是需要排序的数组；$sort_flags 的值可以影响排序的行为，$sort_flags 可以取以下 6 个值。

SORT_REGULAR：正常比较单元（不改变类型），这是默认值。

SORT_NUMERIC：单元被作为数字来比较。

SORT_STRING：单元被作为字符串来比较。

SORT_LOCALE_STRING：根据当前的区域设置把单元当作字符串比较。

SORT_NATURAL：对每个单元以“自然的顺序”对字符串进行排序（PHP 5.4.0 中新增）。

SORT_FLAG_CASE：能够与 SORT_STRING 或 SORT_NATURAL 合并（OR 位运算），不区分大小写排序字符串（PHP 5.4.0 中新增）。

sort（）函数不仅对数组进行排序，同时删除了原来的键名，并重新分配自动索引的键名，例如：

```
<? php
    $array1=array("a"=>5, "x"=>3, 5=>7, "c"=>1);
    $array2=array(2=>"c",4=>"a",1=>"b");
    if(sort( $array1))
        print_r( $array1);
        //输出:Array ([0] => 1 [1] => 3 [2] => 5 [3] => 7 )
    else
        echo "排序\ $array1 失败";                    //不输出
    if(sort( $array2))
        print_r( $array2);
        //输出:Array ([0] => a [1] => b [2] => c )
? >
```

注意：在对含有混合类型值的数组排序时要小心，因为这可能会产生不可预知的结果。

2. asort（）函数

asort（）函数也可以对数组的值进行升序排序，语法格式和 sort（）类似，但使用 asort（）函数排序后的数组还保持键名和值之间的关联，例如：

```
<? php
    $fruits = array("d" =>"lemon", "a" =>"orange", "b" =>"banana", "c" =>"apple");
    asort( $fruits);
    print_r( $fruits);
    //输出:Array ( [c] => apple [b] => banana [d] => lemon [a] => orange )
? >
```

3. ksort（）函数

ksort（）函数用于对数组的键名进行排序，排序后键名和值之间的关联不改变，例如：

```
<? php
 $fruits = array("d" =>"lemon", "a" =>"orange", "b" =>"banana", "c" =>"apple");
ksort( $fruits);
print_r( $fruits);
    //输出:Array ( [a] => orange [b] => banana [c] => apple [d] => lemon )
? >
```

5.4.2 反向排序

前面介绍的 sort（）、asort（）、ksort（）这 3 个函数都是对数组按升序排序。而它们都对应有一个反向排序的函数，可以使数组按降序排序，分别是 rsort（）、arsort（）、krsort（）函数。

反向排序的函数与升序排序的函数用法相同，rsort（）函数按数组中的值降序排序，并将数组键名修改为一维数字键名；arsort（）函数将数组中的值按降序排序，不改变键名和值之间的关联；krsort（）函数将数组中的键名按降序排序。

【例 5.4-1】对数组元素进行降序排序。

新建 EX5.4-1.php 文件，输入以下代码：

```
<? php
    $numbers = array(3,6,1,28,11,32,46);          //定义数组变量
    echo "<br>数组未排序前元素依次为:<br>";
    foreach ( $numbers as  $a){                   //使用 foreach()循环遍历数组
        echo  $a . "    ";    //输出数组元素
    }
    echo "<hr>";
    sort( $numbers);                              //升序排列数组元素
    echo "<br>数组升序排列后元素依次为:<br>";
```

```
foreach ( $numbers as  $a){
    echo  $a . "    ";    //1 3 6 11 28 32 46
}
sort( $numbers,SORT_STRING);                  //把数组元素作为字符串类型升序排列
echo "<br>数组元素被作为字符串升序排列后依次为:<br>";
foreach ( $numbers as  $a){
    echo  $a . "    ";    //1 11 28 3 32 46 6
}
echo "<hr>";
rsort( $numbers);
echo "<br>数组降序排列后元素依次为:<br>";
foreach ( $numbers as  $a){
    echo  $a . "    ";
}
? >
```

运行结果如图 5-5 所示。

图 5-5　数组降序排序

5.4.3　对多维数组排序

array_multisort（）函数可以一次对多个数组排序，或根据多维数组的一维或多维对多维数组进行排序。语法格式如下：

```
bool array_multisort( array  $ar1 [ , mixed  $arg [ , mixed  $... [ , array  $... ] ] ] )
```

本函数的参数结构比较特别，且非常灵活。第一个参数必须是一个数组。接下来的每个参数可以是数组或者是下面列出的排序标志。

排序顺序标志：

SORT_ASC 为默认值，按照上升顺序排序。

SORT_DESC 为按照下降顺序排序。

排序类型标志：

SORT_REGULAR 为默认值，按照通常方法比较。

SORT_NUMERIC 为按照数值比较。

SORT_STRING 为按照字符串比较。

注意：每个数组之后不能指定两个同类的排序标志，每个数组后面指定的排序标志仅对该数组有效。

使用 array_multisort（）函数排序时字符串键名保持不变，但数字键名会被重新索引。当函数的参数是一个数组列表时，函数首先对数组列表中的第一个数组进行升序排序，下一个数组中值的顺序按照对应的第一个数组的值的顺序排列，以此类推。例如：

```
<? php
    $ar1 = array(3,5,2,4,1);
    $ar2 = array(8,6,9,7,10);
    array_multisort( $ar1, $ar2);                    //对 $ar1、$ar2 排序
    print_r( $ar1);
    //输出:Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )
    echo "<br/>";
    print_r( $ar2);
    //输出:Array ( [0] => 10 [1] => 9 [2] => 8 [3] => 7 [4] => 6 )
? >
```

说明：第一个数组中的值原先的顺序是 3，5，2，4，1，对应的第二个数组中的值的顺序是 8，7，9，6，10。排序后第一个数组中的值为 1，2，3，4，5，值 1 在第一位，则第二个数组中与值 1 相对应的值 10 也排在第二个数组中的第一位，以此类推，则第二个数组最后的顺序是 10，9，8，7，6。如果在第一个数组中有相同的值，则按照第二个数组中对应的值进行排序，以此类推。这类似于 SQL 中的 ORDER BY 子句——对表的各列以行来排序。

注意：数组列表中所有数组的数组元素一定要相等，否则使用 array_multisort（）函数时会发出警告。

如果要对多维数组排序，原理与对多个数组排序类似。将多维数组中的各个元素看作多个数组。

【例 5.4-2】 多维数组排序。

新建 EX5.4-2.php 文件，输入以下代码：

```
<? php
    //初始化一个二维数组
    $ar = array(
        array("10", 11, 100, 100, "a"),
        array( 1, 2, "2", 3, 1)
    );
    array_multisort( $ar[0], SORT_ASC, SORT_STRING,
```

```
    $ar[1], SORT_NUMERIC, SORT_DESC);
var_dump( $ar);                          //使用 var_dump 函数显示数组信息
? >
```

运行结果如图 5-6 所示。

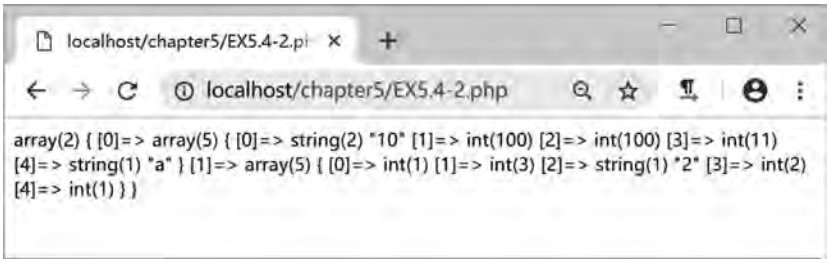


图 5-6　多维数组排序

说明：本例排序后，第一个数组变成"10"，11，100，100，"a"（被当作字符串以升序排列）。第一个数组中有相同的值，所以按照第二个数组的规则排序。第二个数组将变成 1，2，"2"，3，1（被当作数字以降序排列）。

5.4.4　对数组重新排序

1. shuffle（）函数

该函数的作用是将数组用随机的顺序排列，并删除原有的键名，建立自动索引。例如：

```
<? php
    $arr=range(1,10);                    //产生有序数组
    foreach( $arr as  $value)
        echo  $value. "";
        //输出有序数组,结果为 1 2 3 4 5 6 7 8 9 10
    echo "<br/>";
    shuffle( $arr);                      //打乱数组顺序
    foreach( $arr as  $value)
        echo  $value. "<br/>";
    //输出新的数组顺序,每次运行,结果都不一样
? >
```

2. array_reverse（）函数

该函数的作用是将一个数组单元按相反顺序排序，语法格式如下：

```
array array_reverse(array  $array [ , bool  $preserve_keys ])
```

如果 $preserve_keys 值为 TRUE 则保留原来的键名，为 FALSE 则为数组重新建立索引，默认为 FALSE。例如：

```
<? php
    $array=array("a" =>1,2,3,4);
```

```
    $ar1 = array_reverse( $array);
    $ar2 = array_reverse( $array,TRUE);
    print_r( $ar1);        //输出:Array ( [0] => 4 [1] => 3 [2] => 2 [a] => 1 )
    print_r( $ar2);        //输出:Array ( [2] => 4 [1] => 3 [0] => 2 [a] => 1 )
? >
```

5.4.5 自然排序

natsort（）函数实现了一个和人们通常对字母、数字、字符串进行排序的方法一样的排序算法，并保持原有键/值的关联，这被称为“自然排序”。natsort（）函数对大小写敏感，它与 sort（）函数的排序方法不同。例如：

```
<? php
    $array1 = $array2 = array("img12", "img10", "img2", "img1");
    sort( $array1); //使用 sort 函数排序
    print_r( $array1);
    //输出:Array ( [0] => img1 [1] => img10 [2] => img12 [3] => img2 )
    natsort( $array2); //自然排序
    print_r( $array2);
    //输出:Array ( [3] => img1 [2] => img2 [1] => img10 [0] => img12 )
? >
```

说明：从输出结果可以看出，自然排序符合人们通常对数字、字符串排序的习惯。另外还有一个自然排序函数 natcasesort（ ），这个函数与 natsort（）函数用法相同，只是不区分大小写。

5.5 数组常用操作

由于数组的灵活性和方便性，其在 PHP 编程中经常被用到。与数组操作相关的函数有很多，下面介绍一些常用函数。

5.5.1 数组计算

使用 count（）、sizeof（）函数可以计算数组中的元素个数，而使用 array_count_values（）函数可以计算数组中一个值出现的次数。语法格式如下：

```
array array_count_values(array  $input)
```

array_count_values（）函数返回一个数组，该数组用 $input 数组中的值作为键名，以该值在 $input 数组中出现的次数作为值。例如：

```
<? php
    $arr = array(1,2,3,1,3,1,4,1,1,4,2);
    $result = array_count_values( $arr);
```

```
    print_r( $result);
    //输出:Array ( [1] => 5 [2] => 2 [3] => 2 [4] => 2 )
? >
```

使用 array_product （） 函数 （PHP 5. 1. 0） 计算数组中所有值的乘积，语法格式如下：

```
number array_product ( array $array )
```

array_product （） 函数以整数或浮点数返回一个数组中所有值的乘积。例如：

```
<? php
    $a=array(2, 4, 6, 8);
    echo "product(a) = ". array_product( $a) . "\n";
    //输出:product(a) = 384
? >
```

5. 5. 2　数组的合并与拆分

在程序开发过程中，通常需要将多个数组合并为一个数组，或者将某个数组中的一部分取出构成一个新的数组，此时可以使用数组的合并与拆分函数。

1. 合并一个或多个数组——array_merge （） 函数

该函数的功能是把一个或多个数组合并为一个新数组。其语法格式如下：

```
array array_merge (array $array1[, array $...])
```

该函数返回合并后的新数组，第 2 个数组中的值附加在前一个数组值的后面，往后依次排列。如果两个或更多个数组元素有相同的字符串键名，则最后的元素会覆盖前面的元素；如果是数字键名，则后面的值不会覆盖原来的值，而是附加到后面，且合并后的数组键名将会以连续的方式重新进行键名索引。

如果仅向 array_merge （） 函数输入一个数组，且键名是整数，则该函数将返回带有整数键名的新数组，其键名以 0 开始进行重新索引。

【例 5. 5-1】 合并数组。

新建 EX5. 5-1. php 文件，输入代码如下：

```
<? php
    $a1 = array("a" =>"red","b" =>"green");
    //定义键值为字符串的数组 $a2 = array("c" =>"blue","b" =>"yellow");
    $a3 = array("c" =>"blue","d" =>"yellow");
    $a4= array("3" =>"blue","5" =>"yellow","7" =>"red","9" =>"green");
    //键值为整数的数组
    echo "<br>合并数组含有相同的字符串键值:<br>";
    print_r(array_merge( $a1, $a2));
    //Array ( [a] => red [b] => yellow [c] => blue )
    echo "<br><br>合并数组没有相同的字符串键值:<br>";
    print_r(array_merge( $a1, $a3));
    //Array ( [a] => red [b] => green [c] => blue [d] => yellow )
    echo "<br><br>合并键名为整数的单个数组:<br>";
```

```
    print_r( array_merge( $a4) );
    //Array ( [0] => blue [1] => yellow [2] => red [3] => green )
? >
```

运行结果如图 5-7 所示。

图 5-7　合并数组

2. 从数组中取出一段序列——array_slice（）函数

array_slice（）函数在数组中根据条件取出一段序列。其语法格式如下：

```
array array_slice( array $array, int $start [ , int $length [ ,bool $preserve_keys] ])
```

该函数返回根据 start 和 length 参数所指定的 array 数组中的一段序列，star 规定取出数组子集的开始位置，如果该值为正数，则从前往后开始取；如果该值为负数，则从后向前取 start 绝对值。“-2”表示从数组的倒数第二个元素开始。

可选参数 length 规定取出数组元素的个数。如果该值为正数，则返回该数量的元素；如果该值为负数，则函数将在距离数组末端 length 远的地方终止取出。如果没有设置该值，则返回从 start 参数设置的位置开始直到数组末端的所有元素。

可选参数 preserve_keys 规定函数是保留键名还是重置键名。当其值为 true 时，表示保留键名；当其值为 false（默认）时，表示重置键名。

3 将数组中的选定元素用其他元素替代——array_splice（）函数

array_splice（）函数从数组中移除选定的元素，并用新元素替代。该函数返回移除或替换元素后的数组。其语法格式如下：

```
array array_splice ( array & $array, int $start [ ,int $length [ ,mixed $replacement] ])
```

该函数把 array 数组中由 start 和可选参数 length 指定的元素去掉，如果定义了可选参数 replacement，则用 replacement 数组中的元素取代，返回最后生成的数组。其中 array 中的数字键名不被保留。

注意：如果函数没有移除任何元素（length = 0），则将从 start 参数的位置插入被替换数组。

【例 5.5-2】 拆分数组。

新建 EX5.5-2.php 文件，输入代码如下：

```
<? php
    $a1 = array("a" =>"red","b" =>"green","c" =>"blue","d" =>"yellow");
    $a2 = array("a" =>"purple","b" =>"orange");
    echo "<br> 数组默认为:<br>";
    foreach ( $a1 as $key => $a){                    //使用 foreach()循环遍历数组
        echo " $key => $a \n";                       //输出数组元素
    }
    echo "<hr>";
    echo "从数组中取出一部分元素:<br>";
    print_r(array_slice( $a1,1,2));                  //Array ( [b] => green [c] => blue )
    echo "<br>";
    print_r(array_slice( $a1,-2,1));                 //Array ( [c] => blue )
    echo "<hr>";
    echo "将数组中的选定元素用其他元素替代:<br>";
    print_r(array_splice( $a1,0,2, $a2));            //Array ( [a] => red [b] => green )
        echo "<hr>";
    echo "被替代后的数组元素:<br>";
    foreach ( $a1 as $key => $a){                    //使用 foreach()循环遍历数组
        echo " $key => $a\n";                        //输出数组元素
    }
? >
```

运行结果如图 5-8 所示。

图 5-8　拆分数组

5.6 预定义数组

除自定义数组外，PHP 还提供了一组预定义数组，这些数组获取来自 Web 服务器、运行环境和用户输入的数据等信息。这些数组在全局范围内自动生效，也被称为自动全局变量或者超全局变量。

常用预定义数组见表 5-1。

表 5-1 常用预定义数组

数 组	说 明
$_SERVER [ ]	获取服务器和客户配置及当前请求环境有关的信息。如 $_SERVER ['REMOTE_ADDR'] 获取浏览当前页面的客户 IP 地址
$_GET [ ]	获取用 GET 方法传递的参数的有关信息
$_POST [ ]	获取用 POST 方法传递的参数的有关信息
$_COOKIE [ ]	获取和设置当前网站的 Cookie 标识
$_FILES [ ]	获取通过 POST 方法向服务器上传的数据的有关信息
$_ENV [ ]	PHP 解析所在服务器环境的有关信息
$_REQUEST [ ]	记录通过各种方法传递给脚本的变量，特别是 GET，POST 和 COOKIE
$_SESSION [ ]	存储与所有会话变量有关的信息
$GLOBALS [ ]	包含全局作用域内的所有变量

5.7 综 合 案 例

本节通过一个综合案例来说明 PHP 数组常用函数的使用。

【例 5.7-1】 接收用户输入的学生学号、姓名、成绩等信息，将接收到的信息存入数组并按照成绩升序排序。之后再以表格的形式输出，如果存在学号为 081101 的学生，则输出其姓名与成绩。

新建 EX5.7-1.php 文件，输入以下代码：

```
<! DOCTYPE html>
<! -- HTML5 表单 -->
<style type="text/css">
table,div,td{
    text-align:center;
}
table{
    margin:0 auto;
```

```
}
p{
    font-size:18px;
    color:#FF0000;
}
</style>
<form name=fr1 method=post>
    <table border=1>
        <tr>
            <td><div>学号</div></td>
            <td><div>姓名</div></td>
            <td><div>成绩</div></td>
        </tr>
<? php
    for( $i=0; $i<5; $i++)                          //循环生成表格的文本框
    {
? >
        <tr>
            <td><input type=text name="XH[ ]"></td>
            <td><input type=text name="XM[ ]"></td>
            <td><input type=text name="CJ[ ]"></td>
        </tr>
<? php }? >
        <tr>
            <td colspan="3">
                <input type="submit" name="bt_stu" value="提交">
            </td>
        </tr>
    </table>
</form>
<p align=center>注意:学号值不能重复</p><br/>
<! -- 以上是输入表单 -->
<? php
    if( isset( $_POST['bt_stu' ]))                  //判断按钮是否被按下
    {
        $XH= $_POST['XH' ];                         //接收所有学号的值存入数组 $XH
        $XM= $_POST['XM' ];                         //接收所有姓名的值存入数组 $XM
        $CJ= $_POST['CJ' ];                         //接收所有成绩的值存入数组 $CJ
        //对以上三个数组排序, $CJ 为首要数组
        array_multisort( $CJ, $XH, $XM);
        for( $i=0; $i<count( $XH); $i++)
```

```
            //将三个数组的值组成一个二维数组 $sum
            $sum[ $i]=array( $XH[ $i], $XM[ $i], $CJ[ $i]);
            echo "<div>排序后成绩表如下:</div>";
            //表格的首部
            echo "<table border=2><tr><td>学号</td><td>姓名</td><td>成绩</td></tr>";
            foreach( $sum as  $value)                    //使用 foreach 循环遍历数组 $sum
            {
                //使用 list( )函数将数组中的值赋给变量
                list( $stu_number, $stu_name, $stu_score)= $value;
                //输出表格内容
echo "<tr><td> $stu_number</td><td> $stu_name</td><td> $stu_score</td></tr>";
            }
            echo "</table><br/>";                        //表格尾部
            reset( $sum);                                //重置 $sum 数组的指针
            while(list( $key, $value)= each( $sum))      //使用 while 循环遍历数组
            {
                list( $stu_number, $stu_name, $stu_score)= $value;
                if( $stu_number=="081101")               //查询是否有学号为 081101 的值
                {
                    echo "<p align=center>";
                    echo  $stu_number."的姓名为:". $stu_name.",";
                    echo "成绩为:". $stu_score;
                    break;                               //找到则结束循环
                }
            }
        }
    ? >
```

运行后在页面的表格文本框中依次输入下列 5 行值：

1716010213　王×　66

1716010220　张×　75

1716010208　李×　92

1716010210　刘×　69

1716010228　郑×发　88

输入后单击“提交”按钮，运行结果如图 5-9 所示。

图 5-9　学生成绩排序

本章小结

本章主要介绍了 PHP 数组的应用。在学完本章内容后，读者应重点掌握以下知识。

（1）数组由一系列有序的变量组成，每个变量都有编号，形成一个可操作的整体。

（2）在 PHP 中创建数组的方式有很多种，读者应该灵活运用。

（3）PHP 支持两种数组：索引数组（indexed array）和关联数组（associative array），前者使用数字作为键（下标），默认索引值从 0 开始；后者使用字符串作为键（下标），也可以是数值和字符串混合的形式。

（4）数组本质上是用来存储、管理和操作一组变量的，PHP 支持一维数组和二维数组。

除自定义数组外，PHP 还提供了一组预定义数组，这些数组获取来自 Web 服务器、运行环境和用户输入的数据等信息。

习　题

一、填空题

1. _____由一系列有序的变量组成，每个变量都有编号，形成一个可操作的整体。

2. PHP 支持两种数组：_____数组（indexed array）和_____数组（associative array），前者使用数字作为键（下标），默认索引值从 0 开始；后者使用字符串作为键（下标），也可以是数值和字符串混合的形式。

3. 一个数组中只要有一个键名不是数字，那么该数组就称为_____数组。

4. 在 PHP 中，要对数组某一元素进行输出，通常使用_____和_____语句；要将数组结构输出，可以通过_____函数来实现。

5. 在 PHP 编程中，经常需要将数组元素转换成字符串，或者将字符串转换成数组元素，使用_____和_____函数可以分别实现上述效果。

6. _____函数可用于统计数组中元素的个数。

7. _____函数可以根据关联数组的值，对数组进行升序排列；_____函数可以根据关联数组的键，对数组进行升序排列。

8. _____函数用于搜索数组中是否存在指定的值。

9. array_key_exists（ ）函数用于检查某个数组中是否存在指定的键名，如果键名存在则返回_____，否则返回_____。

10. _____函数把一个或多个数组合并为一个新数组。

二、编程题

1. 有一数组 $a = array（20，34，12，54，64，43，7），请按由小到大的次序重新排序。

2. 有一数组 $b = array（15，30，45，123，67，98，26，75，6，234），请将其逆序输出。

3. 有一数组 $b = array（115，320，415，23，167，98，36，35，76，134），请找出其最大值和最小值。

第 6 章

字符串处理与正则表达式

【本章内容简介】

在 Web 编程中，经常需要对字符串进行处理和分析。正确使用和掌握字符串相关操作，能在开发过程中节约大量时间，有效提高开发效率。正则表达式是一种对字符串进行模式匹配和替换的规则，在字符串处理中起着非常重要的作用。本章便来学习字符串操作和正则表达式的相关知识。

【本章学习要求】

通过本章的学习，要求了解字符串的组成和表示形式，掌握字符串处理函数的应用，了解正则表达式的作用和语法规则，掌握 Perl 兼容的正则表达式的应用。

6.1 字符串操作

无论哪种语言，字符串操作都是重要的部分。PHP 提供了大量的字符串操作函数，使用简单、功能全面。前面介绍的 echo（ ）和 print（ ）函数就是典型的字符串操作函数。由于 PHP 是弱语言类型，所以当使用字符串操作函数时，其他类型的数据也会被当作字符串来处理。本节将系统地介绍如何操作字符串。

6.1.1 字符串的定义与显示

字符串是由一对单引号（'）或一对双引号（"）包含的零个或多个字符组成的集合。构成字符串的字符主要包括以下几种类型。

- 字母类型：A，B，a，b 等。
- 数字类型：1，2 等。
- 特殊字符：#，*，+，^等。
- 不可见字符：\ n（换行符），\ r（回车符），\ t（Tab 字符）等。

其中，不可见字符用来控制字符串的格式化输出，在浏览器上不可见，只能看见输出的结果。

字符串的界定符号有以下 3 种。

- 单引号（'）。
- 双引号（"）。
- 界定符（<<<）。

字符串的显示可以使用 echo（ ）和 print（ ）函数，这在之前已经介绍过。echo（ ）函数和 print（ ）函数并不是完全一样，二者还存在一些区别：print（ ）具有返回值，返回 1，而 echo（ ）则没有，所以 echo（ ）比 print（ ）要快一些，也正是因为这个原因，print（ ）能应用于复合语句中，而 echo（ ）则不能。例如：

```
<? php
    $return = print "test";                    //输出"test"
    echo  $return;                             //输出 1
? >
```

另外，echo（ ）可以一次输出多个字符串，而 print（ ）则不可以。例如：

```
echo "I", "love", "PHP";                       //输出"IlovePHP"
print "I", "love", "PHP";                      //将提示错误
```

6.1.2 字符串的格式化

除了 echo（ ）、print（ ）函数外，PHP 还提供了一些字符串格式化输出的函数，例如，printf（ ）、sprintf（ ）和 vsprintf（ ）函数。

函数 printf（ ）将一个通过替换值建立的字符串输出到格式字符串中，这个命令和 C 语言中的 printf（）函数结构和功能一致。语法格式如下：

```
int printf(string  $format [ , mixed  $args])
```

第一个参数 $format 是格式字符串，$args 是要替换进来的值，格式字符串里的字符“%”指出了一个替换标记。

格式字符串中的每一个替换标记都由一个百分号组成，后面可能跟有一个填充字符、一个对齐方式字符、字段宽度和一个类型说明符。字符串的类型说明符为“s”。例如：

```
<? php
    //显示字符串
    $str = "hello";
    printf("%s\n", $str);                      //输出"hello"并回车
    printf("%010s\n", $str);                   //在字符串前补 0,将字符串补成 10 位
    //显示数字
    $num = 10;
    printf("%d", $num);                        //输出 10
? >
```

说明：所有的转换说明都以一个“%”开始，如果想打印一个“%”，必须使用“%%”。填充字符表明该字符用于填充结果，使结果为适当大小的字符串，默认情况下使用空格填充；对齐方式字符对字符串和数字有不同的作用，对于字符串，减号“-”使该字符串右对齐（默认为左对齐），对于数字，加号“+”使正数在输出的时候以加号开头；

字段宽度是指字符串应该输出的宽度，如果字符串实际长度小于该值，则使用填充字符填充；类型说明符表示要替换进来的值将要转化的数据类型，如字符串的类型说明符为“s”，十进制整数的类型说明符为“d”，浮点数的类型说明符为“f”，八进制整数的类型说明符为“o”等。

函数 sprintf（ ）所带的参数和 printf（ ）函数一样，但是返回的是内置的字符串，使得可以在变量中存储字符串供以后使用。

函数 vprintf（ ）允许在格式后面用数组作为参数，用法和 printf（ ）函数基本相同。

函数 vsprintf（ ）和函数 vprintf（ ）一样，可以用数组作为参数，但不输出字符串。

6.1.3　常用的字符串函数

1. 获取字符串长度

字符串长度函数包括 strlen（）函数和 mb_strlen（）函数。

（1）strlen（）函数。函数的语法格式如下：

```
int strlen(string  $str)
```

函数功能：返回字符串 $str 的字符长度。

【例 6.1-1】使用 strlen（）函数计算字符串的长度。

新建 EX6.1-1.php 文件，输入以下代码：

```
<? php
      $str="hello 世界";
     echo strlen( $str);
? >
```

程序运行结果：9

提示：对函数 strlen（）来说，当 PHP 采用的字符集为 UTF-8 时，则一个英文字符占一个字节长度，一个汉字占两个字节长度。

（2）mb_strlen（）函数。函数的语法格式如下：

```
int mb_strlen(string  $str,string encoding)
```

函数功能：根据设定的编码方式，返回字符串所包含的字符数；多字节的字符被计为 1。

函数的参数说明见表 6-1。

表 6-1　函数 mb_strlen（ ）的参数说明

参　数	描　述
str	规定要处理的字符串
encoding	可选。规定字符串的编码方式。常用的编码方式如下： GB2312 或 GBK：一个英文字符占一个字节长度，一个汉字占两个字节长度； UTF-8：每个中文或英文字符均占一个字节长度； 默认采用 PHP 内部字符编码 UTF-8

【例 6.1-2】使用函数 mb_strlen（）计算字符串的长度。

新建 EX6.1-2.php 文件，输入以下代码：

```
<? php
    $str="你好 world";
    echo "中文 GB2312 编码方式:",mb_strlen( $str,"GB2312"),"<br/>";
    echo "UTF-8 编码方式:",mb_strlen( $str,"UTF-8"),"<br/>";
? >
```

运行结果如图 6-1 所示。

图 6-1　mb_strlen（）函数

2. 大小写转换

在字符串操作过程中，通常需要对其大小写进行转换，此时可以使用大小写转换函数。常见大小写转换函数及其语法格式如下：

```
string strtolower (string  $str) ;                    //转换为小写
string strtoupper(string  $str) ;                     //转换为大写
string ucfirst(string  $str) ;                        //整个字符串首字母大写
string lcfirst(string  $str) ;                        //整个字符串首字母小写
string ucwords (string  $str) ;                       //整个字符串中每个单词首字母大写
```

【例 6.1-3】大小写转换。

新建 EX6.1-3.php 文件，输入以下代码：

```
<? php
    $str = "We want To FLY";                          //定义字符串类型的变量
    echo strtolower ( $str) . "<br>";                 //输出转换为小写的字符串
    echo strtoupper( $str) . "<br>";                  //输出转换为大写的字符串
    echo ucfirst( $str) . "<br>";                     //输出转换为首字母大写的字符串
    echo lcfirst( $str) . "<br>";                     //输出转换为首字母小写的字符串
    echo ucwords( $str) . "<br>";                     //输出转换为单词首字母大写的字符串
    echo  $str;                                       //输出原字符串
? >
```

运行结果如图 6-2 所示。

图 6-2　大小写转换

3. 去除字符串首尾空格和特殊字符

用户在输入数据时，若无意中输入了多余的空格，则需要去除字符串两端多余的空格和特殊字符。PHP 程序提供了 ltrim（）函数去除字符串左端的空格和特殊字符、rtrim（）函数去除字符串右端的空格和特殊字符、trim（）函数去除字符串两端的空格和特殊字符。它们的语法格式如下：

```
string trim(string $str [, string $charlist])
string rtrim(string $str [, string $charlist])
string ltrim(string $str [, string $charlist])
```

可选参数 $charlist（PHP 4.1.0 引入）是一个字符串，指定要删除的字符，如果省略则默认删除的字符见表 6-2。

表 6-2　trim（）、ltrim（）、rtrim（）函数的默认删除字符

字　符	ASCII 码	意　义
""	32（0x20）	空格
"\t"	9（0x09）	制表符
"\n"	10（0x0A）	换行
"\r"	13（0x0D）	回车
"\0"	0（0x00）	空字节
"\x0B"	11（0x0B）	垂直制表符

ltrim（）、rtrim（）、trim（）函数分别用于删除字符串 $str 中最左边、最右边和两边的与 $charlist 相同的字符，并返回剩余的字符串。例如：

【例 6.1-4】 去除字符串首尾空格或指定字符。

新建 EX6.1-4.php 文件，输入以下代码：

```
<? php
    $str1 = "hello";
    echo trim( $str1)."<br>";                    //输出"hello"
    $str2 = "aaahelloa";
    echo ltrim( $str2, "a")."<br>";              //输出"helloa"
```

```
    $str3="worldwo";
    echo rtrim( $str3, "wo")."<br>";                    //输出"world"
?>
```

运行结果如图 6-3 所示。

图 6-3　去除字符串首尾空格或指定字符

4. 查找子字符串

PHP 中用于查找、匹配或定位字符串的函数非常多，这里只介绍比较常用的 strstr（）函数和 stristr（）函数，这两者的功能、返回值都一样，只是 stristr（）函数不区分大小写。

strstr（）函数的语法格式如下：

```
string strstr ( string  $str , mixed  $search [ , bool  $before_search ] )
```

说明：strstr（）函数用于查找字符串 $search 在另一字符串 $str 中第一次出现的位置，并返回从该位置到 $str 结尾的所有字符。若未找到，则函数返回 FALSE。PHP 5.3.0 新增可选的 $before_search，默认值为 FALSE，若 $before_search 参数为 TRUE，strstr（）将返回 $search 第一次出现之前的字符串部分。例如：

```
<? php
    echo strstr("hello world","llo");                   //输出"llo world"
    echo strstr("hello world","llo",true);              //输出"he"
    $str="I love PHP";
    $search="PHP";
    if(strstr( $str, $search))
        echo "有 PHP";                                  //输出"有 PHP"
    else
        echo "没有 PHP";                                //不输出
? >
```

5. 字符串与 ASCII 码

在字符串处理中，使用 ord（）函数可以返回字符的 ASCII 码，也可以使用 chr（）函数返回 ASCII 码对应的字符，例如：

```
<? php
    echo ord("a");                                      //输出 97
```

```
    echo chr(98);                    //输出"b"
? >
```

6.1.4　字符串的比较

在字符串操作中，字符串的比较是很常用的，比较主要是对字符串的类型及大小写的比较。字符串的比较可以使用比较运算符“==”“!=”“===”和“!==”来进行，也可以使用比较函数来进行。

使用“!=”和“==”比较的两个对象不一定要类型相等，整型也可以和字符串比较，如“123=="123"”返回 TRUE。而“!==”和“===”比较的对象类型要严格相同才可能返回 TRUE，如“123==="123"”返回 FALSE。

经常使用的字符串比较函数有 strcmp（）、strcasecmp（）、strncmp（）和 strncasecmp（）。语法格式如下：

```
int strcmp(string $str1 , string $str2)
int strcasecmp(string $str1 , string $str2)
int strncmp(string $str1 , string $str2 , int $len)
int strncasecmp(string $str1 , string $str2 , int $len)
```

这 4 个函数都用于比较字符串的大小，如果 $str1 比 $str2 大，则它们都返回大于 0 的整数；如果 $str1 比 $str2 小，则返回小于 0 的整数；如果两者相等，则返回 0。

不同的是，strcmp（）函数用于区分大小写的字符串比较；strcasecmp（）函数用于不区分大小写的比较；strncmp（）函数用于比较字符串的一部分，从字符串的开头开始比较，$len 是要比较的长度；strncasecmp（）函数的作用和 strncmp（）函数一样，只是 strncasecmp（）函数不区分大小写。

6.1.5　字符串的替换

字符串替换是指使用指定的字符串，替换掉原来字符串中的相关字符，以组成新的字符串来满足新的要求。

字符串替换操作中最常用的就是 str_replace（）函数，语法格式如下：

```
<? php
    echo strcmp("aBcd","abde");              //输出-1,比较了"B"和"b","B"<"b"
    echo strcasecmp("abcd","aBde");          //输出-1,比较了"c"和"d","c"<"d"
    echo strncmp("abcd","aBcd",3);           //输出 1,比较了"abc"和"aBc"
    echo strncasecmp("abcdd","aBcde",3);     //输出 0,比较了"abc"和"aBc"
? >
```

说明：str_replace（）函数使用新的字符串 $replace 替换字符串 $subject 中的 $search 字符串。$count 是可选参数，表示要执行的替换操作的次数，$count 是 PHP 5 中添加的。例如：

```
<? php
    $str="I love you";
```

```
    $replace="lucy";
    $end=str_replace("you",$replace,$str);
    echo $end;                          //输出"I love lucy"
?>
```

```
mixed str_replace ( mixed $search , mixed $replace , mixed $subject [ , int & $count] )
```

str_replace（）函数对大小写敏感，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，例如：

```
<? php
    $str="What Is Your Name";
    $array=array("a","o","A","O","e");
    echo str_replace( $array, "",, $str);
    //多对一的替换,输出"Wht Is Yur Nm"
    $arrayl=array("a","b","c");
    $array2=array("d","e","f");
    echo str_replace( $arrayl, $array2, "abcdef");
    //多对多的替换,输出"defdef"
?>
```

说明：使用多对多替换时，第一个数组中的元素被第二个数组中对应的元素替换，如果有一个数组比另一个数组元素数少，那不足的部分会当作空格来处理。

PHP 中还有一个 substr_replace（）函数，其功能是替换字符串的一部分，语法格式如下：

```
mixed substr_replace(mixed $string , string $replacement , int $start [ , int $length ])
```

$start 是开始替换的位置的偏移量，从 0 开始计算，如果为 0 或是一个正值，就是从字符串开始处计算的偏移量；如果是负值，就是从字符串末尾计算的偏移量。

$length 是可选参数，表示要替换的长度，如果不给定则从 $start 位置开始一直到字符串结束；如果 $length 为 0，则替换字符串会插入到原字符串中；如果 $length 是正值，则表示要用替换字符串替换掉的字符串长度；如果 $length 是负值，则表示从字符串末尾开始到 $length 个字符为止停止替换。例如：

```
<? php
    echo substr_replace("abcdefg","OK",3);          //输出"abcOK"
    echo substr_replace("abcdefg","OK",3,3);        //输出"abcOKg"
    echo substr_replace("abcdefg","OK",-2,2);       //输出"abcdeOK"
    echo substr_replace("abcdefg","OK",3,-2);       //输出"abcOKfg"
    echo substr_replace("abcdefg","OK",2,0);        //输出"abOKcdefg"
?>
```

6.1.6 字符串与 HTML

1. 将字符转换为 HTML 实体形式

HTML 代码都是由 HTML 标记组成的，如果要在页面上输出这些标记的实体形式，如

“<table></table>”，就需要使用一些特殊的函数将一些特殊的字符（如“<”“>”等）转换为 HTML 的字符串格式。函数 htmlspecialchars（）可以将字符转化为 HTML 的实体形式，该函数转换的特殊字符及转换后的字符见表 6-3。

表 6-3　可以转化为 HTML 实体形式的特殊字符串

原字符	字符名称	转换后的字符
&	AND 记号	&
“	双引号	"
‘	单引号	'
<	小于号	<
>	大于号	>

htmlspecialchars（）函数的语法格式如下：

```
string htmlspecialchars ( string  $string [ , int  $quote_style [ , string  $charset [ , bool  $double_encode ] ] ] )
```

参数 $string 是要转换的字符串，$quote_style、$charset 和 $double_encode 都是可选参数。$quote_style 指定如何转换单引号和双引号字符，取值可以是 ENT_COMPAT（默认值，只转换双引号）、ENT_NOQUOTES（都不转换）和 ENT_QUOTES（都转换）等。$charset 是字符集，默认为 UTF-8（PHP 5.4.0）。参数 $double_encode 如果为 FALSE，则不转换成 HTML 实体，默认为 TRUE。例如：

```
<? php
    $new = "<a href='test'>test</a>";
    echo htmlspecialchars( $new);
    //页面中输出“<a href='test'>test</a>”
    echo htmlspecialchars( $new,ENT_NOQUOTES);
    //页面中输出“<a href='test'>test</a>”
? >
```

2. 将 HTML 实体形式转换为特殊字符

使用 htmlspecialchars_decode（）函数可以将 HTML 的实体形式转换为 HTML 格式，这和 htmlspecialchars（）函数的作用刚好相反。Html_entity_decode（）函数可以把所有 HTML 实体形式转换为 HTML 格式，和 htmlentities（）函数的作用相反。例如：

```
<? php
    $html = htmlspecialchars_decode( "<a href='test'>test</a>" );
    echo  $html;
? >
```

输出结果在页面上看到“test”超链接。

3. 换行符的转换

在 HTML 文件中使用“\n”，显示 HTML 代码时不能显示换行的效果，这时可以使用 nl2br（）函数，这个函数可以用 HTML 中的“
”标记代替字符串中的换行符

“\n”。例如：

```
<? php
    $str="hello\nworld";
    echo $str;                          //输出"hello world"不会有换行符
    echo nl2br( $str);                  //"hello"后面换行,再输出"world"
? >
```

6.1.7 其他字符串函数

1. 字符串与数组

在 PHP 编程中，经常需要将数组元素转换成字符串，或者将字符串转换成数组元素，使用 explode（）和 implode（）函数可以分别实现上述效果。

（1）字符串转化为数组。使用 explode（）函数可以用指定的字符串分割另一个字符串，并返回一个数组。

语法格式如下：

```
array explode(string $separator , string $string [ , int $limit ])
```

说明：此函数返回由字符串组成的数组，每个元素都是 $string 的一个子串，它们被字符串 $separator 作为边界点分割出来。例如：

```
<? php
    $str = "I love beijing!";                   //定义字符串变量
    $arr = explode("", $str);                   //以空格分割字符串为数组元素
    print_r( $arr);
? >
```

运行结果：Array（[0] => I [1] => love [2] => beijing!）。

如果设置了 $limit 参数，则返回的数组包含最多 $limit 个元素，而最后那个元素将包含 $string 剩余部分。如果 $limit 参数是负数，则返回除了最后的- $limit 个元素外的所有元素。

如果参数 $separator 为空字符串（""），explode（）函数将返回 FALSE。如果 $separator所包含的值在 $string 中找不到，则 explode（）函数将返回包含 $string 单个元素的数组。

（2）数组转化为字符串。使用 implode（）函数可以将数组中的字符串连接成一个字符串，语法格式如下：

```
string implode(string $glue , array $pieces)
```

$pieces 是保存要连接的字符串的数组，$glue 是用于连接字符串的连接符。例如：

```
<? php
    $array=array("hello","how","are","you");
    $str=implode(",", $array);                  //使用逗号作为连接符
    echo $str;                                  //输出"hello,how,are,you"
? >
```

implode（）函数还有一个别名，即 join（）函数。

2. 字符串加密函数

PHP 提供了 crypt（）函数完成加密功能，语法格式如下：

```
string crypt(string $str [, string $salt ])
```

这个函数完成的是单向加密的功能，即字符串一旦被加密就无法转化为原来的形式。函数中 $str 是需要加密的字符串，第二个可选参数 $salt 是一个位字串，它能影响加密的暗码，进一步排除预计算攻击的可能性。

从 PHP 5.3.0 开始，PHP 包含了它自己的 MD5 Crypt 实现，包括标准 DES 算法、扩展的 DES 算法以及 Blowfish 算法。如果系统缺乏相应的实现，那么 PHP 将使用它自己的实现，以便在默认状态下获得更高的安全性。例如：

```
<? php
    $str="字符串";
    $password=crypt( $str);
   echo  $password;
   if(crypt("字符串", $password)= = $password){
       echo "验证成功!";
   }
? >
```

6.2　正则表达式

用户在注册为某网站的会员时，经常需要输入用户名、电话号码等信息，偶尔输错时还会收到不合格验证信息，需要重新输入。对字符串的验证是一种常见的 Web 逻辑应用，是在服务器端脚本中用正则表达式实现的。

PHP 曾同时支持两种风格的正则表达式语法：POSIX 和 Perl。但在实践中，实现相同功能使用 Perl 兼容的正则表达式在效率上明显占优势，自 PHP 5.3.0 起，POSIX 正则表达式扩展被废弃，未来 PHP 6 将不会再支持它，故本节只对 Perl 兼容的正则表达式进行介绍。

6.2.1　正则表达式简介

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符，及这些特定字符的组合，组成一个“规则字符串”，该“规则字符串”描述在查找文字主体时待匹配的一个或多个字符串。可以说，正则表达式是作为一个模板，将某个字符模式与所搜索的字符串进行匹配。

接触过 DOS 的用户可能知道，如果想要匹配当前文件夹下所有的文本文件，可以输入“dir *.txt”命令，按【Enter】键后所有“.txt”文件都会被列出来。此处的“*.txt”即可理解为一个简单的正则表达式。

在 PHP 中，正则表达式基本上有以下 3 个作用：

- 判断给定字符串是否匹配正则表达式。
- 用新文本替换匹配文本。
- 将一个字符串拆分为一组更小的信息块，从字符串中获取其特定部分。

6.2.2 正则表达式的语法规则

正则表达式由一些普通字符和一些元字符（特殊字符）以及模式修正符组成。普通字符包括大小写的字母、数字、标点符号、非打印字符（如换行符、回车符等）以及双引号、单引号等符号，而元字符则是一些具有特殊含义的符号，如“*”“?”等。

在最简单的情况下，一个正则表达式看上去就是一个普通的字符串。例如，正则表达式“testing”中没有包含任何元字符，它可以匹配“testing”和“testing123”等字符串，但是不能匹配“Testing”。

1. 元字符

元字符是指在正则表达式中具有特殊意义的字符。要想真正用好正则表达式，必须正确理解元字符的应用。

常用元字符及对其的简单描述，见表 6-4。

表 6-4 常用元字符表

字 符	描 述	举 例
\	相当于 PHP 中的转义字符，将特殊字符变为普通字符	例如，匹配单个字符，“\.”匹配一个点号。“\-”匹配连字符“-”，“\\”匹配符号“\”
^	匹配字符串的开始位置	例如，“^he”表示以“he”开头的字符串
$	匹配字符串的结束位置	例如，“ok $”表示以“ok”结尾的字符串
*	匹配前面的字符 0 次或多次	例如，“zo*”能匹配“z”以及“zoo”。“*”等价于{0,}
+	匹配前面的字符 1 次或多次（大于 1 次）	例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”。“+”等价于{1,}
?	匹配前面的字符 0 次或 1 次	例如，“do(es)?”可以匹配“do”或“does”中的“do”。“?”等价于{0, 1}
{n}	n 是一个非负整数。匹配前面的字符 n 次	例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个“o”
{n,}	n 是一个非负整数。至少匹配 n 次	例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“foooood”中的所有“o”。“o{1,}”等价于“o+”。“o{0,}”则等价于“o*”
{n, m}	m 和 n 均为非负整数，其中 n≤m。最少匹配 n 次且最多匹配 m 次	例如，“o{1, 3}”将匹配“fooooood”中的前三个“o”。“o{0, 1}”等价于“o?”。请注意在逗号和两个数之间不能有空格
.（点）	匹配除“\r\n”之外的任何单个字符	例如，“s.t”可以匹配“sat”“sit”和“set”，但不匹配“seat”

续表

字　符	描　述	举　例
x | y	匹配 x 或 y	例如，如“z | food”能匹配“z”或“food”，“[z | f] ood”则匹配“zood”或“food”
[xyz]	字符集合。匹配所包含的任意一个字符	例如，“[abc]”可以匹配“plain”中的“a”
[^xyz]	负值字符集合。匹配未包含的任意字符	例如，“[^abc]”可以匹配“plain”中的“plin”
[a-z]	字符范围。匹配指定范围内的任意字符	例如，“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符
[^a-z]	负值字符范围。匹配不在指定范围内的任意字符	例如，“[^a-z]”可以匹配不在“a”到“z”范围内的任意字符
\ b	匹配一个单词边界，也就是指单词和空格间的位置	例如，“er\ b”可以匹配“never”中的“er”，但不能匹配“verb”中的“er”
\ B	匹配非单词边界	例如，“er\ B”能匹配“verb”中的“er”，但不能匹配“never”中的“er”

提示：只有连字符在字符组内部，并且出现在两个字符之间时，才能表示字符的范围；如果出现在字符组的开头，则只能表示连字符本身。

反斜杠指定的预定义字符集，见表 6-5。

表 6-5　反斜杠指定的预定义字符集

预定义字符集	说　明
\ d	匹配一个数字字符，等价于[0-9]
\ D	匹配一个非数字字符，等价于[^0-9]
\ s	匹配任何不可见字符，包括空格、制表符、换行符等，等价于[\ f\ n\ r \ t\ v]
\ S	匹配任何可见字符，等价于[^ \ f\ n\ r\ t\ v]
\ w	匹配包括下画线的任何单词字符，类似但不等价于“[A-Za-z0-9_]”
\ W	匹配任何非单词字符，等价于“[^A-Za-z0-9_]”

2. 模式修正符

模式修正符的作用是规定正则表达式该如何解释和应用，PHP 中的常用模式修正符见表 6-6。

表 6-6　PHP 中的常用模式修正符

修正符	说　明
i	忽略大小写模式
m	多行匹配。仅当表达式中出现“^”“$”中的至少一个元字符且字符串有换行符时，“m”修饰符才起作用。“m”修饰符可以改变“^”为表示每一行的头部
s	改变元字符“.”的含义，使其可以代表所有字符（包括换行符），其他模式不能匹配换行符
x	忽略空白字符

6.2.3 Perl 兼容正则表达式函数

“Perl 兼容正则表达式（Perl Compatible Regular Expression）”简称 PCRE，也有称其为“PCRE 兼容正则表达式”。Perl 是一种编程语言，其字符处理功能非常强大，此处的 PCRE 就是使用了 Perl 的正则函数库。

在 PCRE 中，表达式应被包含在定界符中，一般是斜线“/”。实现 PCRE 风格正则表达式的函数有以下 5 个。

1. preg_grep（）函数

函数语法格式如下：

```
array preg_grep (string  $pattern, array  $input)
```

函数 preg_grep（）返回一个数组，其中包括 $input 数组中与给定的 $pattern 模式相匹配的元素。对于数组 $input 中的每个元素，preg_grep（）只进行一次匹配。

【例 6.2-1】 preg_grep（）函数的应用。

新建 EX6.2-1.php 文件，输入以下代码：

```
<? php
    $subjects = array("Mechanical Engineering", "Medicine", "Social Science", "Agriculture", "
Commercial Science", "Politics");
    //匹配所有仅由一个单词组成的科目名,"i"表示不区分大小写
    $alonewords = preg_grep("/^[a-z] * $/i", $subjects);
    var_dump( $alonewords);
? >
```

运行结果如图 6-4 所示。

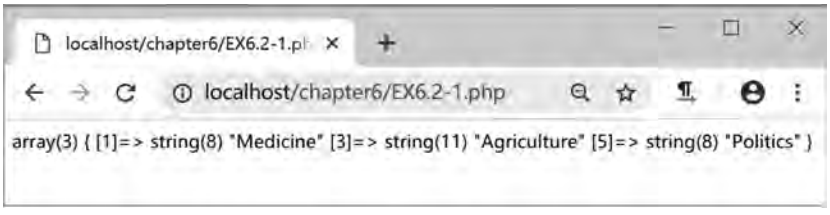


图 6-4　preg_grep（）函数的应用

2. preg_match（）和 preg_match_all（）函数

preg_match（）函数用于匹配字符串。其语法格式如下：

```
int preg_match ( string  $pattern, string  $subject [, array  $matches])
```

该函数在字符串 subject 中搜索与表达式 pattern 相匹配的内容，成功则返回整个模式匹配的次数（可能为零），出错则返回 false。如果定义了数组 matches，则将每次匹配的结果存储到该数组中。

preg_match（）第一次匹配成功后就会停止匹配，如果要实现全部结果的匹配，即搜索到 subject 结尾处，则需使用 preg_match_all（）函数。

【例 6.2-2】使用 preg_match（）和 preg_match_all（）函数匹配字符串。

新建 EX6.2-2.php 文件，输入以下代码：

```
<? php
    $str = "php 功能强大,学习 php 是一件快乐的事。";
    $preg = "/[php]+/";
    $a = preg_match( $preg, $str, $match1);
   echo $a."<br>";
   var_dump( $match1);
    $b = preg_match_all( $preg, $str, $match2);
   echo "<p>". $b."<br>";
   var_dump( $match2);
? >
```

运行结果如图 6-5 所示。

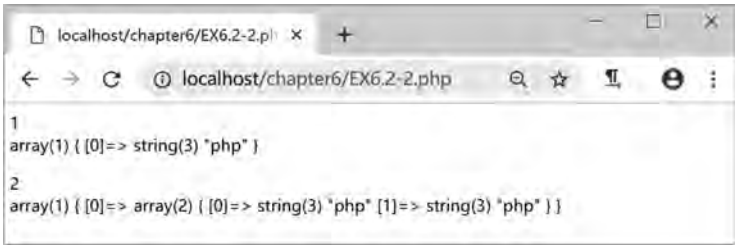


图 6-5　使用 preg_match（）和 preg_match_all（）函数

3. preg_quote（ ）函数

该函数可以对特殊字符进行自动转义，其语法格式如下：

```
string preg_quote( string $str [, string $delimiter] )
```

该函数给字符串 str 中的所有特殊字符前面加上一个反斜线，对其进行自动转义。如果定义了参数 delimiter，则该参数所包含的字符串也将被转义，函数返回转义后的字符串。

【例 6.2-3】对字符串进行自动转义。

新建 EX6.2-3.php 文件，输入以下代码：

```
<? php
    $k1 = " $40 for a g3/400";
    $k2 = preg_quote ( $k1, "/");
   echo "原字符串:". $k1. "<br>";
   echo "转义字符串:". $k2;
? >
```

运行结果如图 6-6 所示。

图 6-6　字符串自动转义

4. preg_replace（）函数

preg_replace（）函数用于正则表达式的搜索和替换，其语法格式如下：

```
mixed preg_replace( mixed $pattern, mixed $replacement, mixed $subject [, int $limit ])
```

该函数在字符串 subject 中匹配表达式 pattern，并将匹配项替换成字符串 replacement。如果定义了参数 limit，则替换 limit 次。上述参数除 limit 外都可以是一个数组。如果 pattern 和 replacement 都是数组，将以其键名在数组中出现的顺序来进行处理，这不一定和索引的数字顺序相同。如果使用索引来标识哪个 pattern 将被哪个 replacement 替换，应该在调用 preg_replace（）之前用 ksort（）函数对数组进行排序。

【例 6.2-4】替换字符串。

新建 EX6.2-4.php 文件，输入以下代码：

```
<? php
    $k1 = "The quick brown fox jumped over the lazy dog.";        //定义变量
    $k2 = preg_replace('/\s/','-', $k1);
   //定义变量,并将替换后的字符串赋给它
   echo "原字符串:". $k1. "<br>";                                  //输出字符串变量 $k1
   echo "替换空格后的字符串:". $k2;                                //输出字符串变量 $k2
? >
```

运行结果如图 6-7 所示。

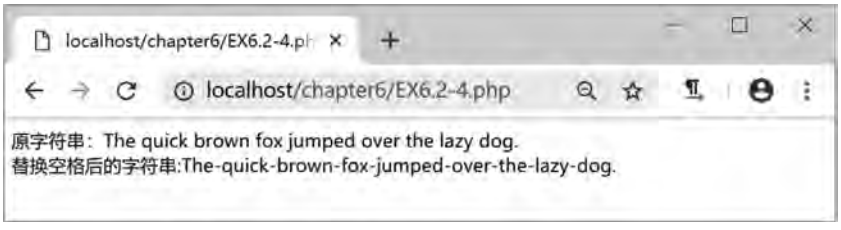


图 6-7　替换字符串

5. preg_split（）函数

preg_split（）函数用于分割字符串，其语法格式如下：

```
array preg_split( string $pattern, string $subject [, int $limit [, int $flags]] )
```

该函数返回一个数组，包含字符串 subject 经正则表达式 pattern 作为边界所分割出的子串。如果定义了参数 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。

【例 6.2-5】 分割字符串。

新建 EX6.2-5.php 文件，输入以下代码：

```
<? php
    $str = "php mysql,apache ajax";                    //定义字符串变量
    $keywords = preg_split("/[\s,]+/", $str);
    //定义变量,并将分割后的字符串赋给它
    print_r( $keywords);                               //输出分割后的字符串组成的数组
? >
```

运行结果如图 6-8 所示。

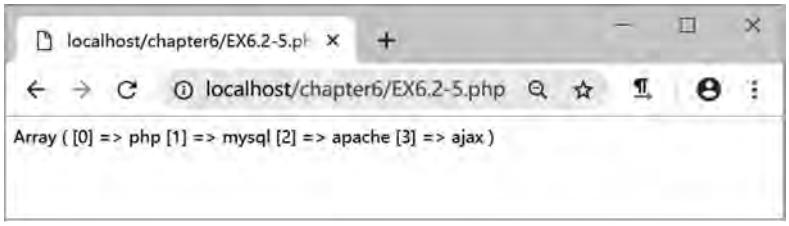


图 6-8　分割字符串

注意：在实际应用中，普通的字符串函数比正则表达式函数运行速度要快得多。在处理简单的字符串时，最好使用字符串函数来完成。例如，使用 explode（ ）函数就可以满足分割字符串的需要，而不必使用正则表达式函数 preg_split（ ），因为这样会影响效率。在处理一些复杂字符串时，若字符串函数不能解决问题，可以选择正则表达式函数来完成。

6.3　综 合 案 例

本节通过两个综合案例来说明 PHP 中字符串和正则表达式的使用。

【例 6.3-1】 新建一个留言簿，留言簿上有 Email 地址和用户的留言，请提取客户的 Email 地址和留言内容，要求 Email 地址中@ 符号前不能有点“.”或逗号“,”。将 Email 地址中@ 符号前的内容作为用户的用户名，并将用户留言中第一人称“我”修改为“本人”。

新建 EX6.3-1.php 文件，输入以下代码：

```
<form name="f1" method="post" action="">
<font face="方正舒体" size=4>您的 Email 地址:</font><br>
<input type="text" name="Email" size=31><br>
<font face="方正舒体" size=4>您的留言:</font><br>
<textarea name="note" rows=10 cols=30></textarea>
```

```
<br><input type="submit" name="bt1" value="提交">
<input type="reset" name="bt2" value="清空">
</form>
<! --以上是留言簿表单-->
<? php
if( isset( $_POST['bt1'] ) )
{
    $Email= $_POST['Email'];                    //接收 Eamil 地址
    $note= $_POST['note'];                      //接收留言
    if(! $Email||! $note)                       //判断是否取得值
        echo "<script>alert('Email 地址和留言请填写完整! ')</script>";
    else
    {
        $array=explode("@", $Email);            //分割 Email 地址
        if(count( $array)! =2)                  //如果有两个@ 符号则报错
        echo "<script>alert('Email 地址格式错误! ')</script>";
    else
        {
            $username= $array[0];               //取得@ 符号前的内容
            $netname= $array[1];                //取得@ 符号后的内容
            //如果 username 中含有"."和","则报错
            if(strstr( $username,".") or strstr( $username,","))
            echo "<script>alert('Email 地址格式错误! ')</script>";
            else
            {
                //将留言中的"我"用"本人"替代
                $newnote=str_replace("我","本人", $note);
                echo "<font face='黑体' size=4>";
                echo "用户". $username ."您好!  ";
                echo "您是". $netname."网友! <br>";
                echo "<br>您的留言是:<br>   ". $newnote."<br>";
                echo "</font>";
            }
        }
    }
}? >
```

运行程序，在 Email 栏输入“ANNIE@ 163. com”，在留言栏输入“我觉得 PHP 是个很有趣的东西，要好好学习，以后从事 PHP 开发工作!”，单击【提交】按钮，运行结果如图 6-9 所示。

【例 6. 3-2】 使用正则表达式验证用户输入的表单内容是否满足格式要求。

图 6-9　留言簿内容处理

新建 EX6.3-2Hpage.php 文件，输入以下代码：

```
<! DOCTYPE html>
<html>
<head>
    <title>注册页面</title>
    <style type="text/css">
    <! --
        .STYLE1{font-size: 14px; color:red;}
    -->
    div{
        text-align:center;
        font-size:24px;
        color:#0000FF;
    }
    table{
        margin:0 auto;
    }
    </style>
</head>
<body>
<form name="fr1" method="post" action="EX6.3-2Ppage.php">
    <div>新用户注册</div>
    <table border="1">
        <tr>
```

```
            <td>用户名:</td>
            <td><input type="text" name="ID"></td>
            <td class="STYLE1">* 不超过 10 个字符(数字,字母和下划线)</td>
        </tr>
        <tr>
            <td>密码:</td>
            <td><input type="password" name="PWD" size="21"></td>
            <td class="STYLE1">* 4~14 个数字</td>
        </tr>
        <tr>
            <td>手机号码:</td>
            <td><input type="tel" name="PHONE"></td>
            <td class="STYLE1">* 11 位数字,第一位为 1</td>
        </tr>
        <tr>
            <td>邮箱:</td>
            <td><input type="email" name="EMAIL"></td>
            <td class="STYLE1">* 有效的邮件地址</td>
        </tr>
        <tr>
            <td colspan="3" align="center">
                <input type="submit" name="GO" value="注册">  
                <input type="reset" name="NO" value="取消">
            </td>
        </tr>
    </table>
</form>
</body>
</html>
```

新建 EX6.3-2Ppage.php 文件，输入以下代码：

```
<? php
    include "EX6.3-2Hpage.php";                    //包含文件 EX6.3-2Hpage.php
     $id= $_POST['ID'];
     $pwd= $_POST['PWD'];
     $phone= $_POST['PHONE'];
     $Email= $_POST['EMAIL'];
     $checkid=preg_match('/^\w{1,10} $/', $id);
    //检查字符串是否在 10 个字符以内
     $checkpwd=preg_match('/^\d{4,14} $/', $pwd);

    //检查是否在 4~14 个数字之间
```

```
        $checkphone=preg_match('/^1\d{10}$/', $phone);
        //检查是否是以 1 开头的 11 位数字
        //检查 Email 地址的合法性
        $checkEmail=preg_match('/^[a-zA-Z0-9_\-]+@[a-zA-Z0-9\-]+。[a-zA-Z0-9\-。]
+ $/', $Email);
        //检查 Email 地址的合法性
        if( $checkid&& $checkpwd&& $checkphone&& $checkEmail)
        //如果都为 1,则注册成功
            echo "注册成功!";
        else
            echo "注册失败,格式不对";
    ? >
```

EX6.3-2Hpage.php 文件运行结果如图 6-10 所示，用户需要输入用户名、密码、手机号和邮箱地址，需要遵守的规则在表单中已经列出。输入完成后单击【注册】按钮，如果格式正确则提示“注册成功”，不正确则提示“注册失败，格式不对”。

图 6-10　用户注册页面

本章小结

本章主要介绍了字符串与正则表达式的相关知识。在学完本章内容后，读者应重点掌握以下知识。

（1）字符串是由一对单引号（’）或一对双引号（”）包含的零个或多个字符组成的集合。可以使用 3 种形式来表示：单引号（’）、双引号（”）和定界符（<<<）。

（2）几乎所有 PHP 脚本的输入/输出都要用到字符串。可以直接在函数中使用字符串，也可以将其存储在变量中。很多字符串操作都可以通过 PHP 内置函数来完成。

（3）正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符及这些特定字符的组合，组成一个“规则字符串”，该“规则字符串”描述在查找文字主体时待匹配的一个或多个字符串。

（4）正则表达式由一些普通字符和一些元字符（特殊字符）以及模式修正符组成。普通字符包括大小写的字母、数字、标点符号、非打印字符以及双引号、单引号等符号，而元字符则是一些具有特殊含义的符号，如“*”“?”等。

习　题

一、填空题

1. 字符串是由_____、_____、_____等组成的一串字符。

2. 实际的网页制作中，常用_____函数获取字符串长度。

3. PHP 提供了 trim（）、rtrim（）和 ltrim（）函数分别用于去除_____空格、串尾部空格和_____空格。

4. 在 PHP 中，如果要截取某个字符串中指定长度的字符，可以使用 _____函数来实现。

5. _____的作用是规定正则表达式该如何解释和应用。

二、简答题

1. 简述字符串的类型。

2. 简述正则表达式的作用和语法规则。

第 7 章

文件与目录访问操作

【本章内容简介】

文件存储在硬盘上的目录中，即使在计算机关机后，文件中的内容也仍然可以保留下来，这是一种永久存储机制。目录是一种特殊的文件，就像在 Windows 操作系统中称为文件夹，用来对其他文件进行分类，因此，目录采用层级结构，即目录中还有其他目录或文件。

文件可以包含任何类型的数据，此外还可以包含文件自身的一些信息，如文件的创建者、创建时间等。为了方便用户操作文件系统，PHP 提供了专门用来操作文件的功能函数，通过这些函数可以获取文件的信息，也可以打开、读取和写入文件。除此之外，PHP 还提供了专门用于操作目录的功能函数，通过这些函数可以对目录执行增加、删除、修改、查询等操作，或者对目录进行遍历，读取目录中的文件信息等。

【本章学习要求】

通过本章的学习，读者应能熟练掌握文件和目录操作的功能函数，并做到以下几点：

（1）理解文件和目录的概念。

（2）掌握打开、关闭文件的方法。

（3）了解文件权限的基本概念，能使用 PHP 程序对文件权限进行操作。

（4）了解目录相关的全部操作，包含读取目录的内容、创建目录、删除目录等。

（5）能够熟练运用 PHP 提供的文件和目录相关函数，对文件和目录执行常用的操作。

任何类型的变量，都是在程序运行期间才将数据加载到内存中的，程序结束时数据就会随之消失了，并不能持久保存。如果需要将数据长久保存起来，以便后期程序再次运行时还可以使用，就要使用文件。

存储数据的基本方法通常有两种：将数据保存到普通文件中和保存到数据库中。本章将重点介绍在 PHP 中如何操作目录和文件。

7.1 文件系统概述

在任何计算机设备中，各种数据、信息、程序主要以文件的形式储存。一个文件通常对应着磁盘上的一个或多个存储单元，利用目录可以有效地对文件进行区分和管理。负责管理和存储文件信息的软件机构称为文件管理系统，简称文件系统，从系统角度来看，文件系统是对文件存储器空间进行组织和分配，负责文件的存储并对存入的文件进行保护和检索的系统，具体地说，它负责为用户建立文件，存入/读出、修改、转储文件，控制文件的存取，当用户不再使用时删除文件等。通过 PHP 中内置的文件处理函数可以完成对服务器端文件系统的操作，但 PHP 对文件系统的操作是基于 UNIX 系统模型的，因此，其中的很多函数类似于 UNIX Shell 命令，在 Windows 中并没有提供 UNIX 的文件系统特性。所以有一些 PHP 文件处理函数不能在 Windows 服务器中使用，但绝大多数函数的功能是兼容的。另外在 PHP 中，对文件读写等操作与 C 语言中的文件读写操作是相同的，如果读者曾经编写过 C 语言或者是 UNIX Shell 脚本程序，就会非常熟悉这些操作。

7.1.1 文件类型

PHP 是以 UNIX 的文件系统为模型的，因此在 Windows 系统中我们只能获得“file”“dir”或者“unknown”三种文件类型。而在 UNIX 系统中，我们可以获得“block”“char”“dir”“fifo”“file”“link”和“unknown”7 种类型，各种文件类型的详细说明见表 7-1。

表 7-1 UNIX 系统中 7 种文件类型说明

文件类型	描 述
block	块设备文件，如某个磁盘分区、光驱等
char	字符设备是指在 I/O 传输过程中以字符为单位进行传输的设备，如键盘、打印机等
dir	目录类型。目录也是文件的一种
fifo	命名管道，常用于将信息从一个进程传递到另一个进程
file	普通文件类型，如文本文件或可执行文件等
link	符号链接，是指向文件指针的指针，类似 Windows 中的快捷方式
unknown	未知类型

在 PHP 中可以使用 filetype（）函数获取文件的类型，该函数接受一个文件名作为参数，如果文件不存在将返回 FALSE。例如，假设当前目录中存在 test. txt 文件：

```
<? php
echo filetype("test.txt");//输出:file
? >
```

又如，假设当前目录中存在 images：

```
<? php
echo filetype("images");//输出:dir
? >
```

提示：本函数的结果会被缓存。请使用 clearstatcache（）来清除缓存。

对一个已知的文件，还可以使用 is_file（）函数判断给定的文件名是否为一个正常的文件，使用 is_dir（）函数判断给定的文件名是否是一个目录，使用 is_link（）函数判断给定的文件名是否为一个符号链接等。

7.1.2　文件属性

在进行编程时，需要使用到文件的一些常见属性，如文件的大小、文件的类型、文件的修改时间、文件的访问时间和文件的权限等。PHP 中提供了非常全面的用来获取这些属性的内置函数，见表 7-2。

表 7-2　PHP 获取文件属性的函数

函数名	描　述
file_existes（）	判断文件是否存在
filesize（）	获取文件的大小
is_readable（）	判断文件是否可读
is_writeable（）	判断文件是否可写
is_executable（）	判断文件是否可执行
filectime（）	获取文件创建的时间
fileatime（）	获取文件的最后访问时间
filemtime（）	获取文件的最后修改时间
stat（）	获取给定文件的大部分属性值，返回关于给定文件有用信息的数组
filetype（）	只读，返回文件的类型，如文件或文件夹
realpath（）	返回文件的物理路径
pathinfo（）	以数组形式返回文件的路径和文件名信息
dirname（）	返回文件相对于当前文件的路径信息
basename（）	返回文件的文件名信息

【例 7.1-1】

```
<? php
    $file='tang.txt';
    echo "<br>文件名:".basename( $file); //echo "<br>文件名:".__FILE__;
    $patharr=pathinfo( $file);
    echo "<br>文件扩展名:". $patharr['extension'];
    echo "<br>文件属性:". filetype ( $file);
    echo "<br>路径:". realpath( $file);
    echo "<br>大小:". filesize ( $file);
```

```
    echo "<br>创建日期:". date('Y-m-d H:i:s',filectime( $file)) ;
? >
```

运行结果如下：

```
文件名：tang.txt
文件扩展名：txt
文件属性：file
路径：D:\phpStudy\WWW\NewBook\tang.txt
大小：18
创建日期：2018-08-17 09:52:15
```

7.2 目录的基本操作

文件目录是指存储在磁盘中的文件的索引，也可以将其视为一个文件夹。在这个文件夹中可以存放其他文件或文件夹。最顶层的目录是磁盘的根目录，用“/”或“\\”表示。“./”代表当前工作目录，“../”表示 Apache 的文件根目录，如果文件夹前不指定位置，则默认在当前工作目录中查找。

使用 PHP 脚本可以方便地对服务器中目录进行操作，包含创建目录、遍历目录、复制目录、删除目录等操作。可以借助 PHP 的系统函数完成一部分，但还有一些功能需要自己定义函数操作。常用的目录操作函数见表 7-3。

表 7-3 常用目录操作函数

函数名	说 明	示 例
mkdir（pathname）	新建一个指定的目录	mkdir（'temp'）
rmdir（dirname）	删除目录，该目录必须为空	rmdir（'data'）
getcwd（void）	取得当前文件所在的目录	echo getcwd（）;
chdir（dirname）	改变当前目录	chdir（'../'）;
opendir（path）	打开目录，返回目录的指针	$dirh=opendir（'temp'）
closedir（）	关闭目录，参数为目录指针	closedir（$dirh）;
readdir（）	遍历目录	$file=readdir（$dirh））
scandir（path，sort）	以数组形式遍历目录，sort 参数可设置升序或降序排列	$arr = scandir（'D:\App'，1）; print_r（$arr）;
rewinddir（）	将目录指针重置到目录开头处，即倒回目录开头	rewinddir（$dirh）

7.2.1 创建和删除目录

使用 mkdir（）函数可以根据提供的目录名或目录的全路径创建新的目录，如果创建成功则返回 TRUE，否则返回 FALSE。例如：

```
<? php
if( mkdir( "./path" ,0700) )      //在当前目录中创建 path 目录
    echo "创建成功";
? >
```

使用 rmdir（）函数可以删除一个空目录，但是必须具有相应的权限。如果是非空的目录就需要进入到目录中，可以使用 unlink（）函数将目录中的每个文件都删除掉，再回来将这个空目录删除。如果目录中还存在子目录，而且子目录也非空，就要使用递归的方法了。

【例 7.2-1】

```
<? php
    function delDir( $directory) { //自定义函数递归的函数整个目录
        if( file_exists( $directory) ) {
            if( $dir_handle = @ opendir( $directory) ) {
                while( $filename = readdir( $dir_handle) ) {
                    if( $filename! ='.' && $filename! ='..') {//一定要排除两个特殊的目录
                        $subFile = $directory."/". $filename;
                        if( is_dir( $subFile) ) {                    //如果是目录条件则成立
                            delDir( $subFile);                       //递归调用自己删除子目录
                        }
                        if( is_file( $subFile) ) {                   //如果是文件条件则成立
                            unlink( $subFile);                       //直接删除这个文件
                        }
                    }
                }
                closedir( $dir_handle);//关闭目录资源
                rmdir( $directory);//删除空目录
            }
        }
    }
    delDir( "test" );//调用 delDir( )函数
? >
```

注意：在当前目录下创建 test 目录，再创建其他文件或者文件夹进行测试。另外，处理递归删除非空目录外，我们也可以使用操作系统命令“rm -rf”删除非空的目录，但也要从安全和跨平台方面考虑尽量不要去使用。

7.2.2　获取和更改当前工作目录

当前工作目录是指正在运行的文件所处的目录。使用 getcwd（）函数或者 dirname（__FILE__）可以取得当前的工作目录，成功则返回当前的工作目录，失败则返回 FALSE。

```
<? php
echo getcwd( ) . "<br/>"; //输出' D:\phpStudy\WWW\NewBook'
echo dirname( __ FILE __ ); //输出' D:\phpStudy\WWW\NewBook'
? >
```

使用 chdir（）函数可以设置当前的工作目录，该函数的参数是新的当前目录，例如：

```
<? php
echo getcwd( )."<br>";              //当前工作目录为' D:\phpStudy\WWW\NewBook'
@ mkdir( "../test" );               //在 Apache 文件根目录中建立 test 目录
@ chdir( '../ test ' );             //设置 test 目录为当前工作目录
echo getcwd( );                     //输出' D:\phpStudy\WWW\NewBook \ test '
? >
```

7.2.3 遍历目录

在进行 PHP 编程时，需要对服务器某个目录下面的文件进行浏览，通常称为遍历目录。取得一个目录下的文件和子目录，就需要用到 opendir（）函数、readdir（）函数、closedir（）函数和 rewinddir（）函数。

（1）函数 opendir（）用于打开指定目录，接受一个目录的路径及目录名作为参数，函数返回值为指向该目录的指针（资源类型）。如果该目录不存在或者没有访问权限，则返回 FALSE。

（2）函数 readdir（）用于读取已经打开的目录，接受已经用 opendir（）函数打开的可操作目录指针作为参数，函数返回当前目录指针位置的一个文件名，并将目录指针向后移动一位。当指针位于目录的结尾时，因为没有文件存在而返回 FALSE。

（3）函数 closedir（）关闭指定目录，接受已经用 opendir（）函数打开的可操作目录指针作为参数。函数无返回值，运行后将关闭打开的目录。

（4）函数 rewinddir（）用于将目录指针重新指向目录开头，以便重新读取目录中的内容。接受已经用 opendir（）函数打开的可操作目录句柄作为参数，将目录指针重置目录到开始处，即倒回目录的开头。

下面用一个实例来说明以上几个函数的使用方法。注意，在使用该例前请确保同一目录下有 studyPHP2018 文件夹。

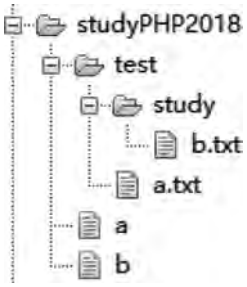
【例 7.2-2】

```
<? php
function readFileFromDir( $dir)
{
    if( ! is_dir( $dir) )
    return false;
     $handle = opendir( $dir);                  //打开目录
    while( ( $file = readdir( $handle) ) ! = = false)
    {
```

```
            if( $file=='.' || $file=='..' )      //判断是否为.或..,默认都会有
            {
                continue;
            }
            $file= $dir.DIRECTORY_SEPARATOR. $file;
            if(is_file( $file))                  //是文件就输出
            {
                print  $file.'<br/>';
            }
            elseif(is_dir( $file))               //如果为目录
            {

                readFileFromDir( $file);         //递归查询
            }
        }
        @closedir( $dir);                        //关闭目录
    }
    $dir='studyPHP2018';
    readFileFromDir( $dir);
?>
```

当前工作目录下有以下文件：



运行结果为：

```
studyPHP2018\a
studyPHP2018\b
studyPHP2018\test\a.txt
studyPHP2018\test\study\b.txt
```

注意：由于 PHP 是弱类型语言，所以将整型值 0 和布尔值 FALSE 视为等价，如果使用比较运算符“==”或“!=”，当遇到目录中有一个文件的文件名为“0”时，则遍历目录的循环将停止。所以在设置判断条件时要使用“===”和“!==”运算符进行强类型检查。

7.3 文件的基本操作

目录可以进行打开、读取、关闭和删除等操作，文件的操作与对目录的操作有类似之处。如需要对文件进行操作前，要求对文件操作的方法如打开、读取、写入和关闭等有一定的掌握。

7.3.1 文件的复制、移动和删除

PHP 提供了大量的文件操作函数，可以对服务器端的文件进行复制、移动、删除、截取和重命名等操作。虽然 PHP 与外部资源接触最多的是数据库，但也有很多情况会应用到普通文件或 XML 文件等。例如文件系统、网页静态化和在没有数据库的环境中持久存储数据等。对文件的操作最常见的就是读（将文件中的数据输入到程序中）和写（将数据保存到文件中），以及一些其他的相关处理，这些操作都可以通过 PHP 提供的众多与文件有关的标准函数完成。文件基本操作函数见表 7-4。

表 7-4 常用的文件操作函数

函 数	语法结构	作 用
copy（）	copy（源文件，目的文件）	复制文件
unlink（）	unlink（目标文件）	删除文件
rename（）	rename（旧文件名，新文件名）	重命名文件或目录，或移动文件
ftruncate（）	ftruncate（目标文件资源，截取长度）	将文件截断到指定长度
file_exists（）	file_exists（目标文件名）	判断文件或文件夹是否存在
is_file（）	is_file（文件名）	判断指定的路径存在且为文件

注意：rename（）函数既可以重命名文件，也可以移动文件，如果旧文件名和新文件名的路径不同，就实现了移动该文件。移动文件还可以使用 move_uploaded_file（）函数。

【例 7.3-1】

```
<? php
if( copy( ' test.txt' ,' ./studyPHP2018/bat.txt' ) )//复制文件
    echo ' 文件复制成功' ;
else echo ' 文件复制失败,源文件可能不存在' ;
    //删除文件示例
//unlink( ' ./test.txt' ) ;
    //移动文件示例
if( file_exists( ' ./studyPHP2018/bat.txt' ) ) //判断源文件是否存在
    {
    if( rename( ' ./studyPHP2018/bat.txt' ,' tang.txt' ) ) //移动到当前路径并重命名为 tang.txt
        echo ' 文件移动并重命名成功' ;
```

```
        else echo '文件移动失败';
        }
? >
```

提示：

（1）复制、移动文件操作都不能自动创建文件夹，因此应保证当前目录下 studyPHP2018 文件夹存在，才能运行该程序。

（2）如果执行删除文件失败，提示 Permission denied，一般是因为网站访问用户没用删除权限，只要在删除文件所在目录上单击右键，在“属性”面板的“安全”选项卡中添加“everyone”权限即可。

7.3.2　获取文件属性函数

在网站开发时，有时需要获取服务器上文件的一些常见属性，如文件的大小、文件类型、文件的修改时间等。PHP 提供了很多获取这些属性的内置函数，见表 7-5。

表 7-5　PHP 获取文件属性函数

函数	说　明	示　例
filesize（）	只读，返回文件的大小	$fsize=filesize（'tang. txt'）
filetype（）	只读，返回文件的类型，如文件或文件夹	filetype（'tang. txt'），返回 file
filectime（）	返回文件创建时间的时间戳	date（'Y-m-d H：i：s'，filectime（'test. php'）））
filemtime（）	只读，返回文件的修改时间	
fileatime（）	只读，返回文件的访问时间	
realpath（）	返回文件的物理路径	realpath（'test. php'）
pathinfo（）	以数组形式返回文件的路径和文件名信息	print_ r（pathinfo（'test. php'）））
dirname（）	返回文件相对于当前文件的路径信息	dirname（' test. php'），返回“.”
basename（）	返回文件的文件名信息	basename（' test. php'）
stat（）	以数组的形式返回文件的大部分属性值	print_ r（stat（' test. php' ）））

说明：

（1）如果要返回当前文件的文件名，除了可使用 basename（‘当前文件名’）名，更简单的方法是使用 PHP 的系统常量“__FILE__”，如 echo“文件名：”．__FILE__；。

（2）对于 Windows 系统，filetype（）返回的文件类型只可能是 file（文件）、dir（目录）、unknown（未知）三种。而在 UNIX 系统中，可以获得 block、char、dir、fifo、file、link 和 unknown 7 种文件类型。这是因为 PHP 是以 UNIX 的文件系统为模型的。

（3）dirname 并不会判断返回的文件的路径信息是否存在，如果要判断路径是否存在，应使用 is_file（）函数。

（4）函数 pathinfo（）返回一个关联数组，其中包括文件或文件夹的目录名、文件名、扩展名、基本名 4 部分，分别通过数组键名 dirname、basename、extension 和 filename

来引用。

现在通过一个示例来获取并显示文件 text. txt 的各种属性。

【例 7. 3-2】

```
<? php
    $file = ' test.txt' ;
    echo "<br>文件名:".basename( $file);
    //echo "<br>文件名:".__ FILE __;
    $patharr=pathinfo( $file);
    echo "<br>文件扩展名:". $patharr[' extension' ];
    echo "<br>文件属性:". filetype ( $file);
    echo "<br>路径:". realpath( $file);
    echo "<br>大小:". filesize ( $file);
    echo "<br>创建日期:". @ date(' Y-m-d H:i:s' ,filectime( $file)) ;
? >
```

其运行结果如图 7-1 所示。

文件名：test.txt
文件扩展名：txt
文件属性：file
路径：D:\phpStudy\WWW\NewBook\test.txt
大小：701
创建日期：2018-08-27 02:30:10

图 7-1　获取文件属性

7. 3. 3　文件操作的一般方法

对目录可以进行打开、读取、关闭、删除等操作，对文件的操作与对目录的操作有类似之处，操作文件的一般方法有打开、读取、写入、关闭等。如果要将数据写入一个文件，一般先要打开该文件，如果文件不存在则先创建它，然后将数据写入文件，最后还需要关闭这个文件。如果要读取一个文件中的数据，同样需要先打开该文件，如果文件不存在则自动退出，如果文件存在则读取该文件的数据，读完数据后关闭文件。

无论如何，如果想要对文件操作，都要先打开文件。使用完文件后需要关闭文件。PHP 对文件操作的一般流程是：①打开文件。②读取或者写入文件。③关闭文件。这些操作都是通过相应的文件访问函数实现的。

1. 打开文件

fopen（）函数用来打开文件。其语法格式为：

```
fopen(string  $filename, string  $mode)
```

其中，参数 $filename 为要打开的文件名（文件路径或者 URL 网址）。如果 $filename 的值是一个由目录和文件名组成的字符串，则 PHP 认为指定的是一个本地文件，将尝试在该文件上打开一个流。如果文件存在，函数将返回一个句柄；如果文件不存在或没有该

文件的访问权限，则返回 FALSE。

如果 $filename 是“scheme：//...”的格式，则被当作一个 URL，PHP 将搜索协议处理器（也被称为封装协议）来处理此模式；如果文件名是以“http：//”开始，则 fopen（）函数将建立一个到指定服务器的 HTTP 连接，并返回一个指向 HTTP 响应的指针；如果文件名是以“ftp：//”开始，fopen（）函数将建立一个连接到指定服务器的被动模式，并返回一个文件开始的指针。如果访问的文件不存在或没有访问权限，函数返回 FALSE。

注意：访问本地文件时，在 UNIX 环境下，目录中的间隔符为正斜线“/”。在 Windows 环境下可以是正斜线“/”或双反斜线“\\”。另外，要访问 URL 形式的文件时，首先要确定 PHP 配置文件中的 allow_url_fopen 选项处于打开状态，如果处于关闭状态，PHP 将发出一个警告，而 fopen（）函数则调用失败。

参数 mode 用来指定以何种模式打开，其可选值及其说明见表 7-6。

表 7-6　参数 mode 的访问文件模式

参数值	说　明
r	只读方式打开文件，从文件头开始读
r+	读写方式打开文件，从文件头开始读写
w	写入方式打开文件，将文件指针指向文件头。如果文件已经存在则删除已有内容，如果文件不存在则尝试创建它
w+	读写方式打开文件，将文件指针指向文件头。如果文件已经存在则删除已有内容，如果文件不存在则尝试创建它
a	写入方式打开文件，将文件指针指向文件末尾，如果文件已有内容将从文件末尾开始写。如果文件不存在则尝试创建它
a+	读写方式打开文件，将文件指针指向文件末尾。如果文件已有内容将从文件末尾开始读写。如果文件不存在则尝试创建它
x	以只写方式创建并打开文件，并将文件指针指向文件头。如果指定文件存在，就会打开失败
x+	以读写方式创建并打开文件，并将文件指针指向文件头。如果指定文件存在，就会打开失败
b	以二进制模式打开，可与 r、w、a 合用

如果 fopen（）函数成功地打开了一个文件，该函数就会返回一个指向这个文件的文件指针（资源类型）。对该文件进行读、写等操作，都需要使用这个指针来访问文件。fopen（）函数举例说明如下。

【例 7.3-3】

```
<? PHP
 $file = fopen ("D:\phpStudy\WWW\NewBook\tang.txt","r");
 $file = fopen ("http://www.abc_PHP.cn/","r");//以只读方式打开网站的首页文件
//以只读方式打开 ftp 目录下的 test.txt 文件
 $file = fopen ("ftp://user:password@ ftp.cn/test.txt","w");
 $file = fopen ("./studyPHP2018/test.txt", "r");
    //以二进制写入方式打开 UNIX 系统目录下的 file.gif 文件
```

```
$file = fopen ("./studyPHP2018/test.txt", "wb");
? >
```

提示：

（1）当以 HTTP 协议的形式打开文件时，只能采取只读的模式，否则会打开失败。

（2）当以 FTP 协议的形式打开文件时，只能采取只读或只写的模式，而不能是读写模式。

（3）如果 $filename 参数中省略了文件路径，则会在当前 PHP 文件所在目录下寻找文件。

2. 关闭文件

文件处理完毕后，需要使用 fclose（）函数关闭文件，语法格式如下：

```
bool fclose( resource $handle)
```

参数 $handle 为要打开的文件指针，文件指针必须有效，如果关闭成功则返回 TRUE，否则返回 FALSE。

【例 7.3-4】

```
<? php
 $handle=fopen("../good/1.php","w");                    //以只写方式打开文件
if(fclose( $handle))                                    //判断是否成功关闭文件
     echo "关闭文件成功";
else
     echo "关闭失败";
? >
```

7.3.4 文件的读取

PHP 提供了多个从文件中读取内容的函数，这些函数的功能见表 7-7，可以根据它们的功能特性在程序中选择使用。

表 7-7 读取文件内容的函数

函 数	说 明
fread（）	读取整个文件或文件中指定长度的字符串，可用于二进制文件读取
fgets（）	读取文件中的一行字符
fgetss（）	读取文件中的一行字符，并去掉所有 HTML 和 PHP 标记
fgetc（）	读取文件中的一个字符
file_get_contents（）	将文件读入字符串
file（）	把文件读入一个数组中
readfile（）	读取一个文件，并输出到输出缓冲

1. fread（）函数

fread（）函数可以用于读取文件的内容，语法格式如下：

```
string fread(int $handle, int $length)
```

说明：参数 $handle 是已经打开的文件指针，$length 是指定读取的最大字节数，$length的最大取值为 8192。如果要读取整个文件，可以通过函数 filezise（）来实现。如果读完 $length 个字节数之前遇到文件结尾标志（EOF），则返回所读取的字符，并停止读取操作。如果读取成功则返回所读取的字符串，如果出错则返回 FALSE。

【例 7.3-5】

```
<? php
 $handle=fopen("test.txt ", "r");                    //打开一个本地文件
 $content="";                                        //将字符串 $content 初始化为空
while(! feof( $handle))                              //判断是否到文件末尾
{
      $data=fread( $handle,8192);                    //读取文件内容
      $content.= $data;                              //将读取到的数据赋给字符串
}
echo  $content;                                      //输出内容
fclose( $handle);                                    //关闭文件
? >
```

说明：上述代码中的 feof（）函数用于判断是否到达文件末尾，feof（）函数只有一个参数，就是文件的句柄，如果文件指针到达文件末尾，则返回 TRUE，否则返回 FALSE。

注意：在读取文件后显示文件内容时，文本中可能含有无法直接显示的字符，如 HTML 标记。这时需要使用 htmlspecialchars（）函数将 HTML 标记转化为实体才能显示文件中的字符。

【例 7.3-6】

```
<html>
<body>
     <h2 align="center">读取文本内容并对 HTML 标记进行转换</h2>
<? php
 $file=fopen("test.txt","r");                        //以只读方式打开 test.txt
 $str=fread( $file,filesize("test.txt"));            //读取文件的全部内容
echo nl2br( $str);                                   //将内容中的回车转<br>再输出
fclose( $file);                                      //关闭文件
? >
</body>
</html>
```

说明：

（1）上面的代码用于读取当前目录下的 test. txt 文件的所有内容。因此，必须保证 test. txt 文件已经存在，否则会出现错误信息。

（2）程序中用 filesize（）获取文件大小，如果需要读取文件部分内容，可以自定义长度。

（3）使用 nl2br（）函数将换行符转换为
标记。

2. fgets（）函数

fgets（）函数用来读取文本文件中的一行，格式如下：

```
string fgets(resource handle[, int length])
```

说明：$handle 是已经打开的文件句柄，可选参数 $length 指定了返回的最大字节数，考虑到行结束符，最多可以返回 length -1 个字节的字符串。如果没有指定 $length，默认为 1024 个字节。

【例 7.3-7】

```
<? PHP
 $file=fopen("test.txt","r");
while(! feof( $file)){
      $str=fgets( $file);        //读取文件中的一行,读取完后指针会指向下一行
     echo  $str."<br>";}
fclose( $file);                  //关闭文件
? >
```

说明：feof（）函数可以判断是否到达文件末尾（最后一个字符之后），如果已达到，则返回 TRUE，否则返回 FALSE。因此，程序会一直读取到文件末尾。

3. fgetss（）函数

fgetss（）函数的作用与 fgets（）函数基本相同，也是从文件指针处读取一行数据，不过 fgetss（）函数会尝试从读取的文本中去掉任何 HTML 和 PHP 标记。语法格式如下：

```
string fgetss(resource  $handle [, int  $length [, string  $allowable_tags ]])
```

例如，假设 html 目录下的 1.txt 第一行内容为“php”，显示内容时不显示“php”的加黑效果，可以使用以下代码：

【例 7.3-8】

```
<? php
 $handle=fopen("test.txt ","rb");
 $one=fgetss( $handle);                    //获取第一行数据,并去除 HTML 标记
echo  $one;                                //输出第一行内容
fclose( $handle);
? >
```

4. fgetc（）函数

fgetc（）函数可以从文件指针处读取一个字符，语法格式为：

```
string fgetc(resource  $handle)
```

该函数返回 $handle 指针指向的文件中的一个字符，遇到 EOF 则返回 FALSE。

【例 7.3-9】

```
<? php
 $handle=fopen("test.txt ", "r");
while(! feof( $handle))                    //判断是否到文件尾
{
      $char=fgetc( $handle);               //获取当前一个字符
```

```
    echo ( $char == "\n" ? '<br>' : $char);
}
? >
```

说明：这段代码一次从文件中读出一个字符并保存到 $char 中输出，直到文件结束。如果文件文本中有换行符“\n”，则显示时使用“
”标记来代替，这里使用三元运算符来实现。

5. file_get_contents（）函数

file_get_contents（）函数无须经过打开文件及关闭文件操作就可以将整个或部分文件内容读取到一个字符串中，功能与依次调用 fopen（）、fread（）及 fclose（）函数的功能一样。语法格式如下：

```
string file_get_contents(string $filename [, int $offset [, int $maxlen ]])
```

说明：$filename 是要读取的文件名，可选参数 $offset 可以指定从文件头开始的偏移量，函数可以返回从 $offset 所指定的位置开始长度为 $maxlen 的内容。如果失败，函数将返回 FALSE。例如：

```
<? php
$filestring=file_get_contents("test.txt");
echo nl2br( $filestring);
? >
```

6. file（）函数

file（）函数用于将整个文件读取到一个数组中，语法格式如下：

```
array file(string $filename [, int $use_include_path [, resource $context ]])
```

说明：本函数的作用是将文件作为一个数组返回，数组中的每个元素都是文件中相应的一行，包括换行符在内，如果失败则返回 FALSE。该函数可以读取二进制文件。

【例 7.3-10】

```
<? php
$line=file("test.txt");                    //将文件 test.txt 中内容读取到数组 $line 中
foreach( $line as $file)                   //浏览 $line 数组
{
    echo $file.“<br>”;                     //输出内容
}
? >
```

7. readfile（）函数

readfile（）函数用于输出一个文件的内容到浏览器中，语法格式如下：

```
int readfile(string $filename [, bool $use_include_path [, resource $context ]])
```

例如，读取当前工作目录中的 html 目录下的 test.txt 文件中的内容到浏览器中：

```
<? php
$filename="test.txt";
$num=readfile( $filename);                 //输出文件的所有内容
echo $num;                                 //输出读取到的字节数
```

```
? >
```

7.3.5 文件的写入和追加

文件在写入前需要打开文件，如果文件不存在则先要创建它。在 PHP 中没有专门用于创建文件的函数，一般可以使用 fopen（）函数来创建，文件模式可以是“w”“w+”“a”“a+”。

例如，下面的代码将在 D 盘 data 目录下新建一个名为 index.txt 的文件（data 目录存在）：

```
<? php
 $handle=fopen("D:\data\index.txt", "w");
? >
```

1. fwrite（）函数

文件打开后，向文件中写入内容可以使用 fwrite（）函数，语法格式如下：

```
int fwrite(resource  $handle , string  $string [ , int  $length ])
```

说明：参数 $handle 是写入的文件句柄，$string 是将要写入文件中的字符串数据，$length是可选参数，如果指定了 $length，则当写入了 $string 中的前 $length 个字节的数据后停止写入，否则一直写入，直到字符串末尾。

【例 7.3-11】

```
<? php
 $fp=fopen("new.txt","w");
fwrite( $fp,'这是写入的一行话\n');
fwrite( $fp,'最多写入 12 个字符\n',12);
close( $fp);
? >
```

这样就会将“这是写入的一行话 \ n”写入 new.txt 中，如果 new.txt 不存在，则 fopen（）函数会自动创建文件；如果 new.txt 已经存在并且有内容，则会删除 new.txt 中的内容再写入。

fwrite（）函数还有一个别名为 fputs（）。fwrite（）函数还可用于写入二进制文件。

注意：如果以写入方式“w”打开文件，写入的数据将覆盖旧数据，如果不想覆盖之前的数据而将新数据添加到文件末尾，可以使用追加模式“a”来打开文件。

【例 7.3-12】

```
<? PHP
 $fp=fopen("new.txt","w+");
fwrite( $fp,'这是写入的一行话\n\r');
rewind( $fp);                          //将指针指向文件开头
 $str=fread( $fp,20);                  //读取文中前 20 个字符保存到 $srt 中
echo  $str;
fclose( $fp);
```

```
? >
```

提示：写入后文件指针指向了文件末尾，要读取文件内容的话，需要先将指针移回文件开头。

2. file_put_contents（）函数

file_put_contents（）函数的功能与依次调用 fopen（）、fwrite（）及 fclose（）函数的功能一样。语法格式如下：

```
int file_put_contents(string $filename , string $data [ , int $flags [ , resource $context ]])
```

说明：$filename 是要写入数据的文件名。$data 是要写入的字符串，$data 也可以是数组，但不能为多维数组。在使用 FTP 或 HTTP 向远程文件写入数据时，可以使用可选参数 $flags 和 $context，这里不具体介绍。写入成功后函数返回写入的字节数，否则返回 FALSE。

【例 7.3-13】

```
<? php
 $str= "这是文件 1";
 $array=array("将数组","内容写入","文件 2 中");
file_put_contents("html/1.txt", $str);//将 $str 写入 html 目录下的 1.txt 文件中
file_put_contents("html/2.txt", $array);//将 $array 写入 html 目录下的 2.txt 文件中
? >
```

7.4　文件的上传与下载

在 Web 开发中，经常需要将本地文件上传到 Web 服务器上，也可以从 Web 服务器上下载一些文件到本地磁盘。文件的上传和下载应用十分广泛，在 PHP 中可以接受来自几乎所有类型浏览器上传的文件，PHP 还允许对服务器的下载进行控制管理。在 PHP 中实现文件的上传下载，都需要使用文件系统函数来完成。处理上传文件的函数是 move_uploaded_file（），实现文件下载可以使用 header（）和 readfile（）函数完成。

7.4.1　相关设置

1. php.ini 文件设置

要想在 PHP 程序中顺利实现文件上传功能，首先要在 php.ini 文件中设置文件上传的一些选项。找到 File_Uploads 项，分别设置以下选项。

file_uploads：值为 on，表明服务器支持文件上传，否则不支持。

uploads_tmp_dir：上传文件的临时目录。默认情况下是注释，删除行开头的分号即可启动，文件在成功上传之前首先被保存在该目录下。

uploads_max_filesize：服务器允许上传文件的最大值。

相关配置如图 7-2 所示。

图 7-2　php. ini 配置文件上传设置

2. 全局变量 $_FILES

全局变量 $_FILES 存储的是上传文件的相关信息，该变量是一个二维数组，其存储的信息见表 7-8。

表 7-8　全局变量 $_FILES 信息

元素名	说　明
$_FILES［filename］［"name"］	存储了上传文件的名称
$_FILES［filename］［"size"］	存储了上传文件的大小，单位为字节
$_FILES［filename］［"tmp_name"］	存储了上传文件被存储的临时目录
$_FILES［filename］［"type"］	存储了上传文件的类型
$_FILES［filename］［"error"］	存储了上传文件的结果。若为 0，说明上传成功

7.4.2　文件上传

PHP 主要使用 move_uploaded_file（）函数以及全局变量 $_FILE 来实现文件上传。move_uploaded_file（）函数的语法格式如下：

bool move_uploaded_file(string filename,string destination);

函数功能：将文件上传到指定的目录下。如果上传成功，函数返回 TRUE，否则返回 FALSE。

函数参数说明见表 7-9。

表 7-9　move_uploaded_file（）函数参数说明

参　数	含　义
filename	规定上传文件的临时名称
destination	规定文件上传后新的路径及名称

1. 上传单个文件

将单个文件上传到服务器。

首先创建一个网页文件 upfile. php，其中包含 form 表单及一个文件上传控件（name 属性值为 up_file）。代码如下：

```
<html>
<head>
<title>单个文件上传</title>
</head>
<body>
<table border="0" width="790" cellpadding="0">
    <form method="POST" enctype="multipart/form-data" action="upload_file.php">
    <tr>
        <td width="120"><p align="left">上传单个文件:</td>
        <td> <p align="left">
        <input type="file" name="up_file">
        <input type="submit" value="提交" name="B1"> </p>
        </td>
    </tr>
    </form>
</table>
</body>
    </html>
```

程序运行结果如图 7-3 所示。

图 7-3　单个文件上传

创建 upload_file. php 文件，该程序文件的功能是将文件上传至服务器的指定文件夹下。程序代码如下：

```
<? php
    header("Content-Type:text/html;charset=gb2312");
    //判断是否有上传文件
    if(! empty( $_FILES["up_file"]["name"]))
    {
        //获取上传文件的临时文件名
        $file_name= $_FILES["up_file"]["tmp_name"];
        //上传文件存放目录及保存新文件名
        $destination="./upload/new_file.doc";
        if( $_FILES["up_file"]["size"]>1000000)
```

```
            echo "文件太大";
        else {
            //上传文件
            @ move_uploaded_file( $file_name, $destination);
            echo "文件上传成功!";
            echo "<a href=a.php>返回</a>";
        }
    }
? >
```

注意：使用 move_upload_file（）函数上传文件，在创建 form 表单时，必须设置表单属性为 entype=” multipart/form-data”。

2. 同时上传多个文件

多个文件上传和单个文件上传实现的方法是相似的，只需要在表单中多提供几个文件上传域，并指定 name 属性值为同一个数组即可。代码如下：

```
<html>
<head><title>多文件上传</title></head>
<body>
<h3 align="center">多文件上传系统</h3>
<p>请选择要上传的三张图片文件</p>
<form action="4-26.php" method="post" enctype="multipart/form-data">
    文件 1:<input type="file" name="upfile[]"/><br><br>
    文件 2:<input type="file" name="upfile[]"/><br><br>
    文件 3:<input type="file" name="upfile[]"/><br><br>
  <input type="submit" value="上 传 "/>
</form>
</body>
</html>
```

运行结果如图 7-4 所示。

图 7-4　多文件上传域的添加

为了简化程序，下面实例是通过简单的 JavaScript 脚本和鼠标事件 add（）来根据实际情况添加上传文件域的。

【例 7.4-1】 通过点击“添加要上传的文件”按钮，实现多个文件上传到服务器。

```
<html>
<head><title>多个文件上传系统</title></head>
<body>
<h3>多个文件上传系统</h3>
<script type="text/javascript">
function add()
{
    upload.innerHTML+="请选择要上传文件:<input type=file name='userfile[]'><br>";

}</script>
<form action="" method="post" enctype="multipart/form-data" name="form1">
    <input type="hidden" name="MAX_FILES_SIZE" vlaue="1048600" id="hiddenfield">
    <div id=upload>请选择要上传文件:<input name=userfile[] type="file"><br></div>
    <input type="button" onclick="add()" value="添加要上传的文件">
    <input type="submit" value="上传文件">
</form>
<? PHP
for( $j=0; $j<@ sizeof( $_FILES['userfile']['error']); $j++)
    {
//提取文件名称,可以上传任何文件。
 $ext=substr( $_FILES['userfile']['name'][ $j], strrpos( $_FILES['userfile']['name'][ $j],"."));  $upfile='D:/www/'. $_FILES['userfile']['name'][ $j];//文件上传到的路径
        if(is_uploaded_file( $_FILES['userfile']['tmp_name'][ $j]))
        {
            if(! move_uploaded_file( $_FILES['userfile']['tmp_name'][ $j], $upfile))
        {
            echo  $j."问题在于:无法上传到指定路径";
            exit;
        }
    }
    else
    {
        echo  $j."问题在于:上传的文件格式不符合要求。";
        echo  $_FILES['userfile']['name'][ $j];
        exit;
    }
    echo '第'. $j.'个文件上传成功。'."<br>";
}
```

```
? >
</body>
</html>
```

运行结果如图 7-5 所示。

图 7-5　添加多个上传文件域

7.4.3　文件下载

在网页中可以通过链接下载文件，如：

```
<a href="源文件路径">点击下载</a>
```

但这种方式安全性不高，很容易暴露文件所在的路径，可能会有安全隐患。要实现安全的文件下载，在 PHP 中可以使用 header（）和 readfile（）两个函数来完成。

header（）函数的作用是向浏览器发送正确的 HTTP 报头，报头指定了网页内容的类型、页面的属性等信息。header（）函数的功能有页面跳转、指定网页内容、文件下载等，下面只介绍文件下载功能。

header（）文件下载功能，需要结合 readfile（）函数来实现，例如，下载 upload 目录下的 file. php 文件可以使用以下代码：

【例 7.4-2】

```
<? php
 $textname = "./upload/file.php";                                  //源文件路径
 $newname = "newfile.php ";                                        //新文件名
header("Content-type: text/plain");                                //设置下载的文件类型
header("Content-Length:".filesize( $textname));                    //设置下载文件的大小
header("Content-Disposition: attachment; filename= $newname");     //设置下载文件名
readfile( $textname);                                              //读取文件
? >
```

以上代码执行后会弹出保存文件对话框，选择新地址保存该文件即实现了文件的下载，新文件名为“newfile. php”，运行结果如图 7-6 所示。

图 7-6　文件下载

注意：如果在旧版本中调用 header（）函数时会出错，原因是在调用 header（）函数前就已经输出了内容，如 HTML 标记、空格等，所以要在 PHP 配置文件中将“output_buffering”输出缓存选项的值修改为 on。在新版本中不需要修改。

7.5　综合案例

本节主要介绍利用 PHP 文件上传的功能实现提交“PHP 程序设计作业”的页面。

1. 分析

创建一个静态网页文件 homwork. php，其中包含 form 表单（（enctype 属性值为“multipart/form-data”，method 为“POST”方式）和文件上传控件（name 属性值为“up_file”）。文件启动界面如图 7-7 所示。

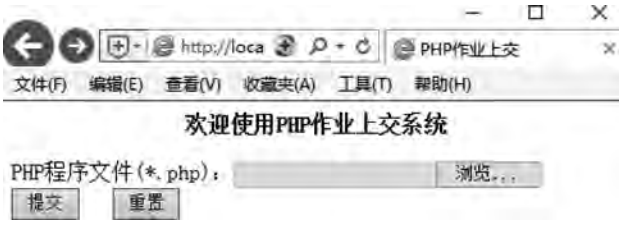


图 7-7　上交系统界面

2. 运行结果

创建 PHP 程序文件 upload_file. php，该文件的功能是上传学生提交的 PHP 程序设计作业文件至服务器指定的目录“PHP_file”下，并读取文件的内容，显示在页面的文本域控件内。程序运行结果如图 7-8 所示。

图 7-8　上交成功并显示文件内容

3. 实现代码

homework. php 文件代码如下：

```
<html>
<head>
<meta http-equiv="Content-Language" content="zh-cn">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>提交 PHP 作业</title>
</head>
<body>
<form method="POST" enctype="multipart/form-data" action="uploadfile.php">
<p align="center"><b>欢迎使用 PHP 程序设计作业上交系统</b></p>
PHP 程序设计作业文件( *.php):
<input type="file" name="up_file"><br/>
<input type="submit" value="上交" name="B1">
<input type="reset" value="取消" name="B2">
</form>
</body>
</html>
```

uploadfile.php 程序文件代码如下：

```
<? php
    header("Content-Type:text/html;charset=gb2312");
    //判断是否有上传文件
    if(! empty( $_FILES["up_file"]["name"]))
    {
        //获取上传文件的临时文件名
        $file_name= $_FILES["up_file"]["tmp_name"];
        //上传的目录及新文件名
        $destination="./php_file/". $_FILES["up_file"]["name"];
        if( $_FILES["up_file"]["size"]>1000000)
```

```
            echo "文件太大";
        else
        {
            //上传文件
            move_uploaded_file( $file_name, $destination);
            //读取文件并显示在文本区内
            echo "<textarea rows=10 name=S1 cols=80 style='font-size: 12pt';>",
            file_get_contents( $destination);
            echo "</textarea>";
        }
    }
? >
```

本章小结

本章主要介绍了文件、目录函数的使用，并通过上传下载文件的操作加深对函数的理解。文件操作主要包括打开文件、读取文件、写入文件、删除和复制文件等，文件目录的操作主要有打开目录、读取目录、创建目录、删除和改变目录等。

习　题

一、简答题

1. 操作文件的一般步骤有哪些？如何读取指定长度的文件内容？
2. 如何创建一个文件？如何向文件中写入指定内容？
3. 在实现文件上传和下载时需要注意哪些方面？

二、编程题

1. 编写程序，递归遍历自己的网站根目录，将所有的文件显示出来。
2. 编写程序，统计网站根目录中所有文件的大小并输出显示。

第 8 章

图形图像处理技术

【本章内容简介】

PHP 提供了许多图像处理函数，利用这些函数可以创建、打开和处理图像，并且既可以在 Web 浏览器中显示图像，也可以把它保存到硬盘上。利用 PHP 提供的 GD 库，可以绘制直线、椭圆和矩形，填充图像的区域，在图像中创建文本，此外，还可以读写 JPEG、PNG、WBMP、XBM 和 GIF 等格式的图像文件，并且利用 GD 库可以在 PHP 脚本中创建和处理非常复杂的图像。更方便的是 PHP 不仅可以将动态处理后的图像以不同格式保存在服务器中，还可以直接将图像流输出到浏览器，例如验证码、电子相册等动态图像处理。

【本章学习要求】

通过本章的学习，能够了解计算机图形技术的一些基础知识，GD 库的配置，创建和输出图像，图像的常用处理技术等。还需要做到以下几点：

(1) 了解几个计算机图形基本概念。

(2) 掌握 GD 库的使用以及处理的文件类型。

(3) 掌握常用的图像处理技术。

(4) 了解在图像中插入文本的方法。

8.1 PHP 中 GD 库的使用

在 PHP 中，有一些简单的图像函数是可以直接使用的，但大多数要处理的图像都需要在编译 PHP 时加上 GD 库。GD 库是一个动态创建图像的开源的函数库，可以从官方网站 http://libgd.github.io/下载最新版的 GD 库，目前 GD 库最新版为 2.2.5，所以 GD 库简称为 GD2。除了安装 GD 库之外，在 PHP 中还可能需要其他的库，这可以根据需要支持哪些图像格式而定。不同的 GD 版本支持的图像格式不完全一样，最新的 GD 库版本支持 GIF、JPEG、PNG、WBMP、XBM 等格式的图像文件。

如果要在 PHP 中配置 GD2 属性，需要打开 php.ini 文件，找到选项“；extension = php_gd2.dll”，将之前的分号删除，如图 8-1 所示。重新启动 Apache 服务器后，就可以使

用 GD2 函数库了。目前 PHP 已将 GD2 函数库作为默认的扩展。

图 8-1　GD2 图像库的启用

加载 GD2 成功后，可以通过 phpinfo（）函数来获取 GD2 函数库的安装信息。编辑 PHP 程序文件，写入代码：

```
<? php
phpinfo( );
? >
```

该程序运行结果如图 8-2 所示。

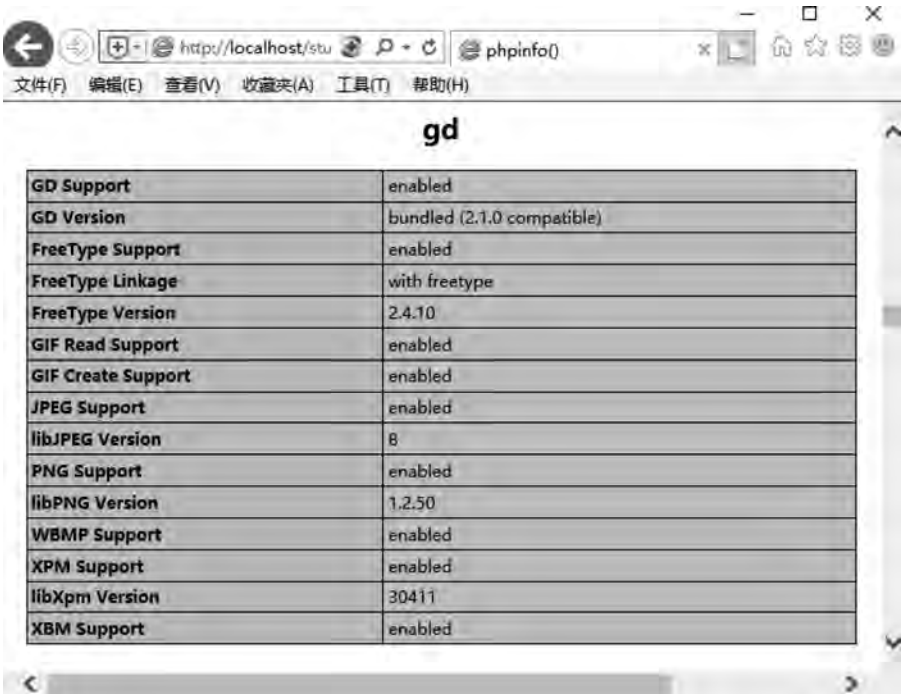


图 8-2　GD2 函数库信息表

8.2　常见的图像处理技术

PHP 中的 GD2 函数库用于创建或处理图像，通过 GD2 函数库可以生成统计图表、动

态图形、图形验证码等。通过 GD 库处理图像的操作，都是先在内存中处理，操作完成以后再以文件流的方式输出到浏览器或保存在服务器的硬盘中。在 PHP 程序中处理图像的操作主要分为以下 4 个步骤。

（1）创建画布。

（2）在画布上绘制图形或输入文本。

（3）保存并输出图形。

（4）销毁图像资源。

8.2.1 创建背景图形

首先来看一个例子。

【例 8.2-1】 绘制一条直线，并输出到浏览器中。

新建 EX8-1.php 文件，输入以下代码：

```
<? php
 $image = imagecreate(500,500);                          //创建一个背景(默认为黑色)
 $bgcolor = imagecolorallocate( $image,255,255,255);     //设置背景颜色为白色
 $red = imagecolorallocate( $image,255,0,0);             //定义颜色为红色
 $black = imagecolorallocate( $image,0,0,0);             //定义颜色为黑色
imageline( $image,150,50,250,150, $black);               //设置直线颜色为黑色
imageline( $image,50,50,150,150, $red);                  //设置直线颜色为红色
header("Content-type:image/png");                        //向浏览器发送头信息,输出 PNG 图片
imagepng( $image);                                       //输出图形
imagedestroy( $image);                                   //清除资源
? >
```

运行结果如图 8-3 所示。

图 8-3 绘制直线

说明：要查看结果，只需要浏览 PHP 页面。如果要在 Web 页面中使用该图片，可以使用如下 HTML 代码：

```
<img src = "ex8_1.php" >
```

注意：在使用 PHP 输出图像时，一定要使用 header（）函数向浏览器发送头信息，

指定浏览器输出相应格式的文件。若头信息为“Content-type：image/png”，则表示浏览器将输出 PNG 格式的图片。如果在程序中输出文本信息，则在浏览器中不会显示。

以上例子只是简要地说明了创建一个图形的过程，下面对这个过程进行详细介绍。

1. 创建背景图形

所有的绘图设计都需要在一个背景图片上完成，而画布实际上就是在内存中开辟的一块临时区域，用于存储图像的信息。以后的图像操作都将基于这个背景画布，该画布的管理就类似于我们在画画时使用的画布一样。

创建画布可以通过函数 imagecreate（）和函数 imagecreatecolor（）来完成，其语法格式如下：

```
resource imagecreate(int x_size, int y_size)
resource imagecreatetruecolor(int x_size, int y_size)
```

函数功能：创建一个指定大小的画布，x_size 和 y_size 是空白图像的宽和高。其中，imagecreate（）函数创建的是一个基于普通调色板的图像，通常支持 256 色；imagecreatecolor（）函数创建的是一个 24 位真彩色的画布，它可以显示 1670 万种颜色，背景颜色默认为黑色。例如：

```
$image1=imagecreate(400, 400);              //创建一个背景,背景默认为黑色
$image2= imagecreatetruecolor (300, 300);  //创建宽为 300 像素,高为 300 像素的画布
```

这两个函数如果创建成功则返回值为一个图像标识符，否则，不会像其他函数一样返回 FALSE，这时，可以使用 die（）函数来捕获错误信息。例如：

```
$image=imagecreate(400,400) or die("创建图形失败!");
```

背景创建之后，可以通过函数 imagesX（）和 imagesY（）来获取画布的高度和宽度。

2. 使用已有图片创建新图形

除了可以创建空白的背景图形外，还可以将已有的图片作为背景图形来创建新的图形，这就需要用到 imagecreatefrom…系列函数，其中最常用的是：imagecreatefromgif（）函数，可以根据已有的 GIF 图片创建新图形；imagecreatefromjpeg（）函数，可以根据已有的 JPEG 图片创建新图形；$imagecreatefrompng（）函数，可以根据已有的 PNG 图片创建新图形。imagecreatefromjpeg（）和 imagecreate（）函数的用法基本相同，不同之处在于后者需要将新图像的宽、高传递给它，而前者需要传递一个字符串参数，即一个现有图像的文件名，然后该函数将会返回一个图像资源供用户使用。语法格式如下：

```
resource imagecreatefromjpeg(string  $filename)
```

例如：

```
<? php
 $image=imagecreatefromjpeg("sun.jpg");        //根据 sun.jpg 图片创建一个新图形
header("Content-type:image/jpeg");
imagejpeg( $image);
imagedestroy( $image);
? >
```

使用这些函数后，已有的图片将作为新的图形背景，之后的操作和 imagecreate（）函

数创建背景图形的操作方法一样。

3. 设置颜色

画布创建完成后，接下来就要设置图像的颜色。颜色的设置可以通过函数 imagecolorallocate（）来完成，其语法格式如下：

int imagecolorallocate(resource image, int red, int green, int blue)

函数功能：为画布中创建的图像填充颜色，并且为画布本身填充颜色。imagecolorallocate（）函数必须被调用以创建每一种用在 $image 所代表的图形中的颜色。

函数参数说明见表 8-1。

表 8-1　设置颜色函数参数说明

参　数	说　明
image	图像的标识符
red	红色颜色值，取值 0~255，或者十六进制 0x00~0xFF
green	绿色颜色值，取值 0~255，或者十六进制 0x00~0xFF
blue	蓝色颜色值，取值 0~255，或者十六进制 0x00~0xFF

【例 8.2-2】 为画布填充颜色。

程序如下：

```
<? php
    header("Content-type:image/jpeg");
    $image=imagecreate(300,300);                          // 创建画布
    $color=imagecolorallocate( $image,160,220,100);       //为画布填充颜色
    //预设定一些颜色
    $white = imagecolorallocate( $image, 255, 255, 255);  //白色
    $black = imagecolorallocate( $image, 0, 0, 0);        //黑色
    //十六进制方式
    $white = imagecolorallocate( $image, 0xFF, 0xFF, 0xFF);
    $black = imagecolorallocate( $image, 0x00, 0x00, 0x00);
? >
```

注意：第一次调用 imagecolorallocate（）函数时会给基于调色板的图形（即用 imagecreate（）函数创建的图形）填充背景颜色。例子中 $white、$black 颜色定义后就可以在其他函数中使用该颜色对其他图形中的某一部分进行着色。

4. 输出图形

当画布创建完成，颜色填充后，就可以生成图像和输出文字了。在 PHP 中可以直接生成的图像格式有 GIF，JPEG，PNG 和 WBMP 等 4 种。如果生成的图像要输出到浏览器上，还需要在 PHP 程序中发送标头信息来设置 MIME 文件类型。针对 GIF、JPEG、PNG 和 WBMP 这 4 种图像分别使用的标头信息文件信息如下：

```
header("Content-type: image/gif");
header("Content-type: image/jpeg")
```

header("Content-type: image/png")

header("Content-type: image/wbmp")

如果需要将已经绘制的图形输出到浏览器或文件中，可以使用相应的函数来完成。例如，可以使用 imagegif（）将图形以 GIF 格式输出，使用 imagejpeg（）将图形以 JPEG 格式输出，使用 imagepng（）将图形以 PNG 格式输出，使用 imagewbmp（）将图形以 WBMP 格式输出等。语法格式如下：

bool imagegif(resource $image [, string $filename])

bool imagejpeg(resource $image [, string $filename [, int $quality]])

bool imagepng(resource $image [, string $filename])

bool imagewbmp(resource $image [, string $filename [, int $foreground]])

函数的参数说明见表 8-2。

表 8-2　各函数参数说明

参　数	说　明
$image	规定由 imagecreate（）或 imagecreatetruecolor（）创建的图像标识符
$filename	可选。指定输出图像的文件名，若忽略，则原始图像流将被直接输出
$quality	可选。规定图像质量。取值范围从 0（最差质量，文件最小）到 100（最佳质量，文件最大），默认为 75
$foreground	可选。规定前景色，默认前景色为黑色

【例 8.2-3】创建画布并输出规定格式图形。

程序如下：

```
<? php
header( "Content-type:image/jpeg" );                    //发送头信息,使用脚本输出 JPEG 格式文件
 $image = imagecreate( 300,150 );                       //创建画布
 $color = imagecolorallocate( $image,160,220,100 );     //填充画布
imagejpeg( $image );                                    //在浏览器中输出图形
? >
```

程序运行结果如图 8-4 所示。

图 8-4　运行结果

5. 销毁图像

PHP 程序处理图像的最后一个环节就是销毁图像。所谓销毁图像就是释放内存与指定图像的存储单元，可以通过 imagedestroy（）函数来完成。该函数的语法格式如下：

```
bool imagedestroy(resource $image)
```

函数功能：释放与图像标识 image 关联的内存。其中参数 $image 是由图像创建函数返回的图像标识符。

【例 8.2-4】 创建画布并输出、销毁图像。

程序如下：

```
<? php
    header("Content-type:image/jpeg");
    $image=imagecreate(300,100);                    // 创建画布
    $color=imagecolorallocate($image,160,220,100);  //填充画布
    imagejpeg($image);                              //输出 JPEG 图像
    imagedestroy($image);                           //销毁图像资源
? >
```

8.2.2 绘制图形

在 PHP 中绘制图形的函数非常丰富，包括点、线、文本文字、颜色块以及各种几何图形等可以想象出来的平面图形，都可以通过 PHP 中提供的各种画图函数完成。在这里只介绍一些常用的图像绘制，如果使用本书没有介绍过的函数，可以参考 PHP 手册实现。另外，这些图形绘制函数都需要使用画布资源，并在画布中的位置通过坐标（原点是该画布的左上角的起始位置，以像素为单位，沿着 *X* 轴正方向向右延伸，*Y* 轴正方向向下延伸）决定，而且还可以通过函数中最后一个参数，设置每个图形的颜色。画布中的坐标系统如图 8-5 所示。

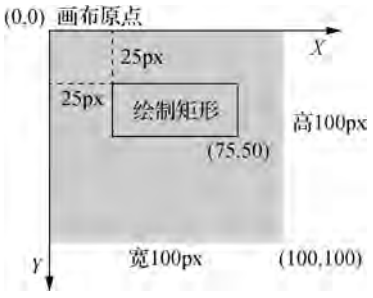


图 8-5　画布中的坐标系统

1. 绘制几何图形

（1）绘制点。在 PHP 中绘制点，可以通过函数 imagesetpixel（）来完成。该函数的语法格式如下：

```
bool imagesetpixel(int $image,int $x,int $y,int $color)
```

函数功能：在 $image 图像上用颜色 $color 在坐标（x，y）上绘制一个点。

【例 8.2-5】 在画布上绘制点。

程序如下：

```
<? php
header( "Content-type:image/jpg" );
 $image=imagecreate(300,200);                    // 创建画布
 $color=imagecolorallocate( $image,215,215,215);  //画布的填充颜色
 $color1=imagecolorallocate( $image,0,0,0);       //点的颜色
for( $i=10; $i<200; $i+=10)                       //在画布上绘制点
      imagesetpixel( $image, $i+30, $i+20, $color1);
imagejpeg( $image);                               //输出 JPEG 图像
imagedestroy( $image);                            //销毁图像资源
? >
```

程序运行结果如图 8-6 所示。

图 8-6　程序运行结果

（2）绘制直线。在 PHP 中绘制直线，可以通过函数 imageline（）来完成。该函数的语法格式如下：

```
bool imageline(int  $image,int  $x1,int  $y1,int  $x2,int  $y2,int  $color)
```

函数功能：在 $image 图像上用颜色 $color 从坐标（$x1，$y1）到（$x2，$y2）上绘制一条直线。

【例 8.2-6】 在画布上绘制一条直线。

程序如下：

```
<? php
header( "Content-type:image/jpeg" );
 $image=imagecreate(300,200);                    // 创建画布
 $color=imagecolorallocate( $image,215,215,215);
 $color1=imagecolorallocate( $image,0,0,0);       //线的颜色
```

```
imageline( $image,20,20,150,150, $color1);
imagejpeg( $image);                                    //输出 JPEG 图像
imagedestroy( $image);                                 //销毁图像资源
? >
```

程序运行结果如图 8-7 所示。

图 8-7　程序运行结果

（3）画一个矩形。几何学中最重要的图形就是矩形，绘制矩形可以使用 imagerectangle（）函数来完成。语法格式如下：

```
bool imagerectangle( resource  $image , int  $x1 , int  $y1 , int  $x2 , int  $y2 , int  $color)
```

说明：imagerectangle（）函数在已经创建的图形 $image 上使用 $color 颜色画出一个矩形，矩形的左上角坐标为（ $x1， $y1），右下角坐标为（ $x2， $y2）。

（4）画一个椭圆。使用 imageellipse（）函数可以画出一个椭圆，语法格式如下：

```
bool imageellipse ( resource  $image , int  $cx , int  $cy , int  $w , int  $h , int  $color )
```

（5）画一个椭圆弧。imageellipse（）函数用于创建一个完整的椭圆，而如果只要创建一个椭圆弧，可以使用 imagearc（）函数。语法格式如下：

```
bool imagearc( resource  $image,int  $cx, int  $cy, int  $w, int  $h, int  $s, int  $e, int  $color)
```

说明：imagearc（）函数以坐标（ $cx， $cy）为中心在图形 $image 上画一个椭圆弧。$w 和 $h 分别指定了椭圆的宽度和高度，当高度和宽度相等时，画出来的就是圆弧。起始和结束点用 $s 和 $e 参数以角度指定。0°位于三点钟位置，以顺时针方向绘画。例如：

```
imagearc( $image,100,100,150,150,0,180, $color);       //画一个半圆弧
imagearc( $image,200,100,150,150,0,360, $color);       //画一个圆
imagearc( $image,300,100,200,150,90,180, $color);      //画一个椭圆弧
```

【例 8.2-7】在画布上绘制一个矩形，并在矩形内绘制一个椭圆、一个圆、一个半弧。程序如下：

```
<? php
 $image=imagecreate(500,300);                          //创建背景图形
 $background=imagecolorallocate( $image,215,215,215);  //背景色设为灰色
 $red=imagecolorallocate( $image,255,0,0);              //定义红色
```

```
imagerectangle( $image,0,0,400, 200 , $red);          //画一个矩形
imageellipse( $image,100,100,200,100, $red);          //画一个椭圆
imageellipse( $image,200,100,200,200, $red);          //画一个圆
imagearc( $image,350,80,100,70,180,360, $red);        //画一个半弧
header("Content-type: image/gif");                    //发送头信息
imagegif( $image);                                    //输出图形
imagedestroy( $image);                                //清除资源
? >
```

程序运行结果如图 8-8 所示。

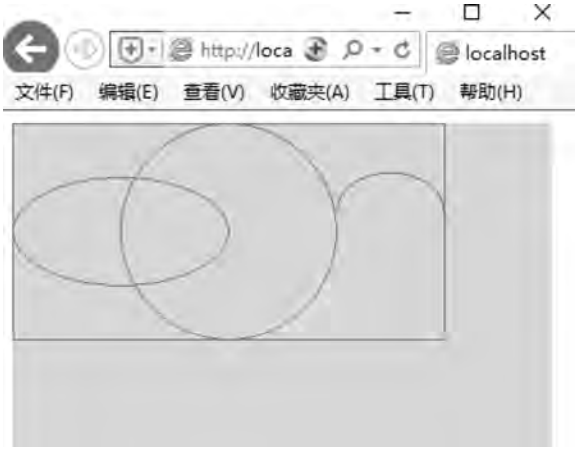


图 8-8　程序运行结果

（6）绘制多边形。多边形的绘制可以通过函数 imagepolygon（）来完成。该函数的语法格式如下：

```
bool imagepolygon(resource  $image,array  $points,int  $num_points,int  $color)
```

函数参数说明见表 8-3。

表 8-3　imagepolygon（）函数参数说明

参　数	说　明
$image	规定由 imagecreate（）或 imagecreatetruecolor（）创建的图像标识符
$points	一个 PHP 数组，包含多边形各顶点的坐标，即 point［0］=x0，points［1］=y0，point［2］=x2，points［3］=y2，依此类推
$num_points	多边形顶点的个数。参数 num_points 的值不大于参数 points 中提供的顶点的个数
$color	规定多边形边线的颜色

【例 8.2-8】在画布上绘制一个多边形和五边形。

程序如下：

```
<? php
 $image=imagecreate(200,200);                         //创建背景图形
```

```
$background=imagecolorallocate( $image,215,215,215);                    //背景色设为灰色
$blue=imagecolorallocate( $image,0,0,255);                              //定义蓝色
$red=imagecolorallocate( $image,255,0,0);                               //定义红色
$coords=array(100,25,24,80,53,170,147,170,176,80);                      //定义坐标数组
imagepolygon( $image, $coords,5, $blue);                                //画出五边形
imagepolygon( $image,array(20,20,175,90,110,165,75,75),4, $red);        //画出多边形
header("Content-type: image/gif");
imagegif( $image);
imagedestroy( $image);
? >
```

程序运行结果如图 8-9 所示。

图 8-9　程序运行结果

PHP 绘制几何图形时，默认的边线宽度为 1 像素。如果想要改变边线的宽度，可以通过函数 imagesetthickness（）来完成。函数的语法格式如下：

```
bool imagesetthickness(resource image,int thickness)
```

函数功能：设置直线、矩形、多边形等图像的边线的宽度为 thickness 像素。如果成功，则返回 TRUE，否则返回 FALSE。

在例 8.2-8 中，如要设置所绘制的图形边线的宽度为 5 像素，可以在设置边线颜色的语句之后添加命令：

```
imagesetthickness( $image,5);
```

2. 绘制文本

在 PHP 中，可以通过 GD2 函数库来绘制英文字符串，也可以绘制中文字符串。

（1）绘制英文字符串。绘制英文字符串可以应用 imagestring（）函数，其语法格式如下：

```
bool imagestring(resource $image,int $font,int $x,int $y,string $str,int $color)
```

函数功能：用 $color 颜色将字符串 $str 水平绘制到 $image 所代表的坐标（$x，$y）处。其中，坐标（$x，$y）是指字符串左上角的坐标。若 $font 的值为 1、2、3、4 或 5，则使用内置字体。

【例 8.2-9】 使用 imagestring（）函数绘制英文字符串。

程序如下：

```
<? php
    header("Content-type:image/jpeg");
    $im=imagecreate(300,100);
    $col=imagecolorallocate($im,215,215,215);
    $col1=imagecolorallocate($im,0,0,0);
    imagestring($im,5,85,45,"Hello PHP!",$col1);
    imagejpeg($im);
    imagedestroy($im);
? >
```

程序运行结果如图 8-10 所示。

图 8-10　程序运行结果

（2）绘制一个字符。使用 imagechar（）函数可以在图形上水平输出一个字符，语法格式如下：

```
bool imagechar(resource $image, int $font, int $x, int $y, string $c, int $color)
```

说明：函数用颜色 $color 将字符 $c 画到 $image 的（$x，$y）坐标处。如果 $c 是字符串，则只输出第一个字符。若 $font 的值为 1、2、3、4 或 5，则使用内置字体，值为 5 时字体最大，为 1 时最小。例如：

```
imagechar($image,5,50,50,'C',$color);
```

imageup（）函数可以垂直地输出一个字符，用法与 imagechar（）函数相同。

（3）绘制中文字符串。绘制中文字符串可以应用 imagettftext（）函数将中文字符串水平绘制，其语法格式如下：

```
array imagettftext(resource $image,float $size,float $angle,int $x,int $y,int $color,string $fontfile,
```

string $text)

函数功能：用 $color 颜色将中文字符串 $text 绘制到 $image 所代表的坐标（$x，$y）处。其中，坐标（$x，$y）是指字符串左上角的坐标。函数的参数说明见表 8-4。

表 8-4　imagettftext（）函数参数说明

参　数	说　明
$image	规定由 imagecreate（）或 imagecreatetruecolor（）创建的图像标识符
$size	规定字体的大小，单位为磅
$angle	规定字体的角度，单位为度
$x，$y	规定文字的 x 坐标值和 y 坐标值
$color	规定文字的颜色
$fontfile	规定字体的文件名称
$text	规定要输出的中文字符串内容

【例 8.2-10】绘制指定字体的中文字符串。

程序如下：

```
<? php
    header("Content-type:image/jpeg");
    $text = iconv("gb2312","utf-8","中国,你好!");      //将字符编码转换为 UFT-8 格式
    $im=imagecreate(400,100);
    $col=imagecolorallocate( $im,215,215,215);
    $col1=imagecolorallocate( $im,0,0,0);
    $font="STLITI.TTF";                                //字体文件
    imagettftext( $im,30,0,100,60, $col1, $font, $text);
    imagejpeg( $im);
    imagedestroy( $im);
? >
```

程序运行结果如图 8-11 所示。

图 8-11　程序运行结果

8.3　填充几何图形

在实际的绘图过程中，经常需要对图形中的某一区域填充颜色，这样图形才会变得更加美观。

1. 区域填充

区域填充就是针对图形中的某个区域进行填充。在 PHP 中可以使用 imagefill（）函数对某一区域进行颜色填充，语法格式如下：

```
bool imagefill( resource  $image, int  $x, int  $y, int  $color )
```

说明：函数在 $image 图形的坐标（ $x， $y）处用颜色 $color 执行区域填充，即与（ $x， $y）点颜色相同且相邻的点都会被填充上该颜色。

【例 8.3-1】 使用 imagefill（）填充图形。

程序如下：

```
<? php
    header( "Content-type:image/jpeg" );
     $im=imagecreate(300,200);
     $color=imagecolorallocate( $im,215,215,215);
     $color1=imagecolorallocate( $im,10,20,100);
    imagearc( $im,60,100,100,100,0,360, $color1);          //画圆
    imagefill( $im,60,100, $color1);                       //填充圆颜色
    imagejpeg( $im);
    imagedestroy( $im);
? >
```

程序运行结果如图 8-12 所示。

图 8-12　程序运行结果

2. 绘制矩形、多边形、椭圆和椭圆弧并填充颜色

PHP 还提供了填充矩形、多边形、椭圆形和椭圆弧的函数，如 imagefilledrectangle（ ）、imagefilledpolygon（ ）、imagefilledellipse（ ）和 imagefilledarc（ ）。

这四个函数的语法格式如下：

```
bool imagefilledrectangle( resource  $image, int  $x1, int
$y1, int  $x2, int  $y2, int  $color)
```

函数说明：在 $image 图像中绘制一个矩形，并用颜色 color 填充该矩形。矩形左上角的坐标为（x1，y1），右下角的坐标为（x2，y2）。

```
bool imagefilledpolygon( resource  $image, int  $x1, int  $y1, int  $x2, int  $y2, int  $color)
```

函数说明：在 $image 图像中绘制一个多边形，并用颜色 $color 填充。

imagefilledpolygon（）用法和 imagefilledrectangle（）相同。

bool imagefilledellipse(resource $image, int $cx, int $cy, int $w, int $h, int $color)

函数说明：使用 imagefilledellipse（）函数可以在已经创建的图形上画一个椭圆，并使用指定颜色进行填充。

bool imagefilledarc(resource $image, int $cx, int $cy, int $w, int $h, int $s, int $e, int $color, int $style)

函数说明：在 $image 中以坐标（$cx，$cy）为中心画一个椭圆弧。如果成功则返回 TRUE，失败则返回 FALSE。$w 和 $h 分别指定了椭圆的宽和高，$s 和 $e 参数以角度指定了起始和结束点。$style 指定了椭圆弧画出来的效果，值可以是以下几个：IMG_ARC_PIE（产生圆形边界）、IMG_ARC_CHORD（用直线连接起始和结束点）、IMG_ARC_NOFILL（指明弧或弦只有轮廓，不填充）和 IMG_ARC_EDGED（用直线将起始和结束点与中心点相连）。

【例 8.3-2】 在一个画布上分别绘制矩形、多边形、椭圆和椭圆弧并填充颜色。

程序如下：

```
<? php
   header("Content-type:image/jpeg");
    $im=imagecreate(500,400);
    $col=imagecolorallocate($im,215,215,215);
 $col1=imagecolorallocate($im,100,20,100);
 $col2=imagecolorallocate($im,255,0,0);
 imagefilledrectangle($im,10,10,50,50,$col1);          //绘制矩形
 imagefilledpolygon($im,array(100,25,24,80,53,170,147,170,176,80),5,$col2);  //绘制多边形
 imagefilledellipse($im,230,100,50,50,$col1);          //绘制椭圆
 imagearc($im,250,100,150,150,0,180,$col2);            //绘制圆弧
 imagejpeg($im);
 imagedestroy($im);
 ? >
```

程序运行结果如图 8-13 所示。

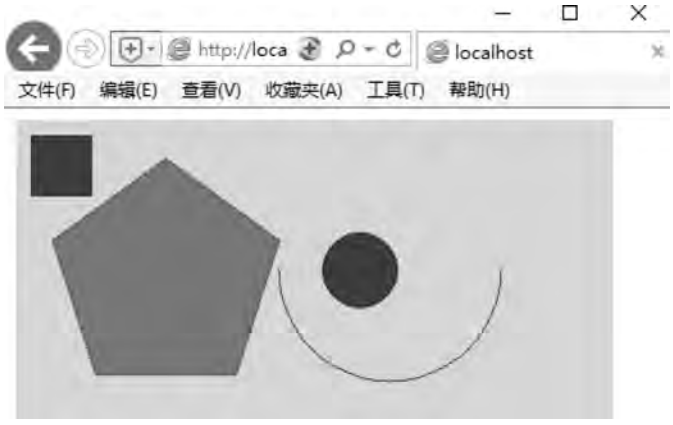


图 8-13　程序运行结果

8.4　图形的处理

1. 颜色处理

使用 imagefilltoborder（）函数可以为指定点进行颜色填充，如果遇到指定颜色的边界时，则停止填充。imagefilltoborder（）函数的语法格式如下：

```
bool imagefilltoboder(resource $image,int $x,int $y,int $boder,int $color)
```

函数功能：从坐标点（$x，$y）开始，用颜色 $color 开始填充，直到遇到颜色为 $boder的边界为止。

【例 8.4-1】 使用 imagefilltoborder（）填充图形。

程序如下：

```
<? php
    header("Content-type:image/jpeg");
    $im=imagecreate(300,200);
    $color=imagecolorallocate($im,215,215,215);          //画布的填充颜色
    $color1=imagecolorallocate($im,10,20,100);           //边线的颜色
    $color2=imagecolorallocate($im,150,150,150);         //填充的颜色
    imagerectangle($im,100,150,50,50,$color1);           //绘制矩形
    imagearc($im,60,100,100,100,0,360,$color1);          //绘制圆
    imagefilltoborder($im,60,100,$color1,$color2);       //填充圆
    imagejpeg($im);
    imagedestroy($im);
? >
```

程序运行结果如图 8-14 所示。

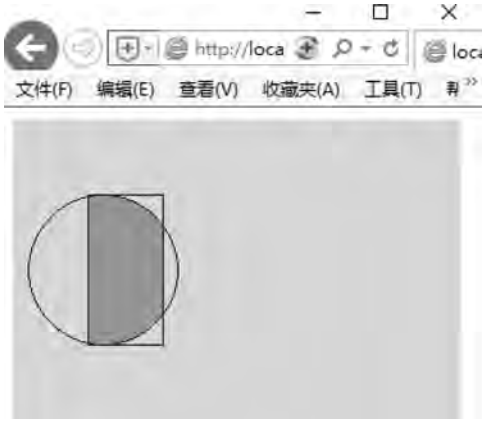


图 8-14　程序运行结果

注意：使用 imagefilltoborder（）函数进行填充时，边界内的所有颜色都会被填充，如果指定的边界色和该点颜色相同，则没有填充；如果图像中没有该边界色，则整幅图像都会被填充。

2. 定义透明颜色

使用 imagecolorallocatealpha（）函数也可以为指定的图形分配颜色，还可以设置颜色的透明度。语法格式如下：

```
int imagecolorallocatealpha ( resource $image , int $red , int $green , int $blue , int $alpha )
```

【例 8.4-2】 使用 imagecolorallocatealpha（）函数设置图形的透明度。

程序如下：

```
<? php
 $image = imagecreatetruecolor(200,200);
imagefill( $image,0,0,imagecolorallocate( $image,215,215,215));
 $red = imagecolorallocatealpha( $image,255,0,0,60);          //红色,透明度为 60
 $blue = imagecolorallocatealpha( $image,0,0,255,60);         //蓝色,透明度为 60
imagefilledellipse( $image,50,50,100,100, $red);              //画红色圆
imagefilledellipse( $image,70,70,100,100, $blue);             //画蓝色圆
header('Content-type: image/gif');
imagegif( $image);
imagedestroy( $image);
? >
```

程序运行结果如图 8-15 所示。

图 8-15　程序运行结果

3. 复制图片的一部分

使用 imagecopy（）函数能够复制图片的一部分到另一个图片中，语法格式如下：

```
bool imagecopy(resource $dst_im, resource $src_im, int $dst_x, int $dst_y, int $src_x, int $src_y,
int $src_w, int $src_h )
```

函数说明：将 $src_im 从坐标（ $src_x， $src_y）开始，宽度为 $src_w、高度为 $src_h 的一部分复制到 $dst_im 中坐标为（ $dst_x， $dst_y）的位置上。

【例 8.4-3】 复制图片的一部分到另一张图片中。

程序如下：

```
<? php
    header("Content-type: image/jpeg");                    //发送头信息
     $image1 = imagecreatefromjpeg("monkey.jpg");          //根据已经存在的图片创建图形
     $image2 = imagecreatefromjpeg("monkey2.jpg");
    imagecopy( $image1, $image2,350,50,50,0,120,120);      //复制图片的一部分
    imagejpeg( $image1);                                   //输出图形 $image1
    imagedestroy( $image1);
    imagededtroy( $image2);
? >
```

程序运行结果如图 8-16 所示。

图 8-16　程序运行结果

4. 复制图片并调整大小

使用 imagecopyresized（）函数也可以实现 imagecopy（）函数的功能，并可以对复制的图片大小进行调整。语法格式如下：

```
bool imagecopyresized(resource $dst_image, resource $src_image, int $dst_x, int $dst_y, int $src_x,
int $src_y, int $dst_w, int $dst_h, int $src_w, int $src_h )
```

例如：

```
imagecopyresized( $image1, $image2,50,5,50,0,50,50,160,160);//复制图片的一部分并调整大小
```

以上这句代码将从 $image2 复制的一部分从宽 160、高 160 缩小为宽 50、高 50 之后再复制到 $image1 中。

【例 8.4-4】 使用 imagecopyresized（）复制调整后的图片的一部分到另一张图片中。

程序如下：

```
<? php
```

```
header("Content-type: image/jpeg");                    //发送头信息
$image1=imagecreatefromjpeg("monkey.jpg");             //根据已经存在的图片创建图形
$image2=imagecreatefromjpeg("monkey2.jpg");
//imagecopy($image1, $image2,350,50,50,0,120,120);
imagecopyresized($image1, $image2,350,5,50,0,100,100,120,160); //复制调整后的图片
imagejpeg($image1);                                    //输出图形 $image1
imagedestroy($image1);
imagededtroy($image2);
?>
```

程序运行结果如图 8-17 所示。

图 8-17　程序运行结果

5. 旋转图像

使用 imagerotate（）函数可以将图像旋转给定角度，语法格式如下：

```
resource imagerotate(resource $src_im, float $angle, int $bgd_color [, int $ignore_transparent] )
```

图像的旋转中心为图像的中心，旋转后的图像会按比例缩小以适合目标图像大小，边缘不会被剪去。函数参数说明见表 8-5。

表 8-5　imagerotate（）函数参数说明

参　数	说　明
$src_im	源图像
$angle	图像旋转角度
$bgd_color	旋转后没有覆盖的部分的颜色
$ignore_transparent	可选。值为非 0 时，透明色会被忽略（否则会被保留）

【例 8.4-5】使用 imagerotate（）函数将图像旋转 60°。

程序如下：

```
<? php
$filename='titey.png';                    //指定一张图片
```

```
header('Content-type: image/jpeg');
$image = imagecreatefrompng($filename);        //根据已有图片创建图形
$rotate = imagerotate($image,60,0,0);          //旋转图片
imagejpeg($rotate);
imagedestroy($image);
?>
```

程序运行结果如图 8-18 所示。

图 8-18　程序运行结果

8.5　其他的图形函数

1. 取得图形信息

之前介绍的 imagesx（）和 imagesy（）函数可以获取图形的宽和高，这里介绍的 getimagesize（）函数可以获取指定图形的尺寸、宽度、高度和类型等信息。该函数将这些信息以数组的形式返回，如果图片指定的图形不是有效的文件，则返回 FALSE。例如：

```
<? php
$image = "monkey.jpg";
$message = getimagesize($image);
print_r($message);
?>
```

程序运行结果为：

```
Array ( [0] => 447 [1] => 527 [2] => 2 [3] => width = " 447" height = " 527" [bits] => 8
[channels] => 3 [mime] => image/jpeg )
```

程序说明：

键名为 0 的值表示图像文件的宽度（像素）；

键名为 1 的值表示图像文件的高度（像素）；

键名为 2 的值表示图像类型的标记，其中 1 为 GIF，2 为 JPEG，3 为 PNG，4 为 SWF，5 为 PSD，6 为 BMP 等；

键名为 3 的值是一个字符串，width 表示图像宽度，height 表示图像高度；

键名为 bits 的值表示图像颜色的位数；

键名为 channels 的值为 3 表示图像的 RGB 图形；

键名为 mime 的值表示图像的类型信息

2. 透明处理

imagecolorat（）函数的功能是取得某像素的颜色索引值。该函数的语法如下：

```
int imagecolorat ( resource $image , int $x , int $y )
```

函数返回 $image 所指定的图形中指定位置像素的颜色索引值。

如果 PHP 编译时加上了 GD 库 2.0 或更高的版本并且图像是真彩色图像，则本函数以整数返回该点的 RGB 值。用移位加掩码来取得红、绿、蓝各自成分的值。

【例 8.5-1】 使用 imagecolorat（）获取图像的 RGB 颜色。

程序如下：

```
<? php
$im = ImageCreateFromPng( " titey.png " ) ;
$rgb = ImageColorAt( $im,100,100) ;
$r = ( $rgb>>16) &0xFF;
$g = ( $rgb>>8) &0xFF;
$b = $rgb&0xFF;
echo $r.' _' . $g.' _' . $b;
? >
```

程序运行结果为：

```
253_228_208
```

也可以利用 RGB 值，通过使用 imagecolorexact（）函数读取图像在调色板中的索引值，函数格式如下：

```
int imagecolorexact ( resource $image , int $red , int $green , int $blue )
```

函数返回图像调色板中指定颜色的索引值。如果颜色不在图像的调色板中，返回 -1。此函数唯一的缺点在于，如果图像的调色板中不存在这个颜色，函数就不会返回一个有效的颜色索引值。

有了索引值之后，就可以使用 imagecolortransparent（）将 image 图像中的透明色设定为 color。image 是 imagecreatetruecolor（）返回的图像标识符，color 是 imagecolorallocate（）返回的颜色标识符。imagecolortransparent（）函数语法如下：

```
int imagecolortransparent( resource $image [ ,int $color] )
```

该函数需要两个参数：图像资源 $image 和需要设置为透明效果的颜色的索引值 $color。imagecolortransparent（）将 image 图像中的透明色设定为 color。Image 是

imagecreatetruecolor（）返回的图像标记符，color 是 imagecolorallocate（）返回的颜色标识符。

3. 不透明处理

imagecopymege（）函数的主要功能是图像的不透明处理，它定义了图像的像素透明（完全可以看穿）或不透明的程序。函数格式如下：

bool imagecopymege(resource $dst_im, resource $src_im, int $dst_x, int $dst_y, int $src_x, int $src_y, int $src_w, int $src_h,int $pct)

这个函数和 imagecopy（）函数的作用基本一样，最后一个参数 $pct 为透明度，它是控制所复制图像的透明度，范围从 0~100。

8.6　PHP 的日期和时间函数

在 Web 程序开发时，时间发挥着重要的作用。不仅在数据存储和显示时需要日期和时间的参与，很多功能模块的开发，时间通常都是至关重要的。例如，网页静态化需要判断缓存时间、页面访问消耗的时间需要计算、根据不同的时间段提供不同的业务等。PHP 为我们提供了强大的日期和时间处理功能，通过内置的时间和日期函数库，不仅能够得到 PHP 程序在运行时所在服务器中的日期和时间，还可以对它们进行任意检查和格式化，以及在不同格式之间进行转换等。

8.6.1　UNIX 时间戳

UNIX 时间戳是保存日期和时间的一种紧凑简捷的方法，是大多数 UNIX 系统中保存当前日期和时间的一种方法，也是在大多数计算机语言中表示日期和时间的一种标准格式。以 32 位的整数表示格林威治标准时间，例如，使用整数 11230499325 表示当前时间。UNIX 时间戳是从 1970 年 1 月 1 日零点（UTC/GMT 的午夜）开始起到当前时间所经过的秒数。1970 年 1 月 1 日零点作为所有日期计算的基础，这个日期通常称为 UNIX 纪元。

因为 UNIX 时间戳是一个 32 位的数字格式，所以特别适用于计算机处理，例如计算两个时间点间相差的天数。另外，由于文化和地区的差异，存在不同的时间格式，以及时区的问题。所以 UNIX 时间戳也是根据一个时区进行标准化而设计的一种通用格式，并且这种格式可以很容易地转换为任何格式。

也因为 UNIX 时间戳是用一个 32 位的整数表示的，所以在处理 1902 年以前或 2038 年以后的事件，将会遇到一些问题。另外，在 Windows 下，由于时间不能为负数，如果使用 PHP 中提供的时间函数处理 1970 年之前的日期，就会发生错误。

8.6.2　日期和时间转变为 UNIX 时间戳

在 PHP 中，如果需要将日期和时间转变成 UNIX 时间戳，可以调用 mktime（）函数。该函数的语法如下：

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

该函数中所有参数都是可选的，如果参数为空，默认将当前时间转变成 UNIX 时间戳。这样，和直接调用 time（）函数获取当前的 UNIX 时间戳功能相同。参数也可以从右向左省略，任何省略的参数都会被设置成本地日期和时间的当前值。如果只想转变日期，对具体的时间不在乎，可以将前三个转变时间的参数都设置为 0。mktime（）函数对于日期运算和验证非常有用，它可以自动校正越界的输入。例如：

```
<? php
echo date("Y-m-d",mktime(0,0,0,12,36,2018))."<br>";     /* 日期超过 31 天,计算后输出 2019-01-05 */
echo date("Y-m-d",mktime(0,0,0,14,1,2019))."<br>";      /* 月份超过 12 月,计算后输出 2020-01-01 */
echo date("Y-m-d",mktime(0,0,0,1,1,2020))."<br>";       /* 没有问题的转变,输出 2020-01-01 */
echo date("Y-m-d",mktime(0,0,0,1,1,19))."<br>";         /* 将 19 年转变为 2019-01-01 */
? >
```

如果需要将英文文本的日期时间描述直接解析为 UNIX 时间戳，可以使用 strtotime（）函数，该函数的语法格式如下：

```
int strtotime(string time [, int now])
```

函数 strtotime（）可以用英语的自然语言创建某个时刻的时间戳，接受一个包含美国英语日期格式的字符串并尝试将其解析为 UNIX 时间戳（自 Janury 1 1970 00：00：00 GMT 起的秒数），其值相对于 now 参数给出的时间，如果没有提供此参数则用系统当前时间。该函数执行成功则返回时间戳，否则返回 FALSE。例如：

```
<? php
echo @ date("Y-m-d",strtotime("now"))."<br>";          //输出系统当前时间
echo @ date("Y-m-d",strtotime("8 may 2019"))."<br>";   //输出 2019-05-08
echo @ date("Y-m-d",strtotime("+1 day"))."<br>";       //输出系统当前时间的后一天
echo @ date("Y-m-d",strtotime("last monday"))."<br>";  //输出系统当前时间的星期一
? >
```

下例通过使用 strtotime（）函数编写一个离毕业时间的倒计时程序，来介绍该函数在项目开发中的实际应用。

【例 8.6-1】 strtotime（）函数编写一个离毕业时间的倒计时。

程序如下：

```
<? php
$now=@ strtotime("now");                          //获取系统当前时间
$endtime=@ strtotime("2019-06-30 08:08:08");      //设定毕业时间,转为时间戳
$second= $endtime- $now;                          //获取毕业时间到现在时间的时间戳(秒)
$year=floor( $second/3600/24/365);                //从时间戳中换算出年数
```

```
$temp = $second - $year * 365 * 24 * 3600;          //从时间戳中去掉整年的秒数,剩下月的秒数
$month = floor( $temp/3600/24/30 );                 //算出月数
$temp = $temp-@ $mouth * 30 * 24 * 3600;            //从时间戳中去掉整月的秒数,剩下天的秒数
$day = floor( $temp/3600/24 );                      //算出天数
$temp = $temp - $day * 24 * 3600;                   //从时间戳中去掉整天的秒数,剩下小时的秒数
$hour = floor( $temp/3600 );                        //算出小时数
$temp = $temp - $hour * 3600;                       //从时间戳中去掉整小时的秒数,剩下分的秒数
$minute = floor( $temp/60 );                        //算出分数
$second = $temp - $minute * 60;                     //最后就只有剩余的秒数了
echo "距离毕业还有{ $year}年{ $month}月{ $day}天{ $hour}小时{ $second}分{ $minute}秒。";
? >
```

运行结果如图 8-19 所示。

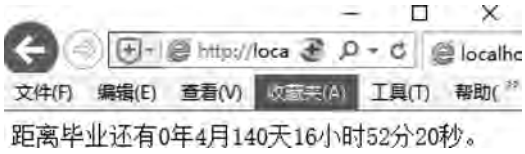


图 8-19　程序运行结果

注意：如果给定的年份是两位数字的格式，则年份值 0~69 表示 2000~2069，70~100 表示 1970~2000。

8.6.3　日期的计算

在 PHP 中，计算两个日期之间相隔的长度，最简单的方法就是通过计算两个 UNIX 时间戳之差来获得。例如，在 PHP 脚本中接收来自 HTML 表单用户提交的出生日期，计算这个用户的年龄。

【例 8.6-2】 从用户提交的出生日期，计算这个用户的年龄。

程序如下：

```
<? PHP
$year = 1999;                                       //从表单中接收的出生年份
$month = 10;                                        //从表单中接收的出生月份
$day = 03;                                          //从表单中接收的出生天数
$birthday = @ mktime(0,0,0, $month, $day, $year);   //将出生日期转变为 UNIX 时间戳
$nowdate = time( );                                 //调用 time( ) 函数获取当前时间
$ageunix = $nowdate - $birthday;                    //通过时间计算年龄
$age = floor( $ageunix/(60 * 60 * 24 * 365) );      //转换时间戳为实际年龄
echo "该用户的年龄为:{ $age}岁。";                   //输出结果
? >
```

程序运行结果如图 8-20 所示。

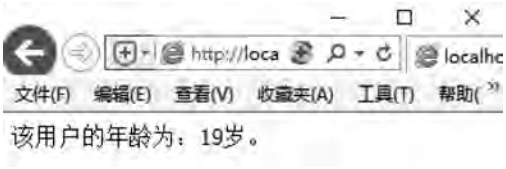


图 8-20　程序运行结果

在以上程序中，调用 mktime（）函数将从用户出生日期转变为 UNIX 时间戳，再调用 time（）函数获取当前时间的 UNIX 时间戳。因为这个日期的格式都是使用整数表示的，所以可以将它们相减。又将计算后获取的 UNIX 时间戳除以一年的秒数，将 UNIX 时间戳转变为以年度量的单位。

8.6.4　获取日期和时间

PHP 提供了多种获取时间和日期的函数，除了通过 time（）函数获取当前的 UNIX 时间戳外，调用 getdate（）函数确定当前时间，通过 gettimeofday（）函数获取某一天中的具体时间。此外，在 PHP 中还可以通过 date_sunrise（）和 date_sunset（）两个函数，获取某地点某天的日出和日落时间。

1. getdate（）函数

使用 getdate（）函数也可以获取日期和时间信息，语法格式如下：

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

说明：$timestamp 是要转化的时间戳，如果不给出则使用当前时间。函数根据 $timestamp返回一个包含日期和时间信息的数组，数组的键名和值见表 8-6。

表 8-6　getdate（）函数返回的数组单元

键　名	说　明	值的例子
hours	小时的数字表示	0~23
mday	月份中第几天的数字表示	1~31
minutes	分钟的数字表示	0~59
mon	月份的数字表示	1~12
month	月份的完整文本表示	January~December
seconds	秒的数字表示	0~59
wday	星期中第几天的数字表示	0（表示星期天）~6（表示星期六）
weekday	星期几的完整文本表示	Sunday~Saturday
yday	一年中第几天的数字表示	0~365
year	4 位数字表示的完整年份	例如：1999 或 2003
0	自 UNIX 纪元开始至今的秒数	系统相关，典型值从–2147483648~2147483647

例如：

```
<? php
 $array1 = getdate( ) ;
 $array2 = getdate( strtotime( ' 2019-05-15' ) ) ;
print_r( $array1) ;
echo " <br>" ;
print_r( $array2) ;
? >
```

运行结果为：

```
Array ( [seconds] => 55 [minutes] => 33 [hours] => 15 [mday] => 13 [wday] => 3 [mon] => 2 [year] => 2019 [yday] => 43 [weekday] => Wednesday [month] => February [0] => 1550072035 )
Array ( [seconds] => 0 [minutes] => 0 [hours] => 0 [mday] => 15 [wday] => 3 [mon] => 5 [year] => 2019 [yday] => 134 [weekday] => Wednesday [month] => May [0] => 1557878400 )
```

2. date ( ) 函数

当日期和时间需要保存到计算机中时，应该使用 UNIX 时间戳作为标准格式，这可以作为一条重要的规则。但 UNIX 时间戳的格式可读性比较差，所以把时间戳格式化为可读性更好的日期和时间，或格式化为其他软件需要的格式。在 PHP 中可以调用 date ( ) 函数格式化一个本地时间和日期，语法格式如下：

```
string date( string  $format [ , int  $timestamp ] )
```

说明：该函数有两个参数，第一个参数 $format 是必需的，它指定了转化后的日期和时间的格式，第二个参数 $timestamp 是可选的，它是一个需要转化的时间戳，如果省略则使用本地当前时间，即默认值为 time ( ) 函数的值。time ( ) 函数返回当前时间的时间戳。date ( ) 函数的常见调用方式如下：

```
echo date( "Y 年 m 月 d 日 H:i:s" ) ; //输出当前的时间格式:2019 年 02 月 13 日 16:00:07
```

date ( ) 函数中的第一个参数，是通过表 8-7 中所提供的特定字符组成的格式化字符串。如果在格式字符串中的字符前加上反斜线来转义，可以避免它按照表 8-7 被解释。如果加上反斜线后的字符本身就是一个特殊序列，那么还要转义反斜线。格式字符串中不能被识别的字符将原样显示。PHP 中所支持的日期格式代码见表 8-7。

表 8-7　PHP 的 date ( ) 函数支持的格式代码

格式化字符	描　述	示　例
d	月份中的第几天，有前导零的 2 位数字	01~31
D	星期中的第几天，用 3 个字母表示	Mon 到 Sun
j	月份中的第几天，没有前导零	1~31
l	星期几，完整的文本格式	Sunday~Saturday
N	ISO-8601 格式数字表示的星期中的第几天	1（星期一）~7（星期天）
S	每月天数后面的英文后缀，用 2 个字符表示	st，nd，rd 或 th，可以和 j 一起用

续表

格式化字符	描　述	示　例
w	星期中的第几天，数字表示	0（星期天）~6（星期六）
z	年份中的第几天	范围在 0~366
W	ISO-8601 格式年份中的第几周，每周从星期一开始	例如：42（当年的第 42 周）
F	月份，完整的文本格式，如 January 或 March	January~December
m	数字表示的月份，有前导零	01~12
M	三个字母缩写表示的月份	Jan~Dec
n	数字表示的月份，没有前导零	1~12
t	给定月份所应有的天数	28~31
L	是否为闰年	如果是闰年为 1，否则为 0
o	ISO-8601 格式年份数字。这和 Y 的值相同，只是如果 ISO 的星期数（W）属于前一年或下一年，则用那一年	例如：1999 或 2003
Y	4 位数字完整表示的年份	例如：1999 或 2003
y	2 位数字表示的年份	例如：99 或 03
a	小写的上午和下午值	am 或 pm
A	大写的上午和下午值	AM 或 PM
B	Swatch Internet 标准时	000~999
g	小时，12 小时格式，没有前导零	1~12
G	小时，24 小时格式，没有前导零	0~23
h	小时，12 小时格式，有前导零	01~12
H	小时，24 小时格式，有前导零	00~23
i	有前导零的分钟数	00~59
s	秒数，有前导零	00~59
e	时区标志	例如：UTC，GMT，Atlantic/Azores
I	是否为夏令时	如果是夏令时为 1，否则为 0
O	与格林尼治时间相差的小时数	例如：+0200
P	与格林尼治时间（GMT）的差别，小时和分钟之间用冒号分隔	例如：+02：00
T	本机所在的时区	例如：EST，MDT
Z	时区偏移量的秒数。UTC 西边的时区偏移量总是负的，UTC 东边的时区偏移量总是正的	-43200~43200
c	ISO 8601 格式的日期	2004-02-12T15：19：21+00：00
r	RFC 822 格式的日期	Thu，21 Dec 2000 16：01：07 +0200
U	从 UNIX 纪元开始至今的秒数	例如：time（）函数

表 8-7 中包括了可用于 date（）函数的所有格式化参数，该函数按照这些参数指定的值生成一个字符串表示。例如：

```
<? php
echo date('jS-F-Y');                          //输出 13th-February-2019
echo date('Y-m-d');                           //输出 2019-02-13
echo date('l M ',strtotime('2019-09-10'));    //输出 Tuesday Sep
echo date("l",mktime(0,0,0,2,14,2019));       //输出 Thursday
echo date('U');                               //输出当前时间的时间戳
? >
```

8.7　综 合 案 例

【例 8.7-1】 输出某个月的日历，要求年份和月份可以进行选择，使用“@”忽略警告信息。

新建 cal. php 文件，输入以下代码：

```
<? php
 $year=@ $_GET['year'];                                  //获得地址栏的年份
 $month=@ $_GET['month'];                                //获得地址栏的月份
if(empty( $year))
      $year=@ date("Y");                                 //初始化为本年度的年份
if(empty( $month))
      $month=@ date("n");                                //初始化为本月的月份
 $day=@ date("j");                                       //获取当天的天数
 $wd_ar=array("日","一","二","三","四","五","六");       //星期数组
 $wd=@ date("w",mktime(0,0,0, $month,1, $year));         //计算当月第一天是星期几
//年链接
 $y_lnk1= $year<=1970? $year=1970: $year-1;              //上一年
 $y_lnk2= $year>=2037? $year=2037: $year+1;              //下一年
//月链接
 $m_lnk1= $month<=1? $month=1: $month-1;                 //上个月
 $m_lnk2= $month>=12? $month=12: $month+1;               //下个月
echo "<table cellpadding=6 cellspacing=0 width=200 bgcolor=#eeeeee><tr align=center bgcolor=#
ccccccc>";
//输出年份,单击“<”链接跳到上一年,单击“>”链接跳到下一年
echo "<td colspan=4><a href='cal.php? year= $y_lnk1&month= $month'>
     <</a>". $year."年<a href='cal.php? year= $y_lnk2&month= $month'>></a></td>";
//输出月份,单击“<”链接跳到上个月,单击“>”链接跳到下个月
echo "<td colspan=3><a href='cal.php? year= $year&month= $m_lnk1'>
     <</a>". $month."月<a href='cal.php? year= $year&month= $m_lnk2'>></a></td> </tr>";
echo "<tr align=center>";
for( $i=0; $i<7; $i++)
```

```
{
    echo "<td> $wd_ar[ $i]</td> ";                    //输出星期数组
}
echo "</tr>";
 $tnum= $wd+@ date("t",@ mktime(0,0,0, $month,1, $year));//计算星期几加上当月的天数
for( $i=0; $i< $tnum; $i++)
{
     $date= $i+1- $wd;                                //计算日数在表格中的位置
    if( $i%7==0) echo "<tr align=center>";            //一行的开始
    echo "<td>";
    if( $i>= $wd)
    {
        if( $date== $day&& $month==@ date("n"))       //如果是当月的当天则将天数加黑
            echo "<b>". $day."</b>";
        else
            echo $date;                               //输出日数
    }
    echo "</td> ";
    if( $i%7==6) echo "</tr> ";                       //一行结束
}
echo "</table>";
? >
```

程序运行结果如图 8-21 所示。

图 8-21　程序运行结果

说明：代码中在使用超链接时，超链接的地址中包含了年份和月份信息，例如，代码中的一句：

```
<a href=' cal.php? year= $year&month= $m_lnk2' >></a>
```

这句中的 URL 形式为“url？参数 1=值 1& 参数 2=值 2”，当单击超链接时，将跳转到该 URL 页面，在该页面使用“ $_GET［ ‘参数’ ］”就可以获得 URL 中参数的值。

本章小结

本章详细介绍了 PHP 中图形图像处理方法和步骤，重点介绍了绘制图形的相关函数，同时也学习了图像处理技术的相关函数，如图像的填充等，最后介绍了 PHP 的日期和时间的处理函数。

习　题

一、简答题

1. 取得当前安装的 GD 库的信息的函数是什么？
2. 取得图像大小的函数是什么？
3. 为一幅图像分配颜色 + alpha 的函数是什么？
4. 新建一个基于调色板的图像的函数是什么？
5. 新建一个黑色图像的函数是什么？
6. 用给定角度旋转图像的函数是什么？
7. 水平地画一行字符串的函数是什么？
8. 载入一新字体的函数是什么？
9. 取得某像素的颜色索引值的函数是什么？

二、编程题

1. 编写程序，在页面上绘制一个三角形，三个点的坐标依次为（20，20）（130，130）和（60，180）。
2. 编写程序，在页面上绘制一个圆，并用红色（200，2，2）填充。
3. 编写程序，在页面上绘制文字，文字内容为“欢迎你登录”。
4. 编写程序，画出下列图像：

第 9 章

PHP 面向对象程序设计

【本章内容简介】

本章通过讲述面向对象的基本概念以及相关的案例实践，让大家对面向对象有一个基本的认识，能够掌握把实际问题抽象成为类对象而加以解决的方法，掌握面向对象的最重要的核心能力。

【本章学习要求】

通过本章的学习，要求掌握面向对象的基本概念，理解类和对象，掌握 PHP 面向对象的开发。

9.1 面向对象的介绍

9.1.1 类和对象的发展（从面向过程到面向对象）

最初的计算机语言只有基本变量（类似基本数据类型），用来保存数据。但是数据多了怎么办？成千上万的数据怎么处理？于是有了数组的概念，通过数组将这同类型的数据进行组织。后来，数据不仅多且复杂了，而且对数据的操作（指的就是函数）也频繁了，但结构体只包含了数据，没有包含对数据操作的方法，于是类和对象产生了。也就是说，面向过程的数据和方法是分离的，而面向对象的数据和方法集成到了一起，便于扩展。

对于描述复杂的事物，为了从宏观上把握、从整体上合理分析，需要使用面向对象的思路来分析整个系统。但是，具体到微观操作，可能需要面向过程的思路去处理。如开车（简单事物），可以用一系列线性过程来描述；但是造车（复杂事物），就无法使用简单的线性思维来解决了。这时可以使用面向对象的思想，将造车看作车壳、发动机、车轮等对象的组合，从宏观上把握造车过程，而不是拘泥于从哪个小零件开始造起。

面向对象和面向过程的总结：

（1）都是解决问题的思维方式，都是代码组织的方式。

（2）解决简单问题可以使用面向过程。

（3）解决复杂问题，宏观上使用面向对象把握，微观处理上仍然是面向过程。

面向对象的思考方式：遇到复杂问题，先从问题中找名词，然后确立这些名词哪些可以作为类，再根据问题需求确定类的属性和方法，以此来理清类之间的关系。

9.1.2　面向对象的基本特征

（1）封装（Encapsulation）。在面向对象设计方法中，封装是指将数据及对数据的操作方法包装或隐藏起来的方法。封装只需知道如何使用，而不需要了解其内部实现细节。例如，对于电视机，使用者只需操作开关键、频道键和声音键等即可，既无需了解电视机的内部工作原理，也不需去操控电视机的内部元件。这不仅给人们带来了使用上的方便，也避免了安全事故发生（如触电）。适当的封装可以让程序更容易理解与维护，也加强了程序的安全性。要访问该封装体的代码和数据，必须通过严格的接口控制。电视机外的那些操作键，可以看作程序片段的“接口”。通俗地说，封装就是需要让用户知道的暴露出来，不需要让用户了解的全部隐藏起来，即“该露的露，该藏的藏”。

（2）继承（extend）。继承的本质是对某一批类的抽象，从而实现对现实世界更好地建模。

继承的另一个好处是：代码重用。新定义的派生类实例可以继承已有基类的属性和行为，而且可以加入新的特性或者是修改已有的特性建立起类的新层次。

（3）多态（polymorphism）。直到系统运行时才根据实际情况决定实现何种操作。

9.2　创建类、属性和方法

一个类代表一类事物，是很多具体事物的总称，类是一个抽象的概念。属性体现一类事物具有什么特征，方法体现这类事物具有什么行为。

9.2.1　对象

首先，需要考虑一个问题：为什么称为面向对象编程，而不是面向类编程？计算机语言越到高级越接近人的思维方式，这是语言发展的一个基本规律。对象是描述客观事物的一个实体，几乎所有东西都可能看作对象。万物皆对象，对象可以是有形的，如坦克、学生、文件；也可以是无形的，如一个方案、头脑中的一个想法。我们人认识世界，其实就是面向对象的。

程序中的对象来源于生活，现实生活中的某个实体经过抽象、建模后就变成程序中的一个对象，即用特定符号描述的对象。对象都有自己的特性，我们叫属性（静态特征）和方法（动态特征）。如学生，其属性有学号、姓名、性别等，其方法有学习、选课等。

由于对象和现实世界的实体联系紧密，所以面向对象编程更加贴近人们的思维和习惯。

9.2.2 类

我们常常说，物以类聚，人以群分。现实生活中的一组相似对象（它们具有共同的特征）可以归为一类。例如，哺乳动物是一个类，它们的共性为：有脊椎、胎生、用肺呼吸、恒温等，一只具体的猫则是一个对象。生物学中的分类法（如林奈氏分类法），就是典型的对生物对象进行归纳成类的方法。程序设计中，把一组相似对象的共同特征抽象出来并存储在一起，就形成了类。每一只哺乳动物都具有哺乳动物类所规定的属性和方法，但是其属性值可能不同，如人的恒温值和猫的恒温值就不一样。

从具体对象到类实际上是一个概括的过程，把对象的共同特征抽取出来形成类，这个过程称为抽象。通俗的解释，抽象即“抽出像的部分”。抽象包括数据抽象和方法抽象，形成属性和方法两方面的特征。

类也可以看作一个模板，或者蓝图，系统根据类的定义来造出对象。我们如何造一辆汽车？类就是汽车的蓝图（图纸），规定了汽车的详细信息，根据图纸可以将汽车造出来。一个模板可以生产很多产品，一张图纸也可以造出很多汽车。

总之，对象是具体事物，我们称 object，或者称 instance（实例）；类是对对象的抽象，我们称 class。

对象和类的关系：特殊到一般，具体到抽象。

类可以看成一类对象的模板，对象可以看成该类的一个具体实例。

类是用于描述同一类型的对象的一个抽象的概念，类中定义了这一类对象所应具有的静态和动态属性。

9.2.3 类的定义和创建

PHP 定义类通常的语法格式如下：

```
<? php
class phpClass {
    var $var1;
    var $var2 = "constant string";
    function myfunc ( $arg1, $arg2) {
        [..]
    }
    [..]
}
? >
```

解析如下：

类使用 class 关键字后加上类名定义。

类名后的一对大括号（ {} ）内可以定义变量和方法。

类的变量使用 var 来声明，变量也可以初始化。

函数定义类似 PHP 函数的定义，但函数只能通过该类及其实例化的对象访问。

我们看一下下面的例子：

【例 9.2-1】

程序如下：

```
<? php
class Site {
    /* 成员变量 */
    var $url;
    var $title;
    /* 成员函数 */
    function setUrl( $par){
        $this->url = $par;
    }
    function getUrl( ){
        echo $this->url . PHP_EOL;
    }
    function setTitle( $par){
        $this->title = $par;
    }
    function getTitle( ){
        echo $this->title . PHP_EOL;
    }
}
? >
```

注：变量 $this 代表自身的对象，PHP_EOL 为换行符。

9.2.4　成员属性

成员变量是类的一个组成部分，一般是基本数据类型（整数、字符串），也可是复合类型（类、数组）。它和普通变量的区别在于，它是在类的内部！

它的定义方式为：

```
class 类名
{
    成员属性(变量);
}
```

比如上节类中的定义 var $var1;就是成员属性。当然,我们需要对属性作出限定,比如访问权限,这些内容在后面作出解释。

9.2.5　成员方法

一个类里面,除了有成员属性外还有行为,这些行为,我们用一个名称代表,这个名称就

是成员方法。比如人可以跑步,可以跳舞,可以唱歌,可以吃饭等等。和成员属性一样,成员方法和普通方法的区别在于,它是在类的内部定义的方法。例如:

```
class Preson{
    public  $name;
    public  $age;
    public  $gender;
    public function Run(){      //声明成员方法
                echo "人在塔在";
            }
}
```

9.3 实例化对象与访问

一个类,只有被实例化后才真实地存在,才占有内存空间。在 PHP 中,使用->符号进行属性的访问。

9.3.1 实例化对象

上节介绍了在 PHP 中怎么样创建一个类,但是我们也知道,类是代表多个具有共同特征的事物的统称,而我们想要得到的是某个对象,也就是某一个具体事物。那么我们怎么得到呢? 可以使用 new 运算符来实例化该类的对象:

```
$runoob = new Site;
$taobao = new Site;
$google = new Site;
```

以上代码我们创建了三个对象,三个对象各自都是独立的,接下来看看如何访问成员方法与成员变量。

9.3.2 对象的属性访问

在实例化对象后,可以使用该对象调用成员方法,该对象的成员方法只能操作该对象的成员变量,例如。

```
// 调用成员函数,设置标题和 URL
$runoob->setTitle( "php 学习");
$taobao->setTitle( "淘宝");
$google->setTitle( "Google 搜索");
$runoob->setUrl( 'www.runoob.com' );
$taobao->setUrl( 'www.taobao.com' );
$google->setUrl( 'www.google.com' );
// 调用成员函数,获取标题和 URL
```

```
$runoob->getTitle();
$taobao->getTitle();
$google->getTitle();
$runoob->getUrl();
$taobao->getUrl();
$google->getUrl();
```

把上面的代码连接起来为：

```
<? php
class Site {
    /* 成员变量 */
    var $url;
    var $title;
    /* 成员函数 */
    function setUrl( $par){
        $this->url = $par;
    }
    function getUrl(){
        echo $this->url . PHP_EOL;
    }
    function setTitle( $par){
        $this->title = $par;
    }
    function getTitle(){
        echo $this->title . PHP_EOL;
    }
}
$runoob = new Site;
$taobao = new Site;
$google = new Site;
// 调用成员函数,设置标题和 URL
$runoob->setTitle( "php 学习");
$taobao->setTitle( "淘宝");
$google->setTitle( "Google 搜索");
$runoob->setUrl( 'www.runoob.com' );
$taobao->setUrl( 'www.taobao.com' );
$google->setUrl( 'www.google.com' );
// 调用成员函数,获取标题和 URL
$runoob->getTitle();
$taobao->getTitle();
$google->getTitle();
$runoob->getUrl();
```

```
$taobao->getUrl();
$google->getUrl();
? >
```

执行以上代码，输出结果为：

```
php 学习
淘宝
Google 搜索
www. runoob. com
www. taobao. com
www. google. com
```

9.4 类的访问控制

PHP 对属性或方法的访问控制，是通过在前面添加关键字 public（公有）、protected（受保护）或 private（私有）来实现的。

public（公有）：公有的类成员可以在任何地方被访问。

protected（受保护）：受保护的类成员可以被其自身以及其子类和父类访问。

private（私有）：私有的类成员只能被其定义所在的类访问。

9.4.1 属性的访问控制

类属性必须定义为公有、受保护、私有之一。如果用 var 定义，则被视为公有。

看下面的例子：

【例 9.4-1】

程序如下：

```
<? php
/ * *
      * Define MyClass
      */
class MyClass
{
      public  $public = 'Public';
      protected  $protected = 'Protected';
      private  $private = 'Private';
function printHello()
      {
            echo  $this->public;
            echo  $this->protected;
            echo  $this->private;
```

```
    }
}
$obj = new MyClass();
echo $obj->public;          // 这行能被正常执行
echo $obj->protected;       // 这行会产生一个致命错误
echo $obj->private;         // 这行也会产生一个致命错误
$obj->printHello();         // 输出 Public、Protected 和 Private
? >
```

9.4.2　方法的访问控制

类中的方法可以被定义为公有、私有或受保护。如果没有设置这些关键字，则该方法默认为公有。

看下面的例子：

【例 9.4-2】

程序如下：

```
<? php
/ * *
     * Define MyClass
     */
class MyClass
{
    // 声明一个公有的构造函数
    public function __construct() { }
    // 声明一个公有的方法
    public function MyPublic() { }
    // 声明一个受保护的方法
    protected function MyProtected() { }
    // 声明一个私有的方法
    private function MyPrivate() { }
    // 此方法为公有
    function Foo()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate();
    }
}
$myclass = new MyClass;
$myclass->MyPublic();               // 这行能被正常执行
$myclass->MyProtected();            // 这行会产生一个致命错误
```

```
$myclass->MyPrivate();          // 这行会产生一个致命错误
$myclass->Foo();                // 公有、受保护、私有都可以执行
? >
```

9.5 类的静态属性和方法

在类里面，有一种特殊的属性和方法，它们用 static 修饰，称为静态属性和静态方法。关于它们的调用（能不能调用，怎么样调用），弄明白它们在内存中的存放位置，就非常容易理解了。静态属性、方法（包括静态与非静态）在内存中，只有一个位置（而非静态属性，有多少实例化对象，就有多少个属性）。

看下面的例子：

【例 9.5-1】

程序如下：

```
<? php
class Human{
    static public  $name = "小妹";
    public  $height = 180;
    static public function tell(){
        echo self:: $name;              //静态方法调用静态属性,使用 self 关键词
        //echo  $this->height;          /* 错。静态方法不能调用非静态属性,因为 $this 代
                                          表实例化对象,而这里是类,不知道 $this 代表哪个
                                          对象 */
    }
    public function say(){
        echo self:: $name . "我说话了"; //普通方法调用静态属性,同样使用 self 关键词
        echo  $this->height;
    }
}
$p1 = new Human();
$p1->say();
$p1->tell();                            //对象可以访问静态方法
echo  $p1:: $name;                      /* 对象访问静态属性。不能这么访问 $p1->name,因
                                          为静态属性的内存位置不在对象里 */
Human::say();                           //错。say() 方法有 $this 时出错,没有 $this 时能出
                                          结果
? >
```

结论：

(1) 静态属性不需要实例化即可调用。因为静态属性存放的位置是在类里，调用方法

为“类名::属性名”。

（2）静态方法不需要实例化即可调用。

（3）静态方法不能调用非静态属性。因为非静态属性需要实例化后，存放在对象里。

（4）静态方法可以调用非静态方法，使用 self 关键词。PHP 里，一个方法被 self:: 后，它就自动转变为静态方法。

9.6　构造函数与析构函数

PHP 语言为类提供的构造函数可自动完成对象的初始化任务，并且提供了与构造函数相对应的析构函数在对象撤销时执行一些收尾任务。

9.6.1　构造函数

构造函数是一种特殊的方法，主要用来在创建对象时初始化对象，即为对象成员变量赋初始值，在创建对象的语句中与 new 运算符一起使用。

PHP 允许开发者在一个类中定义一个方法作为构造函数，语法格式如下：

```
void __construct ( [ mixed $args [ , $... ] ] )
```

在上面的例子中就可以通过构造方法来初始化 $url 和 $title 变量：

```
function __construct( $par1, $par2 ) {
    $this->url = $par1;
    $this->title = $par2;
}
```

有了构造函数，现在就不需要再调用 setTitle 和 setUrl 方法了，可以直接使用构造函数，看下面例子：

【例 9.6-1】

```
$runoob = new Site('www.runoob.com', 'php 学习');
$taobao = new Site('www.taobao.com', '淘宝');
$google = new Site('www.google.com', 'Google 搜索');
// 调用成员函数,获取标题和 URL
$runoob->getTitle();
$taobao->getTitle();
$google->getTitle();
$runoob->getUrl();
$taobao->getUrl();
$google->getUrl();
```

9.6.2　析构函数

析构函数（destructor）与构造函数相反，当对象结束其生命周期时（例如对象所在的

函数已调用完毕），系统自动执行析构函数。

PHP 5 就开始引入了析构函数的概念，这类似于其他面向对象的语言，其语法格式如下：

```
void __destruct ( void )
```

看下面的例子：

【例 9.6-2】

```
<? php
class MyDestructableClass {
    function __construct( ) {
        print "构造函数\n";
     $this->name = "MyDestructableClass";
    }
    function __destruct( ) {
        print "销毁 ". $this->name . "\n";
    }
}
 $obj = new MyDestructableClass( );
? >
```

执行以上代码，输出结果为：

```
构造函数
销毁 MyDestructableClass
```

9.7 类的封装与继承

类的封装与继承是面向对象程序设计的两大基本特征。封装是把现实生活中的一类具有共同特征的事物保护起来，使得部分属性可以访问，部分不可以。继承是为了使代码可重复利用，节省时间、空间。

9.7.1 封装

封装是指将现实世界中存在的某个客体的属性与行为绑定在一起，并放置在一个逻辑单元内。其实前面使用过的类，都是使用了封装的思想，这里不再赘述。

9.7.2 继承

可以使一个类继承并拥有另一个已经存在类的成员属性和方法，被继承的类称为父类或基类，继承类称为子类。通过 extends 关键字实现继承关系。

产生父类或者基类的语法：

```
class father{
```

```
}
产生子类的语法：
class son extends father{
}
```

看下面这个例子：

【例 9.7-1】

```
class father{
    protected  $name;
    function __construct( $name){                    //构造函数
        $this->name= $name;
    }
    function work(){
        echo "{ $this->name}我在工作;
    }
    function __destruct(){}                          //析构函数
}
class son extends father{                            //继承父类
    function play(){
        echo "{ $this->name}我在玩游戏;
    }
}
$my_father=new father("爸爸");                       //创建父类对象
$my_father->work();
$my_son=new son("儿子");
$my_son->work();
$my_son->play();
```

解析：在父类 father 里，定义了一般的属性和方法，然后定义子类。你可能发现在子类里面是没有构造函数和析构函数的，因为子类是继承了父类的所有方法，所以可以调用 $my_son->work（），这就是 PHP 类的继承了。另外要注意：PHP 不能多层继承，如：class A extends B extends C，这样的继承在 PHP 中是无效的，PHP 里只有单继承，不能多继承，需要其他办法变相“实现”多继承。

9.8　类的魔术方法

PHP 提供了一系列用“__”开头的方法，这些方法无须自己手动调用，会在合适的时机自动调用，这类方法称为魔术方法。

例如：function __construct（）{}，在新建一个对象时会自动调用此方法。

在 PHP 中提供了以下魔术方法：

（1）__construct（）：构造函数，新建对象时自动调用。例如：

```
class Person{
    public $name;
    public $age;
    function __construct( $name, $age){
        $this->name= $name;
        $this->age= $age;
    }
}
```

在新建 Person 对象时会自动调用__construct（），将传入的参数分别赋给 $name，$age。

（2）__destruct（）：析构函数，当一个对象被销毁前自动调用。例如：

```
class Person{
    public $name;
    public $age;
    function __destruct(){
        echo "这个对象被销毁了";
    }
}
```

同上所述，对象被销毁时执行此魔术方法。

（3）__get（）：访问类中私有属性时自动调用，传递读取属性名，返回 $this->属性名。例如：

```
class Person{
    private $name;
    public $age;
    function __get( $name){
        return $this-> $name;
    }
}
```

如上代码，$name 被设为私有属性后不能直接通过“$this->name;”的方式读取，因此需要设置__get（）魔术方法来读取。

（4）__set（）：给类的私有属性赋值时自动调用，传递需设置的属性名、属性值。例如：

```
class Person{
    private $name;
    public $age;
    function __set( $key, $value){
        $this-> $key= $value;
    }
}
```

同__get（）方法，私有属性不能通过“ $this->name="aaa";”的方式直接设置，需要设置__set（）方法，传入的参数分别是属性名和属性值。

（5）__isset（）：使用 isset 检测对象私有属性时调用，传递检测的属性名，返回 isset（ $this->属性名）。例如：

```
class Person{
    private  $name;
    public  $age;
    function __isset( $name){
        return isset( $this-> $name);
    }
}
```

使用 isset 检测对象私有属性时调用，其他同上。

（6）__unset（）：使用 unset（）函数删除对象私有属性时调用，传递删除的属性名，方法中执行 unset（ $this->属性名）。例如：

```
class Person{
    private  $name;
    public  $age;
    function __unset( $name){
        unset( $this-> $name);
    }
}
```

同上，在用 unset（）函数删除私有属性时调用。

（7）__toString（）：使用 echo 打印对象时调用，返回打印对象时想要显示的内容，返回必须是字符串。例如：

```
class Person{
    public  $name;
    public  $age;
    function __toString(){
        return  $name;
    }
}
```

使用 echo 打印对象时调用，返回打印对象时想要显示的内容。

（8）__call（）：调用一个类中未定义的或未公开的方法时自动调用，传递被调用的函数名及参数列表。例如：

```
class Person{
    public  $name;
    public  $age;
    function __call( $funcName, $funcParams){
        echo "您调用的函数{ $funcName},参数列表";
        print_r( $funcParams);
```

```
            echo "不存在/未公开";
        }
    }
    $zhangsan=new Person();
    $zhangsan->say(1,2,3);
```

如代码所示，调用不存在的“say（1，2，3);”时，会打印“您调用的函数 way，参数列表 Array（［0］=> 123 ）不存在”。

（9）__clone（）：当使用 clone 关键字克隆一个对象时自动调用，作用是为新克隆的对象初始化赋值。例如：

```
class Person{
    public $name;
    public $age;
    function __clone(){
        $this->name="lisi";
    }
}
$zhangsan=new Person("zhangsan",24);
$lisi=clone $zhangsan;
echo $lisi->name;
```

打印出的结果是“lisi”，而不是“zhangsan”。

（10）__sleep（）：对象序列化时自动调用，返回一个数组，数组中的值就是可以序列化的属性。例如：

```
class Person{
    public $name;
    public $age;
    function __sleep(){
        return array("name","age");
    }
}
```

如上述代码，在用 serialize（）序列化对象时只有“name”，“age”属性会被序列化。

（11）__wakeup（）：对象反序列化时自动调用，为反序列化新产生的对象进行初始化赋值。例如：

```
class Person{
    public $name;
    public $age;
    function __wakeup(){
        $this -> name = "lisi";
    }
}
```

上述代码将对象序列化后再反序列化，同时给对象的 name 属性赋初值为“lisi”。

（12）__autolode（）：需要在类外部声明函数，当实例化一个未声明的类时自动调用，传递实例化的类名，可以使用类名自动加载对应的类文件。例如：

```
function __autoload( $className){
    include "class/".strtolower( $className)."class.php";
}
$zhangsan=new Person();
$zhangsan->say();
```

如上述代码，Person 类不存在时调用__autolode（）方法，载入写在文件外的类，“"class/". strtolower（ $className）. ". class. php";”就是类文件的相对路径，$className 就是不在本文件中的类的类名，因此在创建外部类文件时命名方式必须是小写的类名. class. php。__autolode（）方法也是唯一一个不用写在类中的魔术方法。

以上就是 PHP 中的魔术方法，合理利用这些方法可以轻松实现对象的各种操作。

9.9　综 合 案 例

上述我们介绍了对 PHP 面向对象编程，本节看一个综合案例。

【例 9.9-1】

程序如下：

```
<? php
date_default_timezone_set("PRC");
/*
 * 1.静态属性用于保存类的公有数据
 * 2.静态方法里面只能访问静态属性
 * 3.静态成员不需要实例化对象就可以进行访问
 * 4.类的内部可以通过 self 或者 static 关键字访问自身静态成员
 * 5.可以通过 parent 关键字访问父类的静态成员
 * 6.可以通过类的名称在类定义外部访问静态成员
 *
 */
class Human{
    public  $name;
    protected  $height; //只有自身和子类可以访问
    public  $weight;
    private  $isHungry = true; //不能被子类访问
    public static  $sValue = "Static value in Human class"."\n";
    public function eat( $food){
        echo  $this->name."'s eating "."' $food"."\n";
    }
```

```
        public function info( ) {
            echo "HUMAN :". $this->name. " ;". $this->height." ;". $this->isHungry." \n" ;
        }
    }
    class Animal{
    }
    /* 类的定义以关键字 class 开始,后面是类的名称,类的命名通常第一个字母大写,以中括号开始和结束 */
    //在 PHP 中用 extends 关键字表示类的继承,后面跟父类的类名
    //PHP 中 extends 后只能跟一个类的类名,这是 PHP 中的单继承原则
    class NBaplayer extends Human
    {
        // public  $name = "Jordan" ; //定义属性
        // public  $height = "198cm" ;
        // public  $weight = "98kg" ;
        public  $team = "Bull" ;
        public  $playernumber = "23" ;
        private  $age = "40" ; //Private 的类成员只能在内部被访问
        //静态属性在定义时在访问控制关键字后面添加 static 关键字即可
        public static  $president = "David Stern" ;
        // 静态方法在定义时在访问控制关键字后面添加 static 关键字即可
        public static function changePresident( $newpresdt) {
            //在类定义中使用静态成员时,用 self 或者 static 关键字后面跟::操作符即可
            //注意,在访问静态成员属性时,::后面需要跟 $符号
            self :: $president =  $newpresdt;
            //使用 parent 关键字访问父类中的静态成员
            echo parent:: $sValue." \n" ;
        }
        //构造函数,在对象被实例化时自动调用
        function __construct( $name, $height, $weight, $team, $playernumber)
        {
            echo "in NBaplayer constuctor\n" ;
             $this->name =  $name; /* $this 是 php 里面的伪变量,表示对象自身,可以通过 $->this 的方式访问对象的属性和方法 */
             $this->height =  $height;
             $this->weight =  $weight;
             $this->team =  $team;
             $this->playernumber =  $playernumber;
               echo  $this->height." \n" ;
        }
        //析构函数,在程序执行结束时自动调用
```

```
        /*析构函数通常被用于清理程序使用的资源。比如,程序使用了打印机,那么可以在析构函数里释放打印机资源*/
        function __destruct()
        {
            echo "Destroying ". $this->name."\n";
        }
        //定义方法
        public function run()
        {
            echo "running\n";
        }
        public function jump()
        {
            echo "jumping\n";
        }
        public function dribble()
        {
            echo "dribbling\n";
        }
        public function shoot()
        {
            echo "shooting\n";
        }
        public function dunk()
        {
            echo "dunking\n";
        }
        public function pass()
        {
            echo "passing\n";
        }
        public function getAge(){
            echo  $this->name."'s age is ".( $this->age - 2)."\n";
        }
    }
    //类到对象的实例化
    //类的对象为实例化时使用关键字 new,后面是类的名称和一堆括号
    //  $jordan = new NBaplayer("Jordan","198cm","98kg","Bull","23");
    //  $james = new NBaplayer("James","203cm","120kg","Heat","6");
    //对象中的成员属性通过->符号来访问
    //在类定义外部访问静态属性,可以用类名加::操作符的方法来访问类的静态成员
```

```
echo NBaplayer::$president."Before change"."\n";
NBaplayer::changePresident("Aadam Siver");
echo NBaplayer::$president."\n";
echo Human::$sValue."\n";
// echo "Jordan : ".$jordan->president."\n";
// echo "James : ".$james->president."\n";
echo $jordan->name."\n";
//echo $jordan->getAge();
// $jordan->info();
// $jordan->eat("Apple"); //在子类中的对象上可以直接访问父类中定义的属性和方法
//对象中的成员方法通过->符号来访问
// $jordan->dribble();
// $jordan->dunk();
// $jordan->jump();
// $jordan->pass();
// $jordan->run();
// $jordan->shoot();
// //每一次用 new 实例化对象的时候,都会用类名后面的参数列表调用构造函数
// $james = new NBaplayer("James","203cm","120kg","Heat","6");
// echo $james->name."\n";
// //通过把变量设置为 Null,可以触发析构函数的调用
// //当对象不会再被使用的时候,会触发析构函数
// $james1 = $james;
// $james2 = & $james;
// $james2 = null;
// $james1 = null;
// echo "From now on James will not be used.\n";
? >
```

本章小结

本章主要学习了面向对象程序设计的相关知识，介绍了封装、继承、多态的含义和特点，通过创建类、属性和方法来加深对面向对象的学习和理解。我们已经知道了在 PHP 中，怎么样创建一个类了，但是我们也知道，类是代表所多具有共同特征的事物的统称，而我们想要得到的是某个对象，也就是某一个具体事物，这就是我们常说的实例化对象。在实例化对象后，我们可以使用该对象调用成员方法，该对象的成员方法只能操作该对象的成员变量。

PHP 对属性或方法的访问控制，是通过在前面添加关键字 public（公有），protected（受保护）或 private（私有）来实现的。

public（公有）：公有的类成员可以在任何地方被访问。

protected（受保护）：受保护的类成员则可以被其自身以及其子类和父类访问。

private（私有）：私有的类成员则只能被其定义所在的类访问。

在类里面，有一种特殊的属性和方法，它就是用 static 修饰词的属性和方法，我们称它为静态属性和静态方法。需要注意的是：

（1）静态属性和方法不需要实例化即可调用。

（2）静态方法不能调用非静态属性。这是因为非静态属性需要实例化后，存放在对象里。

（3）静态方法可以调用非静态方法，使用 self 关键词。

另外，我们还学习了构造函数与析构函数的使用，PHP 允许开发者在一个类中定义一个方法作为构造函数，析构函数（destructor）与构造函数相反，当对象结束其生命周期时（例如对象所在的函数已调用完毕），系统自动执行析构函数。

构造函数语法格式如下：

void __ construct（［ mixed $args［，$...］］）

析构函数语法格式如下：

void __ destruct（ void ）

最后，我们还对类的封装和继承进行了学习，主要介绍了继承在实际中的应用，同时还对类的魔术方法进行了讲解。

（1）封装：是指将现实世界中存在的某个客体的属性与行为绑定在一起，并放置在一个逻辑单元内。

目的：保护类。

做法：成员私有，方法访问。

__ get（）和__ set（）构造方法特殊点：①写法特殊。②执行时间特殊。③可以实现成员的初始化。

$this 关键字：$this 代表该对象。

（2）继承：子类可以继承父类的一切。

特点：单继承，一个子类只能有一个父类，一个父类可以派生多个子类。

方法重写：子类里建一个和父类重名的方法，会将父类方法覆盖；如果想扩展父类的方法而不是覆盖，可使用 parent:: Say（）。

（3）多态：当父类引用指向子类实例的时候，由于子类对父类的方法进行了重写，所以父类引用在调用该方法的时候表现出的不同就称为多态。

条件：①必须继承。②子类必须对父类方法重写。③父类引用指向子类实例。

（4）静态。

关键字：static。

普通成员是属于对象的，静态成员是属于类的，普通方法里面可以调用静态成员，但静态方法里面不能调用普通成员。用类名调用：Ren:: Say（）;。

self 关键字：代表该类本身。

析构函数：写法特殊，执行特殊，在对象销毁的时候执行__ tostring（）方法。

习　题

一、填空题

1. 继承的关键字为_____。
2. PHP 中有很多以两个下画线开头的方法，称为_____。
3. 在魔术方法中，__construct（）是构造方法，__destruct（）是_____方法。

二、简答题

1. 什么是面向对象？
2. 简述 private，protected，public 修饰符的访问权限。
3. 什么是构造函数？什么是析构函数？它们的作用是什么？

第 10 章

PHP 与 Web 页面交互

【本章内容简介】

Web 应用程序的基本功能就是与用户进行交互，获取并处理用户提交的数据。用户提交数据的方法有：①通过表单提交，如用户注册、用户登录、留言等都是通过表单提交信息的；②使用网址中的 URL 参数发送数据给服务器。这些数据都是以 HTTP 请求的方式发送的。Web 服务器必须能够获取用户通过浏览器发送来的数据，才能与用户进行交互。由于 HTTP Web 协议是无状态协议，因此对于事务处理没有记忆能力，缺少状态意味着如果后续处理需要前面的信息，就必须重传，这样可能导致每次连接传送的数据量增大。于是，两种用于保持 HTTP 连接状态的技术就应运而生了，一种是 Cookie，另一种是 Session。其中，Cookie 将数据存储在客户端，并显示永久的数据存储。Session 将数据存储在服务器端，保证数据在程序的单词访问中持续有效。

【本章学习要求】

通过本章的学习，要求了解表单及表单元素，熟悉在 Web 页面中插入表单的过程，掌握 PHP 程序如何接收客户端通过 POST 和 GET 方式传递过来的参数值，掌握如何创建、读取和删除 Cookie 和掌握启动、注册、使用、删除 Session 的方法等。

10.1 PHP 与表单

Web 表单的功能是让浏览者和网站有一个互动的平台。Web 表单主要用来在网页中发送数据到服务器，例如，提交注册信息时需要使用表单。用户填写信息后进行提交操作，便是将表单的内容从客户端浏览器传送到服务器端，经过服务器上的 PHP 程序处理后，再将用户所需要的信息传递回客户端浏览器。通过获得用户信息，使 PHP 与 Web 表单实现了交互。

10.1.1 <form >标记及其属性

<form>标记用来创建一个表单，即定义表单的开始和结束位置。这一标记有几方面的作用：首先，限定表单的范围，一个表单中的所有表单域标记，都要写在<form>与

</form>之间，单击“提交”按钮时，提交的也是该表单范围内的内容。其次，携带表单的相关信息，例如处理表单的脚本程序位置（action）、提交表单的方法（method）等。这些信息对于浏览者是不可见的，但对于处理表单却起着决定性的作用。

使用<form>标记，并在其中插入相关的表单元素，即可创建一个表单。表单结构如下：

```
<form name="form_name" method="method" action="url" enctype="value" target="target_win">
……
</form>
```

<form>标记的属性见表 10-1。

表 10-1　<form>标记的属性

属　性	说　明
name	表单名称
method	设置表单的提交方式：GET 和 POST 方式
action	指向处理表单页面的 URL（相对位置或绝对位置）
enctype	设置表单内容的编码方式
target	设置返回信息的显示方式，一共有 4 种取值：_blank 将返回信息显示在新的窗口中；_parent 将返回信息显示在父级窗口中；_self 将返回信息显示在当前窗口中；_top 将返回信息显示在顶级窗口中。

【说明】

GET 方式是将表单的内容附加在 URL 地址后面发送；POST 方式是将表单中的信息作为数据块发送给服务器上的处理程序，在浏览器的地址栏中不显示提交的信息。method 属性默认为 GET 方式。

10.1.2　表单元素

表单由表元素组成。常用的表单元素有以下几种标记：输入域标记<input>、选择域标记<select>和<option>、文字域标记<textarea>等。

1. 输入域标记<input>

输入域标记<input>是表单中最常用的标记之一。常用的输入域标记有文本框、按钮、单选按钮、复选框等。语法格式如下：

```
<form>
<input name="file_name"  type="type_name">
</form>
```

参数 name 是指输入域的名称，参数 type 是指输入域的类型。在<input…type=”” >标记中一共提供了 10 种类型的输入域，用户选择使用的类型由 type 属性决定。type 属性的取值及举例见表 10-2。

表 10-2　type 属性的取值及举例

值	举　例	说　明
text	<input name="user" type="text" value="姓名" size="10" maxlength="2000">	name 为文本框的名称，value 是文本框的默认值，size 指文本框的宽度(以字符为单位)，maxlength 指文本框的最大输入字符数
password	<input name="pwd" type="password" value="123" size="10" maxlength="20">	密码域，用户在文本框中输入的字符将被替换显示为 *，以起到保密作用
file	<input name="file" type="file" enctype="multipart/form-data" size="6" maxlength="100">	文件域，当上传文件时，可用来打开一个模式窗口以选择文件。然后将文件通过表单上传到服务器
image	<input name="imageField" type="image" src="images/banner.gif" width="100" height="100" border="0">	图像域是指可以用在提交按钮位置的图片，这幅图片具有按钮的功能
radio	<input name="sex" type="radio" value="1" checked>男 <input name="sex" type="radio" value="0">女	单选按钮，用于设置一组选项，用户只能选择一项。checked 属性用来设置单选按钮默认被选中
checkbox	<input name="favor" type="checkbox" value="1" checked>唱歌 <input name="favor" type="checkbox" value="1">跳舞 <input name="favor" type="checkbox" value="1">画画	复选框，允许用户选择多个选项。checked 属性用来设置复选框默认被选中。例如，收集个人信息时，要求在个人爱好的选项中进行多项选择等
submit	<inputtype="submit" name="Submit" value="提交">	将表单的内容提交到服务器
reset	<inputtype="reset" name="Submit" value="重置">	清除与重置表单内容，用于清除表单中所有文本框的内容，并使选择菜单恢复到初始值
button	<inputtype="button" name="Submit" value="按钮">	按钮可以激发提交表单的动作，可以在用户需要修改表单时，将表单恢复到初始状态，还可以依照程序的需要发挥其他作用。普通按钮一般是配合 JavaScript 脚本进行表单处理的
hidden	<inputtype="hidden" name="bookid">	隐藏域，用于在表单中以隐含方式提交变量的值。隐藏域在页面中对于用户是不可见的，添加隐藏域的目的在于通过隐藏的方式收集或发送信息。浏览者单击“发送”按钮发送表单时，隐藏域的信息也被一起发送到 action 指定的处理页

2. 选择域标记<select>和<option>

通过选择域标记<select>和<option>可以建立列表或菜单。菜单的作用是为了节省空间，正常状态下只能看到菜单的一个选项，单击右侧的下三角按钮，打开菜单后才能看到

全部的菜单。列表可以显示一定数量的选项，如果超过这个数量，会自动出现滚动条，浏览者可以通过拖动滚动条来查看各个选项。

选择域标记的语法格式如下：

```
<select name="name" size="value" multiple>
<option value="value" selected>选项 1</option>
<option value="value" >选项 2</option>
<option value="value" >选项 3</option>
……
</select>
```

参数 name 表示选择域的名称；参数 size 表示列表的行数；参数 value 表示菜单选项的值；参数 multiple 表示以菜单方式显示数据，默认以列表方式显示数据。

选择域标记<select>和<option>的显示方式及示例见表 10-3。

表 10-3　选择域标记<select>和<option>的显示方式及举例

值	举　例	说　明
列表方式	<select name="spec" id="spec"> <option value="0" selected>羽毛球</option> <option value="1" selected>篮球</option> <option value="2" selected>乒乓球</option> </select>	下拉列表框，通过选择域标记<select>和<option>建立一个列表，列表可以显示一定数量的选项，如果超出这个数量，会自动出现滚动条，浏览者可以通过拖动滚动条来查看各个选项。selected 属性用来设置菜单默认被选中
菜单方式	<select name="spec" id="spec" multiple> <optionvalue="0" selected>羽毛球</option> <option value="1" selected>篮球</option> <option value="2" selected>乒乓球</option> </select>	multiple 属性用在菜单列表标记<select>中，指定用户可以使用 shift 和 ctrl 键进行多选

【说明】

在 Web 程序开发过程中，也可以通过循环语句动态地添加菜单项。

name 和 value 刚开始容易弄混，在提交的时候，name=value，在按钮类中 value 是显示，name 不是必需的。

但是在输入类和选择类中，都应有 name 和 value，用于提交数据，在特殊情况，比如新用户注册，value 可以没有默认值。

另外，如果提交的不是字符，比如<input type="file" name="file"/>中 value 不是必需的。

但是，对于初学者最好都写上 name 和 value，可以值为空。

3. 文字域标记<textarea>

文字域标记<textarea>用来建立多行的文字域，可以在其中输入更多的文本。其语法格式如下：

```
<textarea name="name" rows=value cols=value value="value" wrap="value">
……
</textarea>
```

参数 name 表示文字域的名称；参数 rows 表示文字域的行数；参数 cols 表示文字域的列数（这里的 rows 和 cols 以字节为单位）；参数 value 表示文字域的默认值；参数 wrap 用于设定显示和输出时的换行方式，值为 off 表示不自动换行，值为 hard 表示自动硬回车换行，换行标记一同被发送到服务器，输出时也会换行，值为 soft 表示自动软回车换行，换行标记不会发送到服务器，输出时仍然为一列。文字域标记<textarea>的值及说明见表 10-4。

表 10-4　文字域标记<textarea>的值及说明

值	举　例	说　明
textarea	<textarea name="remark" cols="20" rows="4" id="remark">请输入您的建议：</textarea>	文本域，也称为多行文本框，用于多行文本的编辑；wrap 属性默认为自动换行

把 HTML 代码和 PHP 代码都放在一个文件中，在实际项目中，由于前后端工作由不同的人承担，因此，一般将表单放到单独的 HTML 文件中，然后将业务逻辑代码放到单独的 PHP 文件中，接下来用特定的方法将 HTML 表单中的元素值提交到 PHP 文件以进行处理，处理完毕后，再将处理结果返回给用户。

10.1.3　表单数据的传递过程

1. 表单向服务器提交的信息内容

当单击表单的提交按钮后，表单将向服务器发送表单中填写的信息，发送形式是各个表单元素的“name=value&name=value&name=value…”。现在以图 10-1 所示的表单为例来分析表单向服务器提交的内容（输入密码是 123456）。

图 10-1　一个输入了数据的表单

该表单代码如下：

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

```
<p>用户名:<input name="user" id="xm" type="text" size="15"/></p>
<p>密码:<input name="pw" type="password" size="15"/></p>
<p>性别:男<input type="radio" name="sex" value="1"/>
        女<input type="radio" name="sex" value="2"/></p>
<p>爱好:<input name="fav1" type="checkbox" value="1">唱歌
        <input name="fav2" type="checkbox" value="2">跳舞
        <input name="fav3" type="checkbox" value="3">跑步
<p>所学专业<select name="maj">
  <option value="1">计算机</option>
  <option value="1">会计</option>
  <option value="1">商务英语</option>
  <option value="1">工程造价</option>
  </select></p>
<p>个性签名:<br/><textarea name="sign"></textarea></p>
<p><input type="submit" name="submit" value="提交"/></p>
</form>
```

表单向服务器提交的内容总是 name/value 信息对，对于文本类输入框来说，value 的值是用户在文本框中输入的字符。对于选择框（单选框、复选框和列表菜单）来说，value 的值必须事先设定，只有某个选项被选中后它的 value 值才会生效。因此图 10-1 提交的数据是：

```
user=小小 &pw=123456&sex=2&fav2=1&maj=1&sign=我选择,我喜欢! &submit=提交 </form>
```

【提示】

（1）如果表单中只有一个提交按钮，可去掉它的 name 属性（如 name="submit"），防止提交按钮的 name/value 属性对也一起发送给服务器，因为这些是多余的。

（2）<form>标记的 name 属性通常是为 JavaScript 调用该 form 元素提供方便的，没有其他用途。如果没有 JavaScript 调用该 form，则可省略其 name 属性。

2. 表单的三要素

一个最简单的表单必须具有以下三部分内容：

（1）<form>标记，没有它表单中的数据不知道提交到哪里去，并且不能确定这个表单的范围。

（2）至少有一个输入域（如 input 文本域或选择框等），这样才能收集到用户的信息，否则没有信息提交给服务器。

（3）提交按钮，没有它表单中的信息无法提交（当然，如果使用 ajax 等高级技术提交表单，表单也可以不具有第（1）项和第（3）项）。

10.2 获取 URL 参数

当 HTML 页面中的表单参数通过 POST 或 GET 方式提交到服务器的 PHP 代码后，

PHP 代码应该通过合适的方式来获取表单、URL 和 SESSION 变量的值。

GET 方式将表单内容附加在 URL 地址后面发送。提交的内容会以 name = value 的键值对的方式（name 是键，value 是值），在网址“?”后显示在地址栏，用 & 隔开多个变量。这种方式不安全，而且有长度限制。

POST 方式将表单中的信息作为数据块发送给服务器上的处理程序，在浏览器的地址栏不显示提交的信息。属于隐式提交，安全，而且长度没有限制。

浏览器可以通过表单或 URL 字符串向服务器发送数据，这些数据成为 HTTP 请求信息。PHP 提供了很多预定义的超全局变量，见表 10-5，用来获取 HTTP 请求信息，这些超全局变量的数据类型均为数组。

表 10-5　PHP 的超全局变量及功能

超全局变量	功　能
$_POST	获取客户端以 POST 方式发送的 HTTP 请求信息
$_GET	获取客户端以 GET 方式发送的 HTTP 请求信息
$_REQUEST	包含 $_GET、$_POST 和 $_COOKIE 三类数组中的信息
$_SERVER	获取 HTTP 请求中的环境变量信息
$_SESSION	存储和读取为单个用户保存的信息
$_COOKIE	读取用户的 Cookie 信息
$_FILE	获取通过 POST 方式上传文件时的相关信息，为多维数组

10.2.1　使用 $_POST［ ］获取表单数据

如果提交表单数据时，使用的是 POST 方式，可以在 PHP 程序中通过 $_POST［］预定义变量接收表单数据，格式如下：

```
变量名 = $_POST['name']
```

表示将获取的某个表单元素值（如一个文本框中的内容）保存到一个服务器端变量中。例如：$user = $_POST［"user"］，其中，$user 是自定义的一个变量名称，而后面的 user 则是一个表单元素的名称（name 属性值），两者不是一回事。

在实际应用中，表单的 HTML 代码和获取表单数据的 PHP 程序既可以分别写在两个文件中，也可以写在同一个文件中。

【例 10.2-1】编写程序获取用户登录时输入的用户名和密码。

（1）创建 main.php 文件，程序代码如下：

```
<html>
<body>
  <form method="post" action="login.php">
名字：<input type="text" name="username"><br>
密码：<input type="text" name="pw"><br>
<input type="submit" value="登录">
```

```
</form> </body>
</html>
```

（2）创建 login. php 文件，程序代码如下：

```
<? php
    $username= $_POST[ "username" ];
    $pw= $_POST[ "pw" ];
    echo "欢迎". $username."<br>";
    echo "您输入的密码是:". $pw;
? >
```

main. php 的运行结果如图 10-2（a）所示，单击“登录”按钮，就会将表单数据提交给 login. php，login. php 接收并显示数据，如图 10-2（b）所示。

(a)　　(b)

图 10-2

（a）main. php 运行结果；（b）login. php 运行结果

【说明】

（1）在例 10. 2-1 中，<form>标记的 method 属性值为 post，表示该表单提交数据时以 post 方式提交。因此，需用 $_POST [ ] 获取数据。

（2）在例 10. 2-1 中，<form>标记的 action 属性值为“login. php”，表示获取该表单的文件必须是“login. php”。

（3）在例 10. 2-1 中，有三个表单元素（两个文本框和一个提交按钮），其中两个文本框的 name 属性值分别为“username”和“pw”，因此，$_POST ["username"] 和 $_POST ["pw"] 分别表示获取这两个文本框的值。

上述例子也可以合并为一个网页文件，这样可以减少网站内网页文件的数量。

实现方法是：设置<form>标记的 action=''''或 action=''自身文件名''，然后将表单代码和 PHP 代码写在同一个文件中，并判断只有在用户提交了表单后才执行 PHP 代码，代码如下：

```
<form method="post" action="">
名字：<input type="text" name="username"><br>
密码：<input type="text" name="pw"><br>
<input type="submit" name="login" value="登录">
</form>
<? php
if( isset( $_POST[ \' login' ] ) ) {
    $username= $_POST[ ' username' ];
    $pw= $_POST[ ' pw' ];
```

```
    echo "欢迎". $username."<br>";
    echo "您输入的密码是:". $pw;
}
? >
```

运行结果如图 10-3 所示。

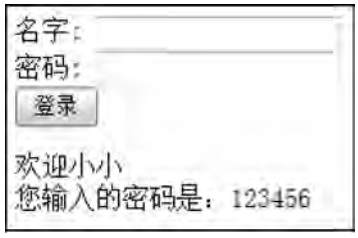


图 10-3　程序运行结果

【思考】

当用户输入信息后，上述例子中同时显示了表单界面和获取的信息，如果只希望输出获取信息，而不再显示表单，该如何修改上述代码?

也可以通过 POST 方式获取单选框、复选框、下拉框和多行文本域等表单数据。

【例 10.2-2】 编写程序，利用表单获取广大读者对某图书的反馈意见。

(1) 创建 main.php 文件，程序代码如下:

```
<html><body>
<h1>网上问卷调查</h1>
<form method="post" action="result.php">
姓名:<input type="text" name="name"><br>
性别:<input type="radio" name="sex" value="1">男
    <input type="radio" name="sex" value="2">女<br>
您使用本书是作为:<input type="checkbox" name="books[]" value="指定教材">指定教材
    <input type="checkbox" name="books[]" value="选用教材">选用教材
    <input type="checkbox" name="books[]" value="辅导教材">辅导教材
    <input type="checkbox" name="books[]" value="自学教材">自学教材<br>
您的专业是:<select name="major">
    <option value="计算机应用">计算机应用</option>
    <option value="软件技术">软件技术</option>
    <option value="大数据">大数据</option>
</select><br>
反馈意见:<textarea name="view" rows="4" cols="40"></textarea><br>
    <input type="submit" value="提交">
</form>
</body></html>
```

（2）创建 result. php 文件，程序代码如下：

```
<? php
 $name = $_POST[ "name" ];
 $sex = $_POST[ "sex" ];
 $books = $_POST[ "books" ];
 $major = $_POST[ "major" ];
 $view = $_POST[ "view" ];
 $booksnum = count( $books);
echo "尊敬的". $name;
if( $sex = = 1) echo "先生".' <br>' ;
if( $sex = = 2) echo "女士".' <br>' ;
echo "您使用这本书是作为:";
for( $i = 0; $i< $booksnum; $i++)
      echo  $books[ $i].' ' ;
echo ' <br>' ;
echo "您的专业是:". $major.' <br>' ;
echo "您的意见是:". $view.' <br>' ;
? >
<p><a href = "JavaScript:history.go( -1)" >返回修改</a></p>
```

程序运行结果如下图 10-4 所示。

图 10-4

（a）main. php 运行结果；（b）result. php 运行结果

【说明】

（1）对于单选框，两个单选框的 name 属性值一样，就表示这是一组，只能选其中一个。

（2）对于复选框，三个复选框的 name 属性值相同，也表示一组，但复选框可以选择多个，如果选中多个，则多个复选框的值将保存在一个数组中，可以用循环语句输出所有被选中的值。

（3）总的来说，表单元素分为两类：

1）对于文本框、密码框、多行文本框这些需要用户输入内容的，$_POST [ ] 获取的就是用户输入的内容。

2）对于单选框、复选框、下拉列表和隐藏域等这些无须用户输入内容的，$_POST［］获取的就是选中项的 value 值，因此必须设置 value 属性值。

（4）当多个复选框属于同一组时，对其 name 属性值命名，一定要命名成数组的形式，如 name="books［］"。这样 $_POST［"books"］返回的结果才会是一个数组，可以利用数组名加索引值来取得其中某个复选框的 value 值，例如 $_POST［"books"］［1］的值为“选用教材”。还可利用 count（）方法获得该数组中的元素总数。

10.2.2　使用 $_GET［］获取表单数据

如果表单是以 GET 方式提交的，即 method 属性值为 GET（或没有设置），则必须用 $_GET［］数组获取表单中数据。下面将例 10.2-1 修改为以 GET 方式提交数据，程序代码如下：

```
<html>
<body>
<form method="get" action="login.php">
名字：<input type="text" name="username"><br>
密码：<input type="text" name="pw"><br>
<input type="submit" value="登录">
</form>
</body>
</html>
```

"login.php" 的程序代码如下：

```
$username= $_GET["username"];
$pw= $_GET["pw"];
echo "欢迎". $username."<br>";
echo "您输入的密码是：". $pw;
```

也可以通过 GET 方式获取单选框、复选框、下拉框和多行文本域等表单数据，方法同 POST 方式。

10.2.3　使用 $_GET［］获取 URL 字符串信息

URL 字符串又称为查询字符串，这些字符串信息总是以 GET 方式提交给服务器的，可以使用 $_GET［］方法获取 URL 字符串信息。

1. URL 字符串的定义

有些网页的 URL 后面会接一些以“?”号开头的字符串，这称为 URL 字符串，例如：

```
https://www.baidu.com/login.php? user=小红 &age=20
```

其中，“? user=小红 &age=20”就是一个 URL 字符串，它包含两个 URL 变量（user 和 age），而“小红”和“20”分别是这两个 URL 变量的值，URL 变量和值之间用“=”号连接，多个 URL 变量之间用“&”连接。

URL 字符串会连同 URL 信息一起作为 HTTP 请求数据提交给服务器端的相应文件，

例如上面的 URL 字符串信息将提交给 login. php。利用 $_GET［］可以获取 URL 字符串中变量的值。例如在 login. php 中编写以下代码，就能获取到这些查询变量的值了：

```
<?
    $user= $_GET["user"];          //获取变量 user 的值,返回"小红"
    $age = $_GET["age"];           //获取变量 age 的值,返回"20"
? >
```

2. 设置 URL 字符串的方法

当网页通过超链接或其他方法从一张网页跳转到另一张网页时，往往需要在跳转的同时把一些数据传递到第二张网页中。可以把这些数据作为 URL 字符串附在超链接的 URL 后，就可以在第二张网页中使用 $_GET［］获取 URL 变量的值，例如：

```
<a href="search.php? key=Web 标准 &pageNo=5">查询结果第 5 页</a>
```

则在 search. php 中就可用 $_GET［］获取第一张网页传递来的 URL 变量的值。这是通过超链接设置 URL 字符串，第二种方法是在<form>标记的 action 属性中设置。

【例 10. 2-3】 在超链接中设置 URL 字符串并获取。

（1）创建 main. php 网页，程序代码如下：

```
<html><body>
<ul>
    <li><a href="get.php? id=1">PHP 程序设计</a></li>
    <li><a href="get.php? id=2">ASP.NET 程序设计</a></li>
    <li><a href="get.php? id=3">JSP 程序设计</a></li>
</ul>
</body></html>
```

（2）创建 get. php 网页，程序代码如下：

```
<? php
 $id=intval( $_GET["id"]);
if( $id==1)
    echo "<p>这是第一本书</p>";
elseif( $id==2)
    echo "<p>这是第二本书</p>";
elseif( $id==3)
    echo "<p>这是第三本书</p>";
else
    echo "<p>参数非法</p>";
? >
```

【说明】

main. php 中所有链接都是链接到同一网页的，只是设置了不同的 URL 变量值，就可以使 get. php 根据不同链接传来的不同 ID 值显示对应的网页内容。

【例 10. 2-4】 在<form>标记的 action 属性中设置 URL 字符串。

程序如下：

```
<html>
<body>
<? php
 $flag= $_GET[ "flag" ];
if( $flag= =' 1' )
      echo "欢迎". $_POST[ ' user' ].' 光临' ;
else
      echo ' <form method=" post" action=" ? flag=1" >
           姓名:<input name=" user" type=" text" size=" 15" />
           <input type=" submit" value=" 提交" />
           </form>'
? >
</body>
</html>
```

【说明】

在<form>标记中，“action="? flag=1"” 省略了文件名表示将表单提交给自身，设置查询字符串 flag=1 用来判断用户是否单击了“提交”按钮，一旦单击了 URL 地址后会增加“? flag=1”，因此可据此输出不同的内容。

3. 设置 URL 字符串的方法总结

如果要设置 URL 字符串，以便将 URL 字符串中的信息传递给相应网页，有以下方法：

（1）在超链接的 href 属性值中的 URL 后添加 URL 字符串；

（2）在超链接的 action 属性值中的 URL 后添加 URL 字符串；

（3）直接在浏览器地址栏中的网页 URL 后手工输入 URL 字符串。

显然，普通用户不会使用第（3）种方法设置 URL 字符串，因此一般使用方法（1）或（2）诱导用户将 URL 变量传递给相关网页。

【提示】

表单如果设置为 GET 方式提交，那么表单中的数据将转换成 URL 字符串发送给服务器。此时若在表单的 action 属性值中也设置 URL 字符串，那么将发生冲突，action 属性值中的 URL 字符串将无效。因此如果在 action 属性值中有 URL 字符串，则表单中只能用 POST 方式提交，如例 10.2-4。

10.3　页 面 跳 转

10.3.1　用 header（）函数实现页面跳转

在 PHP 中进行页面跳转使用 header（）函数是一种十分简单的方法。header（）函数的主要功能是将 HTTP 协议标头（header）输出到浏览器。

header（）函数的定义如下：

```
void header (string string [ ,bool replace [ ,int http_response_code] ])
```

可选参数 replace 指明是替换前一条类似标头还是添加一条相同类型的标头，默认为替换。

第二个可选参数 http_response_code 强制将 HTTP 相应代码设为指定值。header（）函数中 Location 类型的标头是一种特殊的 header（）调用，常用来实现页面跳转。

在 PHP 中，使用 header（）函数可以重定向网页，例如：

```
<?
    header("location:http://www.baidu.com");          //重定向绝对 URL
    header("location:5-8.php");                       //重定向相对 URL
    header("location:? flag=1");                      //重定向到本页,并增加 URL 字符串
    $url='5-1.php';
    header("location: $url");                         //重定向到变量表示的网址
? >
```

【注意】

（1）location 和“:”号之间不能有空格，否则会出错。

（2）在 header 语句之前，服务器不能向客户端发送任何数据，如不能有 echo 语句，也不能有 HTML 代码（包括空行）。

（3）如果 PHP 代码中有多条重定向语句，则会重定向到最后一条语句中的 URL，表明 PHP 在执行重定向语句后，仍然会继续执行后面的语句。如果希望执行完重定向语句后立即停止脚本执行，应在 header 语句后使用 exit（）或 die（）方法退出。

（4）header 函数的功能和 JavaScript 脚本中 location. href 的功能有些相似，如“header ("location: 5-8. php");”又可使用“echo"<script>location. href='5-8. php'; </script>";”来实现。不过 header（）函数要求在重定向之前不允许服务器向浏览器输出任何内容，因此使用该方法要么确保先用 ob_start（）打开服务器缓冲区，使所有的内容先输出到缓存中（还没有输出到浏览器），要么确保在 header（）语句之前没有任何内容输出到页面。

因此下面的写法是错误的：

```
<html><body>
<? header("location:5-8.php"); ? >
</body></html>
```

10.3.2 用 Meta 标签实现页面跳转

Meta 标签是 HTML 中负责提供文档元信息的标签，在 PHP 程序中使用该标签，也可以实现页面跳转。若定义 http-equiv 为 refresh，则打开该页面时将根据 content 规定的值在一定时间内跳转到相应页面。

若设置 content="秒数；url=网址"，则定义了经过多长时间后页面跳转到指定的网址。例如，使用 meta 标签实现后页面自动跳转到 LAMP 兄弟连官方论坛：

```
<meta http-equiv="refresh" content="1;url=http://bbs.lampbrother.net">
```

例如，以下程序 meta.php 实现在该页面中停留一秒后页面自动跳转到 bbs.lampbrother.net：

```
<? php
 $url = "http://bbs.lampbrother.net";
? >
<html>
  <head>
    <meta http-equiv="refresh" content="1;url=< ? php echo $url; ? >">
  </head>
  <body>
    页面只停留一秒……
  </body>
</html>
```

10.3.3　在 JavaScript 脚本中实现页面跳转

在 PHP 中，使用 JavaScript 脚本可以实现页面跳转，方法有以下几种：

（1）window.location.href = "http://www.111cn.net/";　　//这个方法比较常用

（2）self.location = "http://www.111cn.net/";

（3）top.location = "http://www.111cn.net/";

（4）window.navigate ("http://www.111cn.net/");

使用 JavaScript 脚本实现页面跳转一般结合 echo ( )。echo ( ) 是最常用的方法，它用来将服务器端的数据发送给浏览器。所发送的信息可以是字符串变量、变量、HTML 代码、JavaScript 代码等所有浏览器能解释的代码。下面是使用 echo 方法输出信息的示例，此代码可以放在程序中的任何合法位置：

```
< ? php
    $url = "http://bbs.lampbrother.net";
    echo "< script language='javascript' type='text/javascript'>";
    echo "window.location.href='$url'";
    echo "</script>";
? >
```

10.4　会 话 控 制

10.4.1　Cookie 会话技术

Cookie 是一种在客户端浏览器中存储数据并以此跟踪和识别用户的机制。简单地说，

Cookie 是 Web 服务器暂时存储在用户硬盘上的一个文本文件，并随后被 Web 浏览器读取。当用户再次访问该网站时，网站通过读取 Cookie 文件记录这位访客的特定信息，如上次访问的位置、花费的时间、用户名和密码等，从而迅速做出响应。例如，在页面中不需要输入用户的 ID 和密码即可直接登录网站等。也有一些 Cookie 的高级应用，如在购物网站浏览商品页面时，该网站程序可以将用户的浏览历史记录到 Cookie 中，当用户下次再访问时，网站根据用户过去的浏览情况为用户推荐感兴趣的内容。

Cookie 有两种形式：会话 Cookie 和永久 Cookie。

会话 Cookie：不设置过期时间，则表示这个 Cookie 生命周期为浏览器会话期间，只要关闭浏览器窗口，Cookie 就消失了。这种生命期为浏览会话期的 Cookie 被称为会话 Cookie。会话 Cookie 一般不保存在硬盘上而是保存在内存里。

永久 Cookie：设置了过期时间，浏览器就会把 Cookie 保存到硬盘上，关闭后再次打开浏览器，这些 Cookie 依然有效直到超过设定的过期时间。存储在硬盘上的 Cookie 可以在不同的浏览器进程间共享，比如两个 IE 窗口。而对于保存在内存里的 Cookie，不同的浏览器有不同的处理方式。

1. 创建 Cookie

创建 Cookie 最简单的方法是使用 setcookie（）函数，语法如下：

```
setcookie(name, value, expire, path, domain, secure);
```

其中，name 用来定义一个 Cookie 的变量名，value 用来设置 Cookie 变量值，expire 用来定义 Cookie 的有效期；而 path、domain、secure 分别用来规定 Cookie 的有效目录、有效域名以及是否采用 HTTPS 来传输 Cookie，这三个参数不常用。除了 name 和 value 是必需的参数外，其他参数都是可选的。

例如：

```
<? php
setcookie('temporary', 'Alex Porter');                //不设置过期时间
setcookie('user', 'Alex Porter', time()+3600);        //设置过期时间为 1 小时,永久 Cookie
setcookie('age',21, time()+3600,' ',' ',false);       //设置 setcookie 的所有参数
? >
<html>
<body>
</body>
</html>
```

上例中设置了 3 个 Cookie 变量，变量名分别为 temporary、user 和 age。其中 temporary 没有设置过期时间，因此它仅仅是个会话 Cookie，会话 Cookie 并没有保存到文本文件中，关闭浏览器后，temporary 将立即失效。而 user 和 age 均设置了过期时间，因此是永久 Cookie，它们将在关闭浏览器 1 小时后失效。在发送 Cookie 时，Cookie 的值会自动进行 URL 编码，在取回时进行自动解码（为防止 URL 编码，请使用 setrawcookie（）取而代之）。

注意：

（1）setcookie（）函数必须位于 <html> 标签之前。

（2）Cookie 变量的值总是字符串数据类型。

还可以通过另一种方式设置 Cookie 的过期时间，这也许比使用秒表示的方式简单：

```
<? php
 $expire=time()+60 * 60 * 24 * 30;
setcookie("user", "runoob", $expire);
? >
```

在上面的实例中，过期时间被设置为一个月（60 * 60 * 24 * 30）。

2. 读取 Cookie

在 PHP 中可以直接通过超级全局数组 $_COOKIE [ ] 来读取浏览器端的 Cookie 值。

在下面的例子中，我们取回了名为“user”的 Cookie 的值，并把它显示在了页面上：

```
<? php
    //输出 Cookie 值
    echo $_COOKIE["user"];
    //查看所有 Cookie
    print_r( $_COOKIE);
? >
```

在下面的例子中，我们使用 isset（）函数来确认是否已设置了 Cookie：

```
<html>
<body>
<? php
if (isset( $_COOKIE["user"]))
    echo "Welcome ". $_COOKIE["user"] . "! <br />";
else
    echo "Welcome guest! <br />";
? >
</body>
</html>
```

在 PHP 中可以直接通过超级全局数组 $_COOKIE [ ] 来读取浏览器端的 Cookie 值：

```
<? php
if(! isset( $_COOKIE['time' ]))
    setcookie('time' ,date('y-m-d H:i:s' ));
    echo "欢迎首次访问网站!";
}else{
    setcookie('time' ,date('y-m-d H:i:s' ),time()+60);
    echo "上次访问时间为;". $_COOKIE['time' ];
    echo "<br>";
}
echo "您本次访问网站时间为:".date("y-m-d H:i:s");
? >
```

以上程序首先使用 isset（） 函数检测 Cookie 文件是否存在。如果不存在，则使用

setcookie（）函数创建一个 Cookie，并输出相应的字符串；如果 Cookie 文件存在，则使用 setcookie（）函数设置 Cookie 文件的有效时间，并输出用户上次访问网站的时间。最后在页面上输出本次访问网站的当前时间。

3. 删除 Cookie

Cookie 被创建之后，如果没有设置 Cookie 的失效时间，在关闭浏览器的时候会自动删除 Cookie 文件。下面介绍两种删除 Cookie 的方法。

（1）使用 setcookie（）函数删除 Cookie。删除 Cookie 和创建 Cookie 的方法类似，删除 Cookie 也使用 setcookie（）函数。为了删除 Cookie，可以将 Cookie 的变量值设置为空，并且不设置有效期（不设置有效期将删除 Cookie 文件中的 Cookie 变量）。也可以将 Cookie 的有效期设置为过去的某个时间。不管使用哪种方法，浏览器接收到 Cookie 响应头信息后，将自动删除用户硬盘中的 Cookie 文件和内存中的 Cookie 信息。例如：

```
<? php
setcookie('user', 'Alex Porter',' ');                    //删除 Cookie 的方法 1
setcookie('age',21, time()-3600,' ',' ',false);  //删除 Cookie 的方法 2
var_dump( $_COOKIE);          //用来查看上述 Cookie 数组元素是否已经删除
? >
```

（2）在浏览器中手动删除 Cookie。在使用 Cookie 时，会自动生成一个文本文件，存储在浏览器的 Cookies 临时文件夹中，在浏览器中删除 Cookie 文件是一种非常便捷的方法。具体操作步骤如下：

启动浏览器，选择【工具】→【Internet 属性】命令，打开【Internet 属性】对话框，如图 10-5 所示。在【常规】选项卡中单击【删除】按钮，弹出【删除浏览历史记录】对话框，选中【Cookie 和网站数据】复选框，单击【删除】按钮，如图 10-6 所示，即可成功删除全部 Cookie 文件。

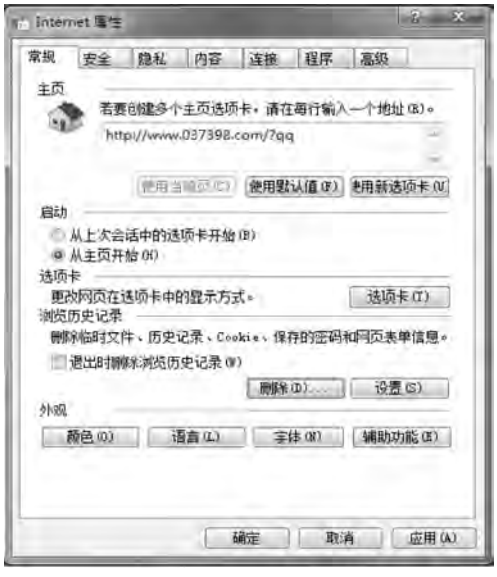


图 10-5 【Internet 属性】对话框

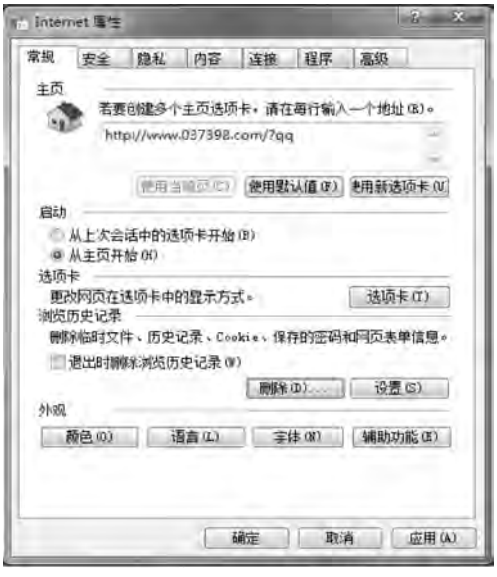


图 10-6 选中删除 Cookie 的选项

4. 浏览器不支持 Cookie 的情况

如果您的应用程序需要与不支持 Cookie 的浏览器打交道，那么您不得不使用其他的办法在您的应用程序中的页面之间传递信息。其中一种方式就是通过表单传递数据，上节已介绍。

10.4.2　Session 会话技术

由于 Cookie 是存储在客户端浏览器中的，有长度的限制，并以文件的形式存在，显而易见是不安全的，另外，Cookie 也不能跨站访问。基于此，提供了会话（Session）技术。所谓“会话”，就是访客从访问网站开始到离开网站的时间范围。

有时我们需要在用户访问网站过程中记住用户的一些信息，例如用户登录以后，网站中的所有页面都能显示用户的登录名，这就需要在整个网站中使用一种“全局变量”保存用户名。但是普通变量的作用域只能在一个网页内，当用户从一张网页跳转到另一张网页时，前一张网页中以变量、常量形式存放的数据就丢失了。只要把用户的信息存储在 Session 变量中，用户在网站页面之间跳转时，存储在 Session 变量中的信息就不会丢失，而是在整个用户会话中一直存在。如果没有 Session，那么用户每进入一个页面都需要登录一次用户名和密码。

Session 的中文是“会话”的意思，在 Web 编程中 Session 代表了服务器与客户端之间的“会话”，意思是服务器和客户端在不断地交流。如果不使用 Session，则客户端每一次请求都是独立存在的。当服务器完成某次用户的请求后，服务器将不能再继续保持与该用户浏览器的连接。这样当用户在网站的多个页面间切换时（请求了多个页面），页面之间无法传递用户的相关信息。

1. 创建 Session

当用户请求网站中的任意一个页面时，若用户尚未建立 Session 对象（如第一次访问），则服务器会自动为用户创建一个 Session 对象（它包含唯一的 Session ID 和其他 Session 变量），并保存在服务器内存中，不同用户的 Session 对象存储着各自特定的信息，服务器将 Session ID 发送到客户端浏览器，而浏览器则将该 Session ID 保存在会话 Cookies 中。当浏览器再次向服务器发送 HTTP 请求时，会将 Session ID 信息一起发送给服务器。服务器根据该 Session ID 查找到对应的 Session 对象，就能识别出用户。这很有利于服务器对用户身份的鉴别，从而实现 Web 页面的个性化。

在 PHP 中，使用 Session 前都需要在页面开头用 session_start（）方法开启 Session 功能，session_start（）函数必须位于 <html> 标签之前，如下所示：

```
<? php session_start(); ? >
<html>
<body>
……
</body>
</html>
```

session_start（）的功能是初始化 Session，也标识着 Session 生命周期的开始。要使用

Session，必须初始化一个 Session 环境，有点类似于 OOP 概念中调用构造函数构建对象实例一样。Session 初始化操作，声明一个全局数组 $_SESSION，映射寄存在内存的 Session 数据。如果 Session 文件已经存在，并且保存有 Session 数据，Session_start（）则会读取 Session 数据，填入 $_SESSION 中，开始一个新的 Session 生命周期。

存储和取回 Session 变量的正确方法是使用 $_SESSION 变量：

```
<? php
session_start( );
// 存储 session 数据
 $_SESSION[ ' views' ] = 1;
? >

<html>
<head>
<meta charset = " utf-8" >
<title>存储和读取 Session</title>
</head>
<body>

<? php
// 检索 session 数据
echo " 浏览量:". $_SESSION[ ' views' ];
? >

</body>
</html>
```

在下面的实例中，我们创建了一个简单的 page-view 计数器。isset（）函数检测是否已设置“views”变量。如果已设置“views”变量，则累加计数器。如果“views”不存在，则创建“views”变量，并把它设置为 1：

```
<? php
session_start( );
if( isset( $_SESSION[ ' views' ] ) )
{
      $_SESSION[ ' views' ] = $_SESSION[ ' views' ] +1;
}
else
{
      $_SESSION[ ' views' ] = 1;
}
echo " 浏览量:". $_SESSION[ ' views' ];
? >
```

【例 10.4-1】 存储 Session 信息。

程序如下：

```
<? php
session_start( );
 $_SESSION[ "uname" ] = "小红";
 $_SESSION[ "uname" ] = "李四";
 $_SESSION[ "sex" ] = "男";
 $_SESSION[ "age" ] = 18;
 $_SESSION[ "user" ] = array( 'name' =>' 明明' ,' psd' =>' 123' );
? >
```

【例 10.4-2】 读取例 10.4-1 的 Session 信息。

程序如下：

```
<? php
session_start( );
echo  $_SESSION[ "uname" ]."<br>";
echo  $_SESSION[ "sex" ]."<br>";
echo  $_SESSION[ "age" ]."<br>";
? >
```

说明：

（1）session_start（）函数前面不能有任何代码输出到浏览器，最好加在页面头部，或先用 ob_start（）函数打开输出缓冲区。

（2）对一个不存在的 Session 变量赋值，将自动创建该变量；给一个已经存在的 Session 变量赋值，将修改其中的值。

（3）必须要先存储 Session 的值，才能读取到 Session 信息，所以上例，必须先运行例 10.4-1，再运行例 10.4-2。

2. 设置 Session 的有效时间

Session 对象的生命期从用户在 Session 有效期内第一次访问网站直到不再访问网站为止的这段时间，即一个 Session 开始于用户打开这个网站中的任意一个网页；结束于用户不再访问这个站点，包括 Session 超时或主动删除 Session 两种情况。

Session 对象并不是一直有效的，它有个有效期，默认为 24min。如果客户端超过 24 分钟没有刷新网页或访问网站中的其他网页，则该 Session 对象就会自动结束。不过可以修改 Session 对象的默认有效期，一种方法是在 PHP 的配置文件 php.ini 中修改系统默认值，另一种方法是利用 ini_set（）方法更改 Session 对象的默认有效期，代码如下：

```
session_start( );
ini_set( 'session.save_path' ,'/tmp/' );                    //设置保存路径
int_set( 'session.gc_maxlifetime' ,60);                     //保存 1 分钟
setcookie(session_name( ),session_id( ),time( )+60,"/");    //设置会话 Cookie 的过期时间
```

【提示】

（1）虽然延长 Session 的有效期有时能方便用户访问，但这也会导致 Web 服务器内存

中保存的用户 Session 信息的时间增长，如果访问的用户很多，会加重服务器的负担。

（2）不能单独对某个用户的 Session 设置有效期。

3. 删除和销毁 Session

删除 Session 常用来实现用户注销的功能，使得用户能够安全退出网站。在 PHP 中，使用 unset（）方法可以删除单个 Session 变量。使用 session_unset（）函数可删除当前内存中 $_SESSION 数组中的所有元素，它等价于 $_SESSION = array（）或 unset（$_SESSION），例如：

```
<? php
session_start( );
unset( $_SESSION[ "name" ] );        //删除 $_SESSION 中一个 Session 变量
session_unset( );                    //删除 $_SESSION 中所有 Session 变量
? >
```

但是，session_unset（）只能删除 $_SESSION 数组中的所有元素，并不能删除对应的 Session ID，也不能删除保存 Session ID 的文件。而 session_destroy（）函数就能删除 Session ID，并销毁 Session 文件，但它不会删除内存中的 $_SESSION 数组中的所有元素，例如：

```
<? php
session_start( );
echo' <p>这个用户的 Session 编号为' .session_id( ).' </p>' ;
 $_SESSION[ "name" ] = "小红";
session_destroy( );                  //清除 Session ID
echo' <p>这个用户的 Session 编号为' .session_id( ).' </p>' ;
echo  $_SESSION[ "user_name" ];      //输出"小红"
? >
```

因此，如果要彻底删除 Session，实现用户安全注销功能，可以将 session_unset（）与 session_destroy（）函数结合使用，并且还需清除浏览器中的会话 Cookie 信息，这可以通过调用 setcookie（）函数将会话 Cookie 设置为过期。

10.4.3 Cookie 与 Session 的区别和联系

HTTP 是无状态的协议，客户每次读取 Web 页面时，服务器都打开新的会话，而且服务器也不会自动维护客户的上下文信息。Session 就是一种保存上下文信息的机制，针对每一个用户，Session 的内容在服务器端，通过 Session ID 来区分不同的客户。

Session 与 Cookie 的主要区别如下：

（1）存放位置。Session 保存在服务器上，客户端不知道其中的信息；Cookie 保存在客户端，服务器可以知道其中的信息。

（2）存放形式。Session 中保存的是对象，Cookie 中保存的是字符串。

（3）路径。Ssession 不能区分路径，同一个用户在访问网站期间，所有的 Session 在任何一个地方都可以访问到；而 Cookie 中如果设置了路径参数，那么同一个网站中不同路径下的 Cookie 是不能互相访问的。

（4）用途。Session 适合做客户的身份验证，Cookie 适合保存用户的个人设置、爱好等。

（5）安全性。Cookie 不是很安全，别人可以分析存放在本地的 Cookie 并进行 Cookie 欺骗，Session 较 Cookie 更安全一些

（6）大小及数量限制。单个 Cookie 在客户端的大小限制是 3KB，就是说一个站点在客户端存放的 Cookie 不能超过 3KB。不同浏览器所含 Cookie 的最大个数不同，一般 30~50 个；一般认为 Session 没有大小限制。

Session 与 Cookie 的联系：

Session 需要借助 Cookie 才能正常工作，如果客户端完全禁止 Cookie，Session 将失效，因为 Session 是由应用服务器维持的一个服务端的存储空间，用户在连接服务器时，会由服务器生成唯一的 SesssionID，用该 SessionID 为标识来存取服务端的 Session 空间。而 SessionID 存储在 Cookie 中，用户提交页面时会将这个 SessionID 提交到服务端，来存取 Session 数据．这一过程是不用开发人员干预的，所以一旦客户端禁用 Cookie，那么 Session 也会失效。

10.5　综 合 案 例

10.5.1　基本能力实训项目

1. 实训题目

制作用户注册表单。

2. 实训目的

熟练和掌握表单的基本操作。

3. 实训内容

用户点击“提交”按钮，提交各表单值，效果如图 10-7 所示。

图 10-7　效果示意图

参考代码如下：

```
<form id="form1" runat="server" method="post" action="login.php">
    <table border="1" cellspacing="1" cellpadding="1" align="center">
    <caption>
    用户注册
    </caption>
    <tr>
        <td width="100" align="right">
        用户名：
</td>
        <td width="150">
        <input type="text" name="uname" value="请输入用户名"/>
        </td>
        </tr>
        <tr>
        <td align="right">
        照片：
        </td>
        <td>
        <input type="file" name="file"/>
        </td>
        </tr>
<tr>
        <td width="100" align="right">
        密码：
        </td>
        <td width="150">
        <input type="password" name="pwd" value=""/>
        </td>
        </tr>
        <tr>
        <td width="100" align="right">
        确认密码：
        </td>
        <td width="150">
        <input type="password" name="pwd" value=""/>
        </td>
</tr>
<tr>
        <td width="100" align="right">
        性别：
```

```
        </td>
    <td width="150">
        <input type="radio" name="sex" id="s1" value="nan"/>
        <label for="s1">男</label>
        <input type="radio" name="sex" id="s2" value="nv"/>
        <label for="s2">女</label>
        <input type="radio" name="sex" id="s0" value="baomi"/>
        <label for="s0">保密</label>
        </td>
        </tr>
        <tr>
        <td width="100" align="right">
        专业:
        </td>
        <td width="150">
            <input type="checkbox" id="a1" name="ck[]"/>
            <label for="a1">Java</label>
            <input type="checkbox"  id="a0" name="ck[]"/>
            <label for="a0">.net</label>
            <input type="checkbox"  id="a3" name="ck[]"/>
            <label for="a3">PHP</label>
        </td>
        </tr>
        <tr>
        <td width="100" align="right">
        出生年份:
        </td>
        <td width="150">
          <select name="year">
            <option value="2000">2000</option>
            <option value="2001">2001</option>
            <option value="2002">2002</option>
            <option value="2003">2003</option>
          </select>
        </td>
    </tr>
    <tr>
        <td width="100" align="right">
            自我介绍:
        </td>
        <td width="150">
```

```
            <textarea rows="4" cols="12"></textarea>
        </td>
    </tr>
    <tr>
        <td colspan="2" align="center">

        <input type="submit" name="" value="提交" />
        <input type="reset" name="" value="重置" />  
    </td>
    </tr>
</table>
```

注：请读者根据前面例子，自行创建 login. php 文件，实现获取表单值。

10.5.2 拓展能力实训项目

1. 实训题目

利用 session 实现用户登录。

2. 实训目的

通过本次项目实训掌握如何利用 Session 跨页面传值。

3. 实训内容

设置简单的登录界面，通过 Session 进行跨页面传值，实现用户登录功能，效果如图 10-8 所示。

图 10-8　效果示意图

（1）创建 login. php 文件，具体参考代码如下：

```
<form action="main.php" method="post">
<table border="0" align="center">
<tr><td colspan="2" align="center">用户登录</td></tr><tr>
<tr><td align="center">用户名:</td><td><input name="username" type="text"></td></tr>
<tr><td align="center">密码:</td><td><input name="pw" type="password"></td></tr>
<tr><td colspan="2" align="center">
<input type="submit" name="submit" value="登录">
<input type="reset" name="submit2" value="重置"></td></tr><tr>
</table>
```

```
</form>
<? php
session_start( );
if( isset( $_POST[ ' submit' ] ) ) {
      $username = $_POST[ ' username' ] ;
      $pw = $_POST[ ' pw' ] ;
      if( $username = = " admin" && $pw = " 123" ) {
            $_SESSION[ ' username' ] = $username;
            $_SESSION[ ' pw' ] = $pw;
           header( " location:login.php" ) ;
      }
      else
           echo " <script>alert( ' 登录失败' ) ;location.href = ' main.php' ;</script>" ;
}
```

（2）创建 main. php 文件，具体参考代码如下：

```
<? php
session_start( ) ;
 $username = @ $_SESSION[ ' username' ] ;
 $pw = @ $_SESSION[ ' pw' ] ;
if( $username)
      echo " 欢迎管理员登录,您的密码为 $pw" ;
else
      echo " 对不起,您没有权限登录本页" ;
? >
```

上述例子也可以通过 Cookie 来实现，效果如图 10-9 所示，具体功能由读者完成。

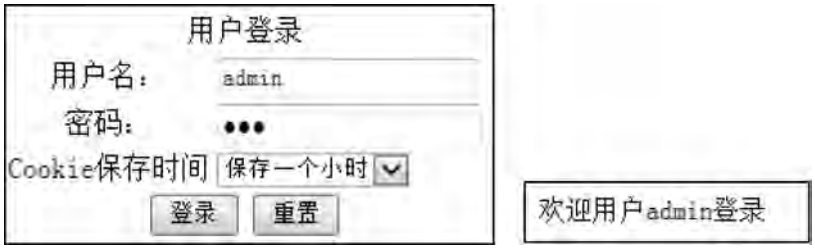


图 10-9　效果示意图

本章小结

本章主要介绍了创建表单及表单元素、提交和获取表单数据、Cookie 和 Session 等内容，通过本章的学习，读者应能够掌握如何实现网页和服务器程序如何传递数据，从而实

现页面和服务器端 PHP 程序的交互，利用 Session 和 Cookie 实现在页面中不需要输入用户 ID 和密码即可以直接登录网站、观察用户什么时候访问网站和什么时候离开网站等操作。

习　题

编程题

1. 尝试创建一个表单，在表单中添加各个常用的元素，并对表单元素命名。

2. 开发一个简单的搜索引擎页面，并获取输入的关键字。

3. 下面是一个获取表单提交信息的程序，名称为 get. php，请写出一个能让它获取得到数据的表单代码。

```
<? php
 $name = $_POST[ "name" ];
 $sex = $_POST[ "sex" ];
 $age = $_POST[ "age" ];
? >
```

4. 开发一个用户注册页面，并输出用户的注册信息。

5. 编写程序，同时使用 Session 和 Cookie 来保存用户的登录信息。

6. 编写程序，使用 Session 变量统计网站在线人数。

第 11 章

MySQL 数据库基础

【本章内容简介】

数据库作为程序中数据的主要载体，在整个项目中扮演着重要的角色。PHP 自身可以与大多数数据库进行连接，但 MySQL 数据库是开源界所公认的与 PHP 兼容性最好的数据库。MySQL 具有安全、跨平台、体积小和高效等特点，可谓 PHP 的“黄金搭档”。MySQL 操作简单、易学易用，受到很多业界人士的青睐。

【本章学习要求】

通过本章的学习，要求掌握 MySQL 的基本概念，掌握数据库的创建和数据表的基本操作。

11.1 数据库概述

PHP 在开发 Web 站点或一些管理系统时，需要对大量的数据进行保存。XML 文件和文本文件虽然可以作为数据的载体，但不易进行管理和对大量数据的存储，所以在项目开发时，数据库显得非常重要。PHP 可以连接的数据库种类较多，其中 MySQL 数据库与其兼容较好，在 PHP 数据库开发中被广泛应用。

11.1.1 MySQL 的定义

MySQL 是一款安全、跨平台、高效，并与 PHP、Java 等主流编程语言紧密结合的数据库系统。该数据库系统是由瑞典的 MySQL AB 公司开发、发布并支持，由 MySQL 的初始开发人员 David Axmark 和 Michael Monty Widenius 于 1995 年建立的。MySQL 的象征符号是一只名为 Sakila 的海豚，代表着 MySQL 数据库的速度、能力、精确和优秀本质。

目前 MySQL 被广泛应用在 Internet 上的中小型网站中。其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，使得很多公司采用 MySQL 数据库以降低成本。

MySQL 数据库可以称得上是目前运行速度最快的 SQL 语言数据库之一。除了具有许多其他数据库所不具备的功能外，MySQL 数据库还是一种完全免费的产品，用户可以直接通过网络下载 MySQL 数据库，而不必支付任何费用。

11.1.2 MySQL 的特点

（1）功能强大：MySQL 中提供了多种数据库存储引擎，其各有所长，适用于不同的应用场合，用户可以选择最合适的引擎以得到最高性能，可以处理每天访问量超过数亿的高强度的搜索 Web 站点。MySQL 支持事务、视图、存储过程、触发器等。

（2）支持跨平台：MySQL 支持至少 20 种以上的开发平台，包括 Linux、Windows、FreeBSD、IBMAIX、AIX 等。这使得在任何平台下编写的程序都可以进行移植，而不需要对程序做任何的修改。

（3）运行速度快：高速是 MySQL 的显著特性。在 MySQL 中，使用了极快的 B 树磁盘表（MyISAM）和索引压缩；通过使用优化的单扫描多连接，能够极快地实现连接；SQL 函数使用高度优化的类库实现，运行速度快。

（4）支持面向对象：PHP 支持混合编程方式。编程方式可分为纯粹面向对象、纯粹面向过程、面向对象与面向过程混合 3 种方式。

（5）安全性高：灵活和安全的权限与密码系统，允许基本主机的验证。连接到服务器时，所有的密码传输均采用加密形式，从而保证了密码的安全。

（6）成本低：MySQL 数据库是一种完全免费的产品，用户可以直接通过网络下载。

（7）支持各种开发语言：MySQL 为各种流行的程序设计语言提供支持，为它们提供了很多的 API 函数，包括 PHP、ASP. NET、Java、Eiffel、Python、Ruby、Tcl、C、C++、Pel 语言等。

（8）数据库存储容量大：MySQL 数据库的最大有效表尺寸通常是由操作系统对文件大小的限制决定的，而不是由 MySQL 内部限制决定的。InnoDB 存储引擎将 InnoDB 表保存在一个表空间内，该表空间可由数个文件创建，表空间的最大容量为 64TB，可以轻松处理拥有上千万条记录的大型数据库。

（9）支持大量的扩展库：MySQL 支持大量的扩展库，如 MySQLi 等，可以为快速开发 Web 应用提供便利。

11.2 数据库的常见操作

MySQL 数据库服务器的安装和维护相对简便，在 Linux/UNIX 操作系统中，使用终端命令即可实现对 MySQL 数据库服务器的管理，而在 Windows 操作系统中，不仅可以在命令提示符窗口中通过命令的方式管理 MySQL 服务器，还可以通过图形界面进行管理。下面将以 Windows 操作系统为例介绍如何对其进行启动、关闭等操作。

11.2.1 启动 MySQL 服务器

在 Window 操作系统中启动 MySQL 服务器的方法主要有两种：通过系统服务方式和在

命令提示符下通过命令方式。

1. 通过系统服务启动 MySQL 服务器

如果 MySQL 设置为 Windows 服务，则可以通过选择“开始”/“管理工具”/“服务”命令，打开 Windows 服务管理器，在服务器的列表中找到 mysql 服务，单击鼠标右键，在弹出的快捷菜单中选择“启动”命令，启动 MySQL 服务，如图 11-1 所示。

图 11-1　MySQL 启动服务界面

2. 在命令提示符下启动 MySQL 服务器

选择“开始”/“运行”命令，在弹出的“运行”窗口输入“cmd”命令，按 Enter 键进入命令提示符窗口。在命令提示符下输入：

```
net start mysql
```

按 Enter 键后，启动 MySQL 服务器，如图 11-2 所示。

图 11-2　MySQL 在控制台启动界面

11.2.2 连接和断开 MySQL 服务器

对 MySQL 数据库进行管理，首先要建立与 MySQL 服务器的连接。下面分别介绍连接和断开 MySQL 服务器的方法。

1. 连接 MySQL 服务器

连接 MySQL 服务器通过 mysql 命令实现。在 MySQL 服务器启动后，选择“开始”/“运行”命令，在弹出“运行”窗口中输入“cmd”命令，按 Enter 键后打开命令提示符窗口。在命令提示符下输入：

```
mysql -h 主机名 - u 用户名  -p 密码
```

输入完命令语句后，按 Enter 键即可连接 MySQL 服务器，如图 11-3 所示。

图 11-3 使用命令连接数据库

2. 断开 MySQL 服务器

连接到 MySQL 服务器后，可以通过在 MySQL 提示符下输入“exit”或者“quit”命令断开 MySQL 连接，如图 11-4 所示。

图 11-4 使用命令断开 MySQL 连接

11.2.3　停止 MySQL 服务器

为有效节省系统资源，在使用完 MySQL 服务后需要将其及时关闭。与启动 MySQL 服务类似，在 Windows 操作系统中，停止 MySQL 服务器主要有两种方法：通过系统服务关闭 MySQL 服务和在命令提示符下通过命令行的方式关闭 MySQL 服务。下面具体介绍这两种实现方式。

1. 通过系统服务停止 MySQL 服务器

如果使用的是安装版的 MySQL 数据库，则可以通过选择“开始”/“控制面板”/”管理工具“/“服务”命令，打开 Windows 服务管理器，在服务器的列表中找到 mysql 服务并右击，在弹出的快捷菜单中选择“停止”命令，停止 MySQL 服务器，如图 11-5 所示。

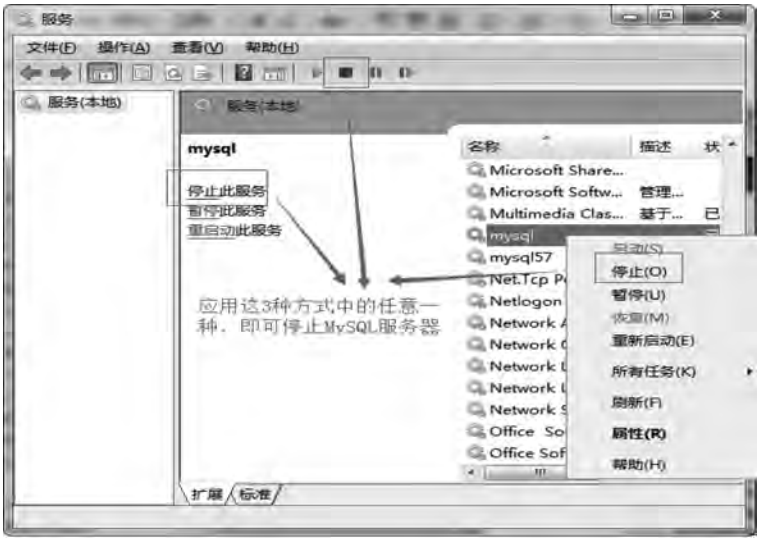


图 11-5　可视化停止 MySQL 服务

2. 在命令提示符下停止 MySQL 服务器

选择“开始”/“运行”命令，输入“cmd”命令，进入命令提示符窗口，在命令提示符下输入：

```
net　stop　mysql
```

按 Enter 键即可停止 MySQL 服务器，如图 11-6 所示。

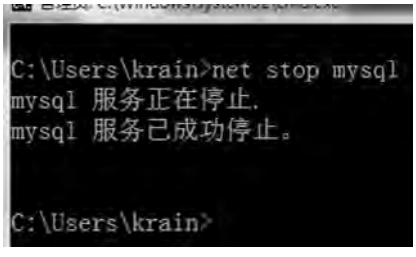


图 11-6　代码化停止 MySQL 服务

11.3 数据库的基础知识

在介绍数据库的管理及操作之前，这里首先介绍一些数据库最常用的术语和基本概念。只有明白了什么是数据库，为什么要使用数据库，使用数据库有什么好处，我们才能合理地运用数据库。

11.3.1 数据库的定义

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库。

每个数据库都有一个或多个不同的 API 用于创建、访问、管理、搜索和复制所保存的数据。

我们也可以将数据存储在文件中，但是在文件中读写数据速度相对较慢。

所以，现在我们使用关系型数据库管理系统（Relational Database Management System，RDBMS）来存储和管理大量数据。所谓的关系型数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。

RDBMS 的特点：

（1）数据以表格的形式出现。

（2）每行为各种记录名称。

（3）每列为记录名称所对应的数据域。

（4）许多的行和列组成一张表单。

（5）若干的表单组成数据库。

11.3.2 RDBMS 专业术语

在我们开始学习 MySQL 数据库前，先了解下 RDBMS 的一些术语：

数据库：数据库是一些关联表的集合。

数据表：表是数据的矩阵。在一个数据库中的表看起来像一个简单的电子表格。

列：一列（数据元素）包含了相同的数据，例如邮政编码的数据。

行：一行（=元组，或记录）是一组相关的数据，例如一条用户订阅的数据。

冗余：存储两倍数据，冗余降低了性能，但提高了数据的安全性。

主键：主键是唯一的。一个数据表中只能包含一个主键。你可以使用主键来查询数据。

外键：外键用于关联两个表。

复合键：复合键（组合键）将多个列作为一个索引键，一般用于复合索引。

索引：使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。类似于书籍的目录。

参照完整性：参照的完整性要求关系中不允许引用不存在的实体。参照完整性与实体完整性是关系模型必须满足的完整性约束条件，目的是保证数据的一致性。

MySQL 为关系型数据库，这种所谓的“关系型”可以理解为“表格”的概念，一个关系型数据库由一个或数个表格组成，见表 11-1。

表 11-1　学生表

学　号	姓　名	性别	出生时间	专　业	总学分	备　注
010301	余×胜	男	1998-3-1	计算机应用	42	有一门课不及格，待补考
010302	商×兴	男	1999-5-4	计算机应用	50	已提前修完一门课，并获学分
010303	徐　×	女	1999-6-10	计算机应用	46	
010304	邵×叶	男	1998-1-24	计算机应用	46	
010305	钱×滨	男	1998-2-20	计算机应用	46	
010306	张×如	男	1998-6-11	计算机应用	46	
010307	卢　×	男	1999-5-8	计算机应用	46	
010309	方×莎	女	1999-11-5	计算机应用	46	三好学生
010310	郑　×	男	1999-3-4	计算机应用	46	
010311	范×明	男	1998-12-10	计算机应用	46	
010312	徐　×	男	1998-5-14	计算机应用	42	有一门课不及格，待补考
270230	鲍×鑫	男	1998-3-7	软件技术	50	转专业的
020401	解×鹏	男	1998-11-20	软件技术	50	
020402	汪×霞	女	1999-6-7	软件技术	50	
020403	叶　×	女	1999-8-9	软件技术	50	
020404	余×鹏	男	1998-12-10	软件技术	50	
020405	徐×德	男	1998-9-8	软件技术	50	
020406	吕　×	男	1998-6-23	软件技术	50	有一门课不及格，待补考
020407	徐×浩	男	1998-12-5	软件技术	54	已提前修完一门课，并获学分
020408	方×飞	男	1998-8-20	软件技术	50	
020409	解×伟	男	1998-11-10	软件技术	50	三好学生
020410	方　×	男	1999-10-24	软件技术	54	已提前修完一门课，并获学分

说明：

表头（header）：每一列的名称。

列（col）：具有相同数据类型的数据的集合。

行（row）：每一行用来描述某条记录的具体信息。

值（value）：行的具体信息，每个值必须与该列的数据类型相同。

键（key）：键的值在当前列中具有唯一性。

11.4 数据库和表的建立与管理

数据库的管理和表的管理操作是数据库的基本管理操作，下面分别以可视化和命令两种方式进行详细说明。数据库的管理包括创建数据库、修改数据库和删除数据库。同样，数据表的管理也包括表的创建、表的修改和表的删除等操作。

11.4.1 数据库管理

1. 使用命令方式创建数据库

我们可以在登录 MySQL 服务后，使用 create 命令创建数据库，语法如下：

create DATABASE *数据库名*;

下面我们创建一个名为 student 的数据库，如图 11-7 所示。

```
C:\Users\krain>mysql -uroot -p123456
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25
Server version: 5.0.51b-community-nt-log MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create DATABASE student;
Query OK, 1 row affected (0.01 sec)

mysql>
```

图 11-7　使用命令方式创建数据库

2. 使用可视化方式创建数据库

在浏览器地址栏输入：http：//localhost/phpMyAdmin，再在弹出的对话框中输入用户名和密码，进入 phpMyAdmin 图形化管理界面，接下来就可以进行 MySQL 数据库的操作，如图 11-8 所示。

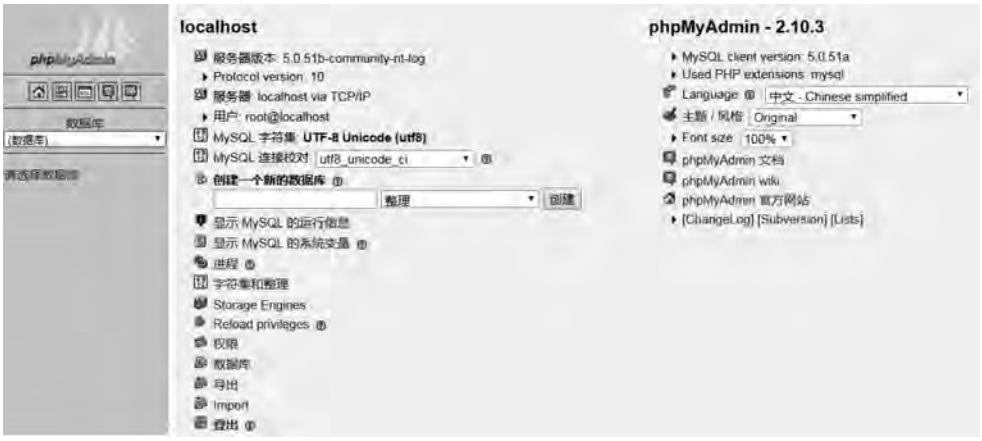


图 11-8　使用可视化方式创建数据库

在 phpMyAdmin 的主界面，首先在创建数据库的文本框填上数据库名字 student，然后在旁边的下拉框中选择所要使用的编码，一般选择 gb2312_Chinese_ci（简体中文编码格式），单击“创建”按钮，创建数据库。成功创建数据库后，将显示如图 11-9 所示信息。

图 11-9　成功创建数据库

3. 修改数据库

在数据库主界面中，在右侧界面还可以对当前数据库进行修改。单击界面中的 操作 超链接，进入修改操作页面。可以对当前的数据库重命名，在“重新命名数据库为”下的文本框中输入新的数据库名称，单击“执行”按钮，即可成功修改数据库名称，如图 11-10 所示。

图 11-10　重命名数据库

4. 删除数据库

要删除某个数据库，首先在左侧的下拉菜单中选择该数据库，然后单击右侧界面中的 删除 超链接即可成功删除指定的数据库。

如果采用命令删除数据库，格式如下：

```
drop database <数据库名>;
```

11.4.2　数据表管理

1. 使用命令创建数据表

创建 MySQL 数据表需要以下信息：

表名

表字段名

定义每个表字段

语法

以下为创建 MySQL 数据表的 SQL 通用语法：

```
CREATE TABLE table_name (column_name column_type);
```

以下例子中我们将在 RUNOOB 数据库中创建数据表 runoob_tbl：

```
CREATE TABLE IF NOT EXISTS 'runoob_tbl'(
    'runoob_id' INT UNSIGNED AUTO_INCREMENT,
    'runoob_title' VARCHAR(100) NOT NULL,
    'runoob_author' VARCHAR(40) NOT NULL,
    'submission_date' DATE,
    PRIMARY KEY ( 'runoob_id' )
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

实例解析：

如果你不想字段为 NULL，可以设置字段的属性为 NOT NULL，在操作数据库时如果输入该字段的数据为 NULL，就会报错。

AUTO_INCREMENT 定义列为自增的属性，一般用于主键，数值会自动加 1。

PRIMARY KEY 关键字用于定义列为主键。你可以使用多列来定义主键，列间以逗号分隔。

ENGINE 设置存储引擎，CHARSET 设置编码。

2. 使用可视化方式创建数据表

创建数据库 student 后，在右侧的操作页面中输入数据表的名称和字段数，然后单击“执行”按钮，即可创建数据表，如图 11-11 所示。

图 11-11　用可视化方式创建数据表

成功创建数据表 student 后，将显示数据表结构界面。在表单中对各个字段的详细信息进行输入，包括字段名、数据类型、长度值、编码格式、是否为空、主键等，以完成对表结构的详细设置。当所有的信息都输入后，单击“保存”按钮，创建数据表结构，如图 11-12 所示。

图 11-12　数据表的设计

成功创建数据表结构后，将显示结果如图 11-13 所示。

图 11-13　数据表

注：字段名称最好使用英文，笔者使用中文是为了使读者容易理解。

3. 修改数据表

一个新的数据表被创建后，进入到数据表页面中，在这里可以通过改变表的结构来修改表，可以执行添加新的列、删除列、索引列、修改列的数据类型或者字段的长度等操作，如图 11-14 所示。

图 11-14　数据表的修改

4. 删除数据表

要删除某个数据表，首先在左侧的下拉菜单中选择数据库，在指定的数据库中选择要删除的数据表，然后单击右侧界面中的 删除 超链接即可成功删除数据表。

11.5　数据表对象管理

对数据表的数据操作，无非就是对表数据进行增、删、改、查等操作，这些操作都可以用两种不同的形式实现，一种是用可视化操作完成，一种是用编写代码的形式完成。下面将详细说明对表的增加、删除、修改三种操作，查询操作将放在下一节再学习。

11.5.1　插入表数据

前面我们建立的 student 数据库中的数据表 student 现在还没有数据，那么现在就来给 student 数据表添加数据。

单击 phpMyAdmin 主界面中的 SQL 超链接，打开 SQL 语句编辑区。在编辑区输入完整的 SQL 语句来完成操作。

在 SQL 中，使用 insert 语句添加数据，如图 11-15 所示编辑完 insert 语句后，单击“执行”按钮，便向数据表中插入一条数据。

图 11-15　插入数据

如果提交的 SQL 语句有错误，系统会给出一个警告，提示用户修改它；如果提交的 SQL 语句正确，则弹出如图 11-16 所示信息。

按照上面的方法，把 stuent 表的数据全部添加完成，如图 11-16 所示。

←T→	学号	姓名	性别	出生时间	专业	总学分
	010301	余×胜	男	1998-03-01	计算机应用	42
	010302	商×兴	男	1999-05-04	计算机应用	50
	10303	徐 ×	女	0000-00-00	计算机应用	46
	10304	邵×叶	男	0000-00-00	计算机应用	46
	10305	钱×滨	男	0000-00-00	计算机应用	46
	10306	张×如	男	0000-00-00	计算机应用	46
	10307	卢 ×	男	0000-00-00	计算机应用	46
	10309	方×莎	女	0000-00-00	计算机应用	46
	10310	郑 ×	男	0000-00-00	计算机应用	46
	10311	范×明	男	0000-00-00	计算机应用	46
	10312	徐 ×	男	0000-00-00	计算机应用	42
	270230	鲍×鑫	男	0000-00-00	软件技术	50
	20401	解×鹏	男	2036-11-09	软件技术	50
	20402	汪×霞	女	0000-00-00	软件技术	50
	20403	叶 ×	女	0000-00-00	软件技术	50
	20404	余×鹏	男	0000-00-00	软件技术	50
	20405	徐×德	男	2036-04-06	软件技术	50
	20406	吕 ×	男	0000-00-00	软件技术	50
	20407	徐×浩	男	0000-00-00	软件技术	54
	20408	方×飞	男	2036-02-07	软件技术	50
	20409	解×伟	男	2036-10-09	软件技术	50
	20410	方 ×	男	0000-00-00	软件技术	54

图 11-16　插入数据成功

11.5.2 修改表数据

1. 使用 SQL 语句修改表数据

在 SQL 语句编辑区应用 update 语句修改数据信息，将方元的总学分修改为 60，语法为：

Update 表名 set 列名=60 where 列名='方元'。

其中 where 代表条件子句。

2. 使用可视化修改表数据

在浏览到的数据表上，在要修改的数据前面的选择框中打钩，然后点下面的 进入到修改界面，如图 11-17 所示。

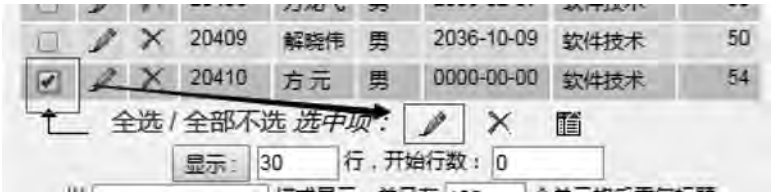


图 11-17 修改表数据

在修改界面上，作相应修改后，点击“执行”按钮就可以了，如图 11-18 所示。

图 11-18 修改表数据完成

11.5.3 删除表数据

1. 使用 SQL 语句删除表数据

在 SQL 语句编辑区应用 delete 语句删除指定条件的数据或全部数据信息，删除姓名为

“方元”的学生信息，添加的 SQL 语句语法为：

```
delete from 表名　where　列名='方元'
```

2. 使用可视化方式删除表数据

可视化删除表数据比较简单，只要在数据界面点 ✕ 就可以了，如图 11-19 所示。

图 11-19　删除数据

11.6　查 询 数 据

在 SQL 语句编辑区应用 select 语句检索指定条件的数据信息，将总学分少于 46 的学生信息全部显示出来，语法为：

```
select * from　表名　where　列名<46
```

单击“执行”按钮即可查询出结果，如图 11-20 所示。

←T→	学号	姓名	性别	出生时间	专业	总学分
✎ ✕	010301	余伟胜	男	1998-03-01	计算机应用	42
✎ ✕	10312	徐星	男	0000-00-00	计算机应用	42

图 11-20　查询数据

除了对整个表的简单查询外，还可以执行复杂的查询（使用 where 子句提交 LIKE、ORDER BY，GROUP BY 等条件查询语句）及多表查询，读者可通过上机进行实践，灵活运用 SQL 语句功能。

11.7　视　　图

视图是从一个或几个基本表（或视图）导出的表。它与基本表不同，是一个虚表。数据库中存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。所以基本表中的数据发生变化，从视图中查询出的数据也就随之改变了。从这个意义上讲，视图就像一个窗口，通过它可以看到数据库中自己感兴趣的数据及其变化。

视图一经定义，就可以和基本表一样被查询、被删除。也可以在一个视图之上再定义新的视图，但对视图的更新（增、删、改）操作则有一定的限制。

11.7.1 初识视图

1. 视图的定义

通俗地讲，视图就是一条 SELECT 语句执行后返回的结果集。所以我们在创建视图的时候，主要的工作就落在创建这条 SQL 查询语句上。

2. 视图的特性

（1）视图是对若干张基本表的引用，是一张虚表，用于查询语句执行的结果，不存储具体的数据（基本表数据发生了改变，视图也会跟着改变）；

（2）可以跟基本表一样，对视图进行增、删、改、查操作（增、删、改操作有条件限制）。

3. 视图的作用

（1）方便操作，特别是查询操作，减少复杂的 SQL 语句，增强可读性；

（2）更加安全，数据库授权命令不能限定到特定行和特定列，但是通过合理创建视图，可以把权限限定到行列级别。

4. 使用场合

（1）权限控制的时候，不希望用户访问表中某些含敏感信息的列。

（2）关键信息来源于多个复杂关联表，可以创建视图提取我们需要的信息，简化操作。

11.7.2 创建视图

创建视图的语法为：

```
CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    VIEW view_name [(column_list)]
    AS select_statement
  [WITH [CASCADED | LOCAL] CHECK OPTION]
```

说明：

（1）OR REPLACE：表示替换已有视图。

（2）ALGORITHM：表示视图选择算法。默认算法是 UNDEFINED（未定义的）：MySQL 自动选择要使用的算法；MERGE：合并；TEMPTABLE：临时表。

（3）select_statement：表示 select 语句。

（4）[WITH [CASCADED | LOCAL] CHECK OPTION]：表示视图在更新时保证在视图的权限范围之内。CASCADE 是默认值，表示更新视图的时候，要满足视图和表的相关条件；LOCAL 表示更新视图的时候，要满足该视图定义的一个条件即可。

推荐使用 WHIT [CASCADED | LOCAL] CHECK OPTION 选项，可以保证数据的安全性

图 11-21 所示为创建视图案例。

```
mysql> create view v_F_players(编号,名字,性别,电话)
    -> as
    -> select PLAYERNO,NAME,SEX,PHONENO from PLAYERS
    -> where SEX='F'
    -> with check option;
Query OK, 0 rows affected (0.00 sec)

mysql> desc v_F_players;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| 编号   | int(11)  | NO   |     | NULL    |       |
| 名字   | char(15) | NO   |     | NULL    |       |
| 性别   | char(1)  | NO   |     | NULL    |       |
| 电话   | char(13) | YES  |     | NULL    |       |
+--------+----------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> select * from  v_F_players;
+--------+-----------+--------+-------------+
| 编号   | 名字      | 性别   | 电话        |
+--------+-----------+--------+-------------+
|      8 | Newcastle | F      | 070-458458  |
|     27 | Collins   | F      | 079-234857  |
|     28 | Collins   | F      | 010-659599  |
|    104 | Moorman   | F      | 079-987571  |
|    112 | Bailey    | F      | 010-548745  |
+--------+-----------+--------+-------------+
5 rows in set (0.02 sec)
```

图 11-21　创建视图案例

11.7.3　查看视图

使用 show create view 语句查看视图信息，视图一旦创建完毕，就可以像一个普通表那样使用，视图主要用来查询

语法为：

select ＊ from 视图名字；

可以看出视图的查看和普通表的查看是一样的。

11.7.4　修改视图

1. 用 CREATE OR REPLACE VIEW 语句修改视图

基本格式：

create or replace view view_name as select 语句；

在视图存在的情况下可对视图进行修改，视图不在的情况下可创建视图。

2. 用 ALTER 语句修改视图

语法如下：

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
VIEW view_name [(column_list)]
AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

注意：修改视图是指修改数据库中已存在的表的定义，当基表的某些字段发生改变时，可以通过修改视图来保持视图和基本表之间一致。

11.7.5 删除视图

删除视图是指删除数据库中已存在的视图。删除视图时，只能删除视图的定义，不会删除数据，也就是说不动基表。

语法如下：

```
DROP VIEW [IF EXISTS]
view_name [, view_name]
```

如果视图不存在，则抛出异常；使用 IF EXISTS 选项使得删除不存在的视图时不抛出异常。

11.8 综 合 案 例

使用 SQL 语句在 MySQL 中的数据库 student 中创建好如表 11-2 所示课程表，表名为 course，并添加上数据。

表 11-2 课程表

课程号	课程名	开课学期	学 时	学 分
114	计算机数学	1	56	3.5
102	C 语言程序设计	1	72	4.5
209	JAVA 程序设计基础	3	64	4
104	SQL 数据库技术	3	64	4
101	网络编程 PHP	4	96	6
109	JSP 程序设计	4	64	4
103	网页设计与制作	3	64	4
210	数据结构	2	64	4
205	软件测试技术	5	64	4

创建 course 表的 SQL 语句为：

```
CREATE TABLE 'course' (
    '课程号' INT( 4 ) NOT NULL,
    '课程名' VARCHAR( 30 ) CHARACTER SET gb2312 COLLATE gb2312_chinese_ci NOT NULL,
    '开课学期' INT( 4 ) NOT NULL,
    '学时' INT( 4 ) NOT NULL,
    '学分' FLOAT( 4 ) NOT NULL,
    PRIMARY KEY ('课程号')
) ENGINE = MYISAM;
```

添加数据的语句为：

```
insert into course values(114,'计算机数学',1,56,3.5);
insert into course values(102,'C 语言程序设计',1,72,4.5);
insert into course values(209,'JAVA 程序设计基础',3,64,4);
insert into course values(104,'SQL 数据库技术',3,64,4);
insert into course values(101,'网络编程 PHP',4,96,6);
insert into course values(109,'JSP 程序设计',4,64,4);
insert into course values(103,'网页设计与制作',3,64,4);
insert into course values(210,'数据结构',2,64,4);
insert into course values(205,'软件测试技术',5,64,4);
```

本章小结

本章对 MySQL 数据库的基本概念、MySQL 常见操作进行了介绍，并详细介绍了 Windows 系统下使用“命令提示符”窗口创建和维护 MySQL 数据库和数据表的方法。通过本章的学习，读者能够了解 MySQL 数据库的基本操作和维护方法，掌握 MySQL 数据库中最基本和最常用命令的语法格式，并能够具备基本管理和维护 MySQL 数据库的能力。

习　题

一、简答题

1. 什么是视图？视图的作用是什么？

2. 试述视图的优点。

3. 数据表的基本操作有哪些，分别使用 SQL 语言哪些关键字？

二、操作题

请使用 MySQL 在本章前面建立的 student 数据库中添加 score 表（见表 11-3），并添加相应的数据，添加数据后，将每个同学的成绩扣掉 10 分。

表 11-3　score 表

学　号	课程号	成　绩	学　号	课程号	成　绩	学　号	课程号	成　绩
010301	114	65	010305	114	65	020405	114	80
010301	102	75	010305	102	80	020405	102	63
010302	114	90	010305	210	68	020405	210	79
010302	102	88	270230	114	85	020407	114	675
010302	210	90	270230	102	78	020407	102	80
010302	205	80	270230	210	87	020407	210	65
010303	114	80	020402	114	66	020409	114	87
010303	102	82	020402	102	86	020409	102	91
010303	210	65	020402	210	70	020409	210	95
010309	114	89	020403	114	75	020406	114	65
010309	102	95	020403	102	80	020406	102	67
010309	210	90	020403	210	85	020410	114	95
010311	114	71	020404	114	76	020410	102	89
010311	102	80	020404	102	78	020410	210	90
010311	210	78	020404	210	85	020410	205	90
010301	210	50	020406	210	56	020407	205	78

第 12 章

PHP 操作 MySQL 数据库

【本章内容简介】

大多数网站中的数据都存储在数据库中，所以任何一种编程语言都不可避免地要对数据库进行操作。PHP 支持对多种数据库的操作，并提供了相关的数据库连接函数和操作函数。本章主要介绍 PHP 访问 MySQL 数据库的流程和具体方法。

【本章学习要求】

通过本章的学习，要求了解 PHP 访问 MySQL 数据库的一般流程，掌握 PHP 对数据库进行插入、修改、删除、查询等操作函数的使用。

12.1 PHP 访问 MySQL 数据库的一般流程

通过上一章的学习，相信读者已经对 MySQL 数据库有了一定认识。本节主要介绍 PHP 访问 MySQL 数据库的一般流程。

使用 PHP 访问 MySQL，必须让 PHP 程序先能连接 MySQL 数据库服务器，再选择一个数据库作为默认操作的数据库，然后才能向 MySQL 数据库管理系统发送 SQL 语句，其基本流程如图 12-1 所示。

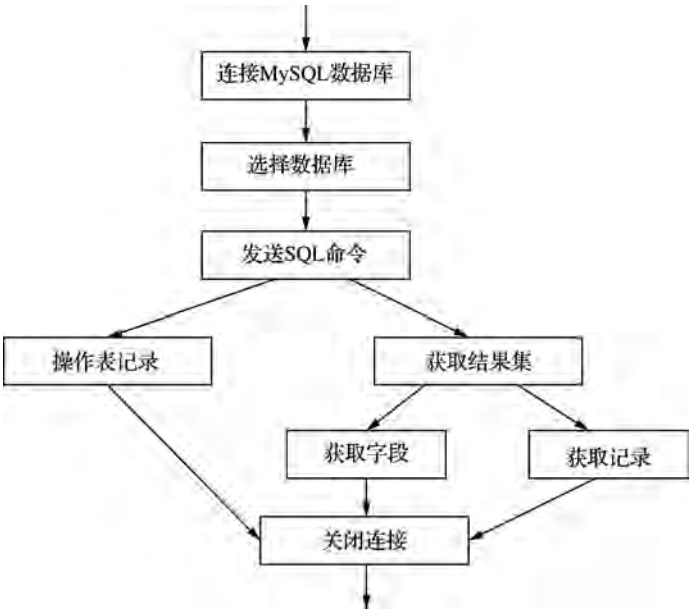


图 12-1 PHP 访问 MySQL 数据库流程

如果发送的是类似 INSERT、UPDATE 或 DELETE 等 SQL 语句，MySQL 执行完

成并对数据表的记录有所影响，说明执行成功。如果发送的是类似 SELECT 这样的 SQL 语句，会返回结果集，还需要对结果集进行处理。处理结果集又包括获取字段信息和获取记录数据两种操作，而多数情况下只需要获取记录数据即可。脚本执行结束后还需要关闭本次连接。

12.2 PHP 访问 MySQL 数据库的具体方法

PHP 中提供了很多操作 MySQL 数据库的函数，利用这些函数，可以完成对 MySQL 数据库的各种操作。PHP 5.0 之后的版本，PHP 推荐使用 mysqli 系列函数来操作 MySQL 数据库，本书主要介绍 mysqli 系列函数。

12.2.1 连接 MySQL 数据库

在操作 MySQL 数据库之前，首先要确保已成功连接 MySQL 数据库。连接 MySQL 数据库服务器常用的函数是 mysqli_connect（ ）。函数的语法格式如下：

mysqli_connect(string $host,string $username,string $password,string $database)

函数功能：通过 PHP 程序连接 MySQL 数据库服务器。

如果连接 MySQL 数据库服务器成功，则函数返回值为一个 MySQL 服务器连接标识（Link_identifier），否则返回值为 FALSE。

函数的参数说明见表 12-1。

表 12-1 mysqli_connect（ ）函数的参数说明

参 数	说 明
$host	MySQL 数据库服务器的 IP 地址或主机名，默认端口值为 3306（常省略）
$username	连接 MySQL 数据库服务器的用户名
$password	连接 MySQL 数据库服务器的密码
$database	所要连接的 MySQL 数据库名称

【例 12.2-1】 实现连接本地 MySQL 数据库服务器（假设 MySQL 数据库服务器中已创建数据库 school）。

程序如下：

```
<? php
    $host = "localhost";              //MySQL 数据库服务器的地址
    $username = "root";               //登录数据库服务器的用户名
    $password = "";                   //登录数据库服务器的密码
    $database = "school";             //登录的数据库名称
    $conn = mysqli_connect( $host, $username, $password, $database);
? >
```

首先使用 mysqli_connect_errno（ ）函数判断连接 MySQL 服务器是否成功，若不成功，

则浏览器上会显示“Warning：mysqli_connect（ ）…”的字样，此时可以使用 mysqli_connect_error（ ）函数提取 mysqli_connect（ ）函数的错误信息；如果连接成功，则 mysqli_connect_error（ ）函数返回空字符串。例 12.2-1 的 PHP 程序可以写为：

```
<? php
    $host = "localhost";
    $username = "root";
    $password = "";
    $database = "school";
    $conn = mysqli_connect( $host, $username, $password, $database);
    if( mysqli_connect_errno( ))
        die("MySQL 数据库连接失败!".mysqli_connect_error( ));
? >
```

说明：mysqli_connect_errno（ ）函数返回上一次连接错误的错误代码值，如果没有错误发生则返回 0；mysqli_connect_error（ ）函数返回上一次连接错误的错误描述，如果没有错误发生则返回空字符串；die（ ）函数用于在操作失败时输出一条信息。

12.2.2　设置数据库字符集

我们在执行 PHP 程序显示数据库的内容时，有时会发现在浏览器页面上本来应该显示的中文字符却变成了一堆乱码，这是因为 MySQL 数据库、PHP 程序、HTML 页面以及浏览器所使用的字符集不一致造成的。MySQL 数据库默认使用的字符集是 UTF-8，我们使用的 IE 浏览器默认使用的字符集是简体中文字符集 GB2312，有时候 HTML 页面使用的也是 GB2312 字符集，这就涉及 PHP 程序中编码的转换问题。

在 PHP 程序里需要将 MySQL 数据库的字符集 character_set_database、客户端字符集 character_set_client、数据库连接字符集 character_set_connection 和结果字符集 character_set_result 设置为一个统一的简体中文字符集，这样才能避免在浏览器页面上出现乱码的现象。

一般为了便于 PHP 程序调试、运行，字符集可以统一设置为 GB2312、GBK 或 UTF-8。调用 PHP 函数 mysqli_query（ ）就可以将 character_set_database、character_set_client、character_set_connection 和 character_set_result 的字符集设置为所需的字符集。mysqli_query（ ）函数的调用位置一般位于连接 MySQL 数据库的命令之后，调用的命令格式如下：

```
mysqli_query( $conn,"set names gb2312");
```

为了方便连接 MySQL 数据库，可以将例 12.2-1 的程序单独写成一个 PHP 程序文件（比如命名为 connect_mysql.php），需要连接 MySQL 数据库时，就在 PHP 程序文件中使用 include（ ）函数或 include_once（ ）函数调用该程序文件。connect_mysql.php 程序文件完整的内容如下：

```
<? php
    global  $conn;                          //数据库连接标识符
    $host = "localhost";                    //MySQL 服务器地址
```

```
    $username="root";                    //登录 MySQL 服务器的用户名
    $password="";                        //登录 MySQL 服务器的密码
    $database="school";                  //要操作的数据库
    $conn=mysqli_connect( $host, $username, $password, $database);
    mysqli_query( $conn, "set names gb2312");
    if(mysqli_connect_errno( $conn))
        die("MySQL 服务器连接失败!".mysqli_connect_error());
? >
```

12.2.3 执行 SQL 语句

在连接 MySQL 服务器、设置字符集后，就可以在 PHP 程序中向 MySQL 服务器发送 SQL 语句或 MySQL 命令了。在 PHP 程序中常常采用 mysqli_query（）函数向 MySQL 服务器发送操作命令，语法格式如下：

```
mysqli_query(resource $conn, string $query[, int $resultmode])
```

函数的参数说明见表 12-2。

表 12-2 mysqli_query（）函数的参数说明

参 数	说 明
$conn	MySQL 服务器的连接 ID
$query	向 MySQL 服务器发送的 SQL 语句或 MySQL 命令
$resultmode	可选。是一个常量，可以是 MYSQLI_USE_RESULT（适用于大量数据的检索）或 MYSQLI_STORE_RESULT（默认）

函数的返回值：针对成功的 select、show、describe 或 explain 查询，将返回一个 mysqli_result 对象集。其他操作（如 delete、update、insert 等 SQL 语句）若成功，将返回 TRUE；如果失败，则返回 FALSE。

1. 发送 insert、update 或 delete 语句

使用 mysqli_query（）函数向 MySQL 服务器发送 insert、update 或 delete 命令语句后，可以使用 mysqli_affected_rows（）函数查看 SQL 语句影响的表记录行数。mysqli_affected_rows（）函数的语法格式如下：

```
mysqli_affected_rows(resource $connection);
```

函数功能：取得函数最近一次与表操作所影响的记录行数。

【例 12.2-2】 school 数据库的 student 表结构见表 12-3，原有数据见表 12-4，现要求向 student 表中插入以下一行数据：

010304，邵方叶，计算机应用 3 班，80

表 12-3　student 表的结构

字段名称	字段类型及长度	字段含义
xh	varchar，20	学号
xm	varchar，20	姓名
bj	varchar，20	班级
cj	int	成绩

表 12-4　student 表的数据

学号（xh）	姓名（xm）	班级（bj）	成绩（cj）
010301	余伟胜	计算机应用 3 班	65
010302	商振兴	计算机应用 3 班	90
010303	徐　丽	计算机应用 3 班	80
010305	钱江滨	计算机应用 3 班	77
010306	张静如	计算机应用 3 班	68
010307	卢　鉴	计算机应用 3 班	88
010309	方莉莎	计算机应用 3 班	98
010310	郑　聪	计算机应用 3 班	95
010311	范小明	计算机应用 3 班	71
010312	徐　星	计算机应用 3 班	55
270230	鲍曙鑫	软件技术 2 班	86
020401	解伟鹏	软件技术 4 班	65
020402	汪晓霞	软件技术 4 班	78
020403	叶　琦	软件技术 4 班	85
020404	余江鹏	软件技术 4 班	80
020405	徐明德	软件技术 4 班	90
020406	吕　涛	软件技术 4 班	58
020407	徐立浩	软件技术 4 班	88
020408	方龙飞	软件技术 4 班	95
020409	解晓伟	软件技术 4 班	90
020410	方　元	软件技术 4 班	78

程序如下：

```
<? php
    include_once("connect_mysql.php");
    header("Content-Type:text/html;charset=gb2312");
     $sql="insert into student(xh,xm,bj,cj) values('010304','邵方叶','计算机应用 3 班',80)";
     $result=mysqli_query( $conn, $sql);
    if( $result)
```

```
        echo "插入记录的行数:".mysqli_affected_rows( $conn);
    else
        echo"数据插入失败!";
? >
```

【例 12.2-3】将 student 表中学号为 270230 的学生的班级改为“软件技术 4 班”。

程序如下:

```
<? php
    include_once("connect_mysql.php");
    header("Content-Type:text/html;charset=gb2312");
     $s_xh='270230';
     $s_bj='软件技术 4 班';
     $sql="update student set bj= ' $s_bj' where xh=' $s_xh'";
     $result=mysqli_query( $conn, $sql);
    if( $result)
        echo "记录修改成功!";
    else
        echo "记录修改不成功!";
? >
```

【例 12.2-4】删除 student 表中学号为 020410 的学生记录。

程序如下:

```
<? php
    include_once("connect_mysql.php");
    header("Content-Type:text/html;charset=gb2312");
     $s_xh="020410 ";
     $sql="delete from student where xh=' $s_xh'";
     $result=mysqli_query( $conn, $sql);
    if( $result)
        echo "记录删除成功!";
    else
        echo "记录删除不成功!";
? >
```

2. 发送 select 语句

使用 mysqli_query（ ）函数向 MySQL 服务器发送 select 命令语句后，mysqli_query（）函数将得到一个结果集（ $result）数据，此时可以使用 mysqli_num_rows（ ）函数查看该结果集的记录行数。mysqli_num_rows（ ）函数的语法格式如下:

```
int mysqli_num_rows(resource  $result);
```

函数的参数说明见表 12-5。

表 12-5　mysqli_num_rows（ ）函数的参数说明

参　数	说　明
$result	由 mysqli_query（）返回的结果集标识符

【例 12.2-5】查询 student 表的记录数。

程序如下：

```
<? php
    include_once("connect_mysql.php");
    header("Content-Type:text/html;charset=gb2312");
    $sql="select * from student";
    $result=mysqli_query( $conn, $sql);
    $row_count=mysqli_num_rows( $result);
    echo "记录数:". $row_count;
? >
```

12.2.4　遍历结果集

在使用 mysqli_query（ ）函数取得查询的结果集后，就可以使用 mysqli_fetch_row（ ）函数、mysqli_fetch_array（ ）函数和 mysqli_fetch_object（ ）函数遍历结果集中的数据。

1. mysqli_fetch_row（ ）函数

函数的语法格式如下：

```
array mysqli_fetch_row(resource $result);
```

函数的参数说明见表 12-6。

表 12-6　mysqli_fetch_row（ ）函数的参数说明

参　数	说　明
$result	由 mysqli_query（）函数返回的结果集标识符

函数功能：从结果集 $result 中获取一行记录，并将该行记录生成一个数组，数组元素的键从 0 开始，数组元素的值依次为 select 语句中“字段列表”的值。依次调用 mysqli_fetch_row（ ）函数将返回结果集的下一行记录，若结果集 $result 中没有更多行记录则返回 FALSE。

【例 12.2-6】利用 mysqli_fetch_row（ ）函数显示 student 表中的所有记录。

程序如下：

```
<? php
    include_once("connect_mysql.php");
    header("Content-Type:text/html;charset=gb2312");
    $sql="select * from student";
    $result=mysqli_query( $conn, $sql);
```

```
while( $s=mysqli_fetch_row( $result) ){
    echo  $s[0]."";              //显示字段 xh 的值
    echo  $s[1]."";              //显示字段 xm 的值
    echo  $s[2]."";              //显示字段 bj 的值
    echo  $s[3]."";              //显示字段 cj 的值
    echo "<br/>";                //输出换行符
}
? >
```

运行结果如图 12-2 所示。

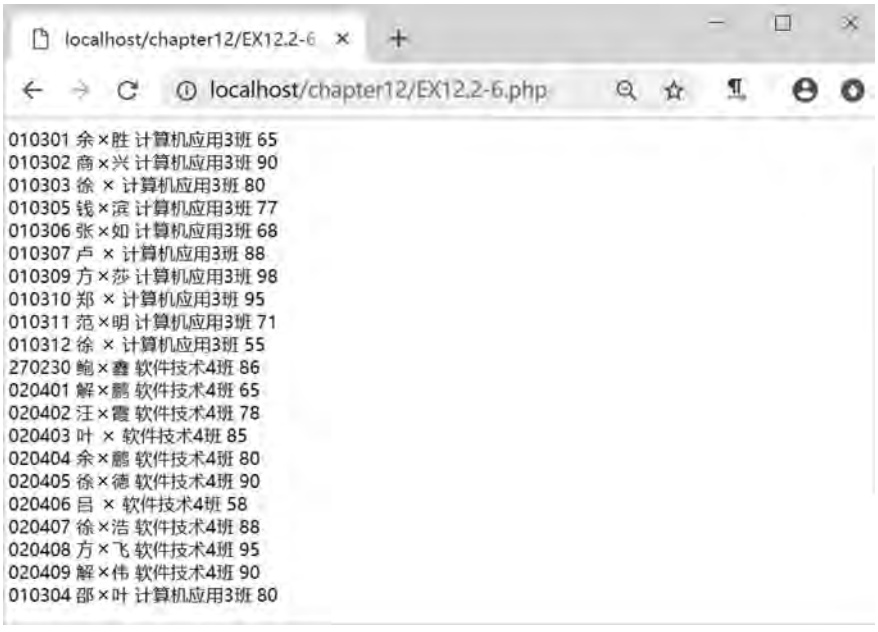


图 12-2　遍历结果

2. mysqli_fetch_array（ ）函数

函数的语法格式如下：

```
array mysqli_fetch_array(resource  $result[ ,int  $result_type])
```

函数的参数说明见表 12-7。

表 12-7　mysqli_fetch_array（）函数的参数说明

参　数	说　明
$result	由 mysqli_query（）返回的结果集标识符
$result_type	可选。规定产生数组的类型，可以是以下值中的一个。 MYSQLI_ASSOC：数组索引形式； MYSQLI_NUM：数字索引数组形式； MYSQLI_BOTH：以上二者都可以

函数功能：该函数是 mysqli_fetch_row（ ）的扩展，函数的返回值除了以数字作为键名储存在数组中外，还使用字段名作为键名储存。

【例 12.2-7】利用 mysqli_fetch_array（ ）函数显示 student 表中的记录。

程序如下：

```
<? php
    include_once("connect_mysql.php");
    header("Content-Type:text/html;charset=gb2312");
     $sql="select * from student";
     $result=mysqli_query( $conn, $sql);
    while( $s=mysqli_fetch_array( $result)){
        echo $s['xh']."";                    //显示字段 xh 的值
        echo $s['xm']."";                    //显示字段 xm 的值
        echo $s['bj']."";                    //显示字段 bj 的值
        echo $s['cj']."";                    //显示字段 cj 的值
        echo "<br/>";
    }
? >
```

运行结果如图 12-2 所示。

3. mysqli_fetch_object（ ）函数

函数的语法格式如下：

```
array mysqli_fetch_object(resource $result,[,string $classname[, array $params]])
```

函数的参数说明见表 12-8。

表 12-8　mysqli_fetch_object（ ）函数的参数说明

参　数	说　明
$result	由 mysqli_query（）返回的结果集标识符
$classname	可选。规定要实例化的类名称，设置属性并返回
$params	可选。规定一个传给 classname 对象构造器的参数数组

函数功能：该函数从结果集中取得一行数据，并作为对象返回。若结果集 result 中没有记录，则函数的返回值为 FALSE。

【例 12.2-8】利用 mysqli_fetch_object（ ）函数显示 student 表中的记录。

程序如下：

```
<? php
    include_once("connect_mysql.php");
    header("Content-Type:text/html;charset=gb2312");
     $sql="select * from student";
     $result=mysqli_query( $conn, $sql);
```

```
while( $s=mysqli_fetch_object( $result)){
    echo $s->xh. "";              //显示字段 xh 的值
    echo $s->xm. "";              //显示字段 xm 的值
    echo $s->bj. "";              //显示字段 bj 的值
    echo $s->cj. "";              //显示字段 cj 的值
    echo "<br/>";
}
? >
```

运行结果如图 12-2 所示。

12.2.5 关闭与 MySQL 数据库的连接

由于 Web 系统中的 PHP 程序需要经常和 MySQL 服务器进行交互，而数据库的连接又是非常宝贵的系统资源，当用户与 MySQL 服务器的连接超过一定数量时，会导致系统性能的下降，甚至死机，因此操作 MySQL 数据库完毕，最好立即关闭与 MySQL 服务器的连接。常用的函数有 mysqli_free_result（ ）和 mysqli_close（ ）。

1. mysqli_free_result（ ）函数

语法格式如下：

```
mysqli_free_result(resource $result)
```

函数的参数说明见表 12-9。

表 12-9 mysqli_free_result（ ）函数的参数说明

参 数	说 明
$result	由 mysqli_query（ ）返回的结果集标识符

函数功能：释放结果集所占用的内存，该函数无返回值。

2. mysqli_close（ ）函数

语法格式如下：

```
mysqli_close(resource $conn)
```

函数的参数说明见表 12-10。

表 12-10 mysqli_close（ ）函数的参数说明

参 数	说 明
$conn	要关闭的 MySQL 连接

函数的功能：关闭与 MySQL 服务器的连接。

函数的返回值：如果关闭成功返回 TRUE，否则将返回 FALSE。

12.3　综 合 案 例

本节主要介绍在 PHP 程序中实现分页显示数据的方法。在 PHP 程序中，如果需要显示的记录很多，而在一个页面上无法显示全部记录，这时就需要分页显示，即将记录分页来显示。

分页是一种将信息分段展示给浏览器用户的技术。浏览器用户每次看到的不是全部信息，只是其中的一部分信息，用户可以指定页码或翻页的方式来找到自己想要的内容。

1. 分析

创建数据库 school，字符集采用 utf8_unicode_ci；创建表 info，表结构见表 12-11。然后导入表 12-12 的数据。

表 12-11　info 表结构

字　段	类型、长度
学号	varchar（20）
姓名	varchar（20）

表 12-12　info 表数据

学　号	姓　名
1616020301	陈×林
1616020302	陈×辉
1616020303	陈×威
1616020304	陈×铭
1616020305	陈×龙
1616020306	陈×城
1616020307	陈×宇
1616020308	洪×涛
1616020447	杨×松
1616020448	杨×浪
1616020449	杨×坤
1616020450	杨×林
1616020451	杨×成
1616020452	叶×龙
1616020453	尹　×
1616010101	蔡×楷
1616010102	蔡×伶
1616010103	陈×山

续表

学　号	姓　名
1616010104	陈×滔
1616010105	陈×强
1616010106	邓×财
1616010107	冯×茂
1616010219	李×燕
1616010220	梁×海
1616010221	梁×欣
1616010222	梁×威
1616010223	梁×晶
1616010224	梁×深
1616010225	梁×彬
1616010226	梁×伟

2. 分页显示的原理

所谓分页显示，就是将数据表中的结果集人为地分成一段一段来显示，工作流程如下。

（1）获取结果集的记录总条数。

要显示的记录总数（$count），可以使用以下 SQL 语句获得：

```
$sql="select * from info";
$result=mysqli_query($con, $sql);
$count=mysqli_num_rows($result);
```

（2）算出总页数 $pagecount。

每页显示的记录的数量（$pagesize）：$pagesize=5;

```
if ($count% $pagesize==0) $pagecount=$count/$pagesize;
else $pagecount=(int)($count/$pagesize+1);
```

（3）规定当前要显示的页数 $currentpage。

首先显示结果集的第 1 页，当用户单击“首页”“上一页”“下一页”“尾页”的超链接时，就显示相应的页数。

```
$page=@ $_GET["page"]
if ($page = null) $currentpage=1;
else $currentpage=intval($page)
```

（4）显示当前页的所有记录。

3. 实现过程

打开浏览器，在地址栏中输入 http://localhost/chapter12/EX12.3-1.php，程序运行结果如图 12-3 所示。

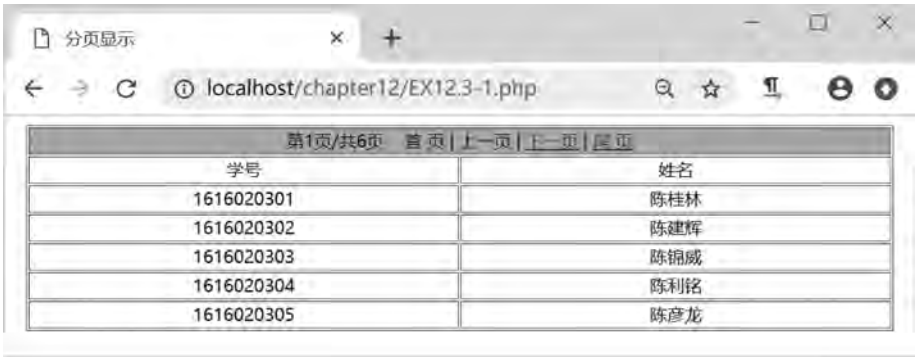


图 12-3　分页显示效果

4. 编程实现

新建 EX12.3-1.php 文件，输入以下代码：

```
<html>
<head>
<title>分页显示</title>
</head>
<body>
<? php
 $con=mysqli_connect("localhost","root","root","school");
mysqli_query( $con,"SET names gb2312");
 $sql="select * from info";
 $result=mysqli_query( $con, $sql);
 $count=mysqli_num_rows( $result);                    //记录总条数 $count。
 $pagesize=5;                                         //每页要显示的记录条数 $pagesize
if ( $count% $pagesize==0)  $pagecount= $count/ $pagesize;
else  $pagecount=(int)( $count/ $pagesize+1);         //总页数 $pagecount
 $row=mysqli_fetch_array( $result,MYSQL_NUM);         //数组 $row 的键名为字段名
 $page=@ $_GET["page"];                               //欲显示的页数 $page
if ( $page==null)  $currentpage=1;else  $currentpage=intval( $page);
for( $i=1; $i<=( $currentpage-1) * $pagesize; $i++)   //指定每一页面显示 5 条记录
{ if (! $row) break;
   $row=mysqli_fetch_array( $result,MYSQL_NUM);
}
? >
<table width="800" border="1" align="center" cellpadding="0">
  <tr>
    <td height="25" colspan="2" align="center" bgcolor="#99CCCC">第 <? php echo $currentpage;? >页/共<? php echo $pagecount;? >页</font>   
```

```
    <? php
    if ( $currentpage==1) echo "首 页 | 上一页 | ";
    else
    {? >
        <a href="EX12.3-1.php? page=1">首 页</a> |
        <a href="EX12.3-1.php? page=<? php echo $currentpage-1;? >">上一页</a> |
    <? php
    }
    if ( $currentpage== $pagecount) echo "下一页 | 尾 页";
    else
    {? >
        <a href="EX12.3-1.php? page=<? php echo $currentpage+1;? >">下一页</a> |
        <a href="EX12.3-1.php? page=<? php echo $pagecount;? >">尾 页</a>

    <? php
    }
    ? ></td>
</tr>
<tr>
    <td width="400" height="25" align="center">学号</td>
    <td width="400" height="25" align="center">姓名</td>
  </tr>
<? php
        for( $i=1; $i<= $pagesize; $i++)
        { if (! $row) break;
    ? >
    <tr>
      <td width="400" height="25" align="center"><? php echo $row[0];? ></td>
      <td width="400" height="25" align="center"><? php echo $row[1];? ></td>
    </tr>
<? php
        $row=mysqli_fetch_array( $result,MYSQL_NUM);
      }
    ? >
</table>
</body>
</html>
```

本章小结

本章主要介绍了 PHP 操作 MySQL 数据库的方法。在学完本章内容后，读者应重点掌握以下知识。

（1）PHP 访问 MySQL 数据库的一般流程为：连接 MySQL 服务器→选择 MySQL 数据库→执行 SQL 语句→关闭 MySQL 连接。

（2）连接 MySQL 服务器，一般使用 mysqli_connect（ ）函数。

（3）要对数据库中的表执行操作，就要使用 mysqli_query（ ）函数执行 SQL 语句。

（4）处理查询结果集，常用函数 mysqli_fetch_array（ ）、mysqli_fetch _object（ ）、mysqli_fetch_row（ ）和 mysqli_num_rows（ ）。

习　题

一、填空题

1. 使用_____函数，可以与 MySQL 服务器建立连接，并设置默认数据库。

2. 要修改连接 MySQL 服务器时定义的默认 MySQL 数据库，或者要选择服务器上的数据库，可以使用_____函数。

3. 要对数据库中的表执行操作，就要使用_____函数执行 SQL 语句。

4. 使用_____函数可以从查询结果集中取得一行作为关联数组、数字数组，或二者兼有。

5. 使用_____函数可以从查询结果集中取得当前行，并将其作为对象返回。

6. 使用_____函数，可以逐行获取查询结果集中的每条记录。

7. 使用_____函数，可以获取查询结果集中的记录数。

二、简答题

简述 PHP 访问 MySQL 数据库的一般流程。

第 13 章

实 训 指 导

实训 1　PHP 运行和开发环境

【实训目的】

了解 Web 应用程序的工作原理，掌握 phpStudy 集成环境的安装，掌握 Apache 服务器的相关配置，学会 EclipsePHP Studio 等 PHP 开发环境的使用，学会简单的 PHP 程序的编写。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

1. 启动 Apache

启动 phpStudy 控制面板，单击“启动”按键，左侧提示信息显示 Apache，MySQL 已经启动，如图 13-1 所示。

图 13-1　phpStudy 控制面板

2. 编辑 PHP 代码

打开 PHP 程序编辑器（如写字板、记事本、EclipsePHP Studio 或 Zend Studio 等），在新创建的文件中输入代码：

```
<? php
    echo "* * * * * * * * * * * * * * * * * *","<br/>";
    echo "* 你好! PHP *","<br/>";
    echo "* * * * * * * * * * * * * * * * * *";
? >
```

说明：

程序中的
是 HTML 标记符号，表示换行的意思。

将文件保存到 phpStudy 安装文件夹下的 C：\ phpStudy \ PHPTutorial \ WWW 文件夹下，文件的扩展名为 . php，如 example_1. php。

3. 运行 PHP 程序

启动浏览器（如 IE），在地址栏中输入 http：//localhost/example_1. php 或者 http：//127. 0. 0. 1/example_1. php 后回车，在浏览器中会看到输出的字符串“你好！PHP”，程序运行效果如图 13-2 所示。

图 13-2　程序运行结果

提示：有时在浏览器中看到的结果为乱码，其原因是由于 PHP 程序使用的字符集与浏览器使用的字符集不匹配造成的。此时需要将浏览器使用的字符集和 PHP 程序使用的字符集改为统一的简体中文字符集（GB2312 或 GBK）。

方法一：在 PHP 程序的开始部位添加语句。

```
header("Content-Type:text/html;charset=gb2312");
```

上述 PHP 程序可以写为：

```
<? php
header("Content-Type:text/html;charset=gb2312");
echo "* * * * * * * * * * * * * * * * * *","<br/>";
echo "* 你好! PHP *","<br/>";
echo "* * * * * * * * * * * * * * * * * *";
? >
```

方法二：打开 IE 浏览器的菜单项“查看”|“编码”，将浏览器的编码改为“简体中文（GB2312）”即可。

实训 2　PHP 语法基础

【实训目的】

（1）掌握 PHP 与 HTML 交互方法。

（2）掌握 PHP 中变量和常量的定义。

（3）掌握 PHP 中运算符的使用方法。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

获取登录表单中的用户名和密码，如果用户名为“admin”，密码为“123456”，则提示登录成功。

新建 login. php 文件，输入以下代码：

```
<html>
<body>
<form action="" method="post">
<table border="1" align="center" cellpadding="0" cellspacing="0">
<tr><td colspan="2" align="center" bgColor="#999999">用户登录表单</td></tr>
<tr><td>用户名:</td><td><input name="username" type="text"></td></tr>
<tr><td>密码:</td><td><input name="password" type="password" size="21"></td></tr>
<tr><td colspan="2" align="center">
<input type="submit" name="submit" value="登录">
<input type="reset" name="submit2" value="重置"></td></tr>
</table>
</form>
</body>
</html>
<? php
if( isset( $_POST['submit'] ) )
{
    $username= $_POST['username'];
    $password= $_POST['password'];
    if( $username==""admin"&& $password==""123456" )
    {
        echo "<script>alert('登录成功');</script>";
```

```
    }
    else
    {
        echo "<script>alert('登录失败');</script>";
    }
}
? >
```

程序运行结果如图 13-3 所示。

图 13-3　程序运行结果

实训 3　PHP 流程控制结构

【实训目的】

（1）熟悉 HTML 标签的使用。

（2）熟悉表单的提交方法。

（3）掌握常用的循环语句。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

（1）计算从 1 开始到指定数字的累加和，指定数字由用户输入。

新建文件 calc. php 文件，输入以下代码：

```
<html>
<body>
<div><font size="4" color="blue" face="黑体">计算累加器</font></div>
<form action="" method="post">
1+2+…+<input type="text" name="num" size="5">
```

```
<input type="submit" name="submit" value="计算">
</form>
</body>
</html>
<? php
if( isset( $_POST['submit'] ))
{
    $num= $_POST['num'];
    $i=0;
    $sum=0;
    if( $num<=2)
        echo "<script>alert('输入的数字必须大于2');</script>";
    else
    {
        while( $i<= $num)
        {
            $sum= $sum+ $i;
            $i++;
        }
        echo "<script>alert('1+2+...+ $num= $sum');</script>";
    }
}
? >
```

程序运行结果如图 13-4 所示。

图 13-4　程序运行结果

(2) 计算器练习。

新建一个 Calculator. php 页面，输入以下代码：

```
<html>
<head>
    <title>我的计算器</title>
    <meta http-equiv="content-type" content="text/html;charset=bg2312"/>
</head>
<body>
<form action="CalResult.php" method="post">
```

```
    <table width="300px" border="0">
        <tr>
            <td>第一个数</td>
            <td><input type="text" name="number1"></td>
        </tr>
        <tr>
            <td>第二个数</td>
            <td><input type="text" name="number2"></td>
        </tr>
        <tr>
            <td>运算符</td>
            <td>
                <select name="oper">
                    <option value="+">+</option>
                    <option value="-">-</option>
                    <option value="*">*</option>
                    <option value="/">/</option>
                </select>
            </td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="计算结果"></td>
        </tr>
    </table>
</form>
</body>
</html>
```

新建一个计算结果的 CalResult. php 页面，输入以下代码：

```
<? php
    //接收用户从 Calculator.php(对应静态页面 浏览器)提交的数据
    //1、接收 number1 number2 ;REQUEST 方法可以接收用户的 post 或者 get 请求数据
     $number1 = $_REQUEST['number1'];
     $number2 = $_REQUEST['number2'];
    //接收运算符
     $oper = $_REQUEST['oper'];
     $res = 0;
    switch ( $oper) {
        case '+':
             $res = $number1 + $number2;
            break;
        case '-':
```

```
            $res = $number1 - $number2;
            break;
        case '*':
            $res = $number1 * $number2;
            break;
        case '/':
            $res = $number1/ $number2;
            break;
        default:
            echo "运算符不正确";
            break;
    }
    echo "运算结果是:". $res;
? >
<br><a href="Calculator.php">返回计算器</a>
```

程序运行结果如图 13-5 所示。

图 13-5　程序运行结果

实训 4　PHP 函数的应用

【实训目的】

（1）掌握 PHP 中函数的定义和使用方法。

（2）掌握 PHP 中变量的作用域。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

（1）设计一个排序函数，当用户输入一串数字后，程序会自动进行排序。

新建一个文件 sort. php，输入以下代码：

```
<? php
//定义排序函数 my_sort( )
function my_sort( $array)
{
    for( $i=0; $i<count( $array); $i++)
    {
        for( $j= $i+1; $j<count( $array); $j++)
        {
            if( $array[ $i]> $array[ $j])
            {
                $tmp= $array[ $j];
                $array[ $j]= $array[ $i];
                $array[ $i]= $tmp;
            }
        }
    }
    return  $array;
}
echo "请输入需要排序的数据:<br>";
echo "<form method=post>";
for( $i=1; $i<6; $i++)
{
    echo "<input type='text'  name='stu[ ]'  size='5' >";
    if( $i<5)
        echo "-";
}
echo "<input type='submit'  name='bt'  value='提交' >";
echo "</form>";
if(isset( $_POST['bt' ]))
{
    $stu= $_POST['stu' ];
    $arr_stu=my_sort( $stu);
    echo "排序后的数据如下所示:<br>";
    while(list( $key, $value)= each( $arr_stu))
    {
        echo  $value."<br>";
    }
}
? >
```

程序运行结果如图 13-6 所示。

（2）编写一个函数，输入是 5 个分数，输出是去掉一个最高分和去掉一个最低分后的平均分。

新建文件 aver. php，输入以下代码：

```
<? php
 $score = array(62,78,73,86,95);
function test( $arr = array( ))
{
     $sum = $max = $min = $arr[0];
    for( $i = 1; $i<count( $arr); $i++)
    {
        if( $max< $arr[ $i])
            $max = $arr[ $i];
        if( $min> $arr[ $i])
            $min = $arr[ $i];
         $sum = $sum+ $arr[ $i];
        }
     $sum = $sum- $max- $min;
    //echo count( $arr), $max, $min, $sum;
    return  $sum/5;
}
echo test( $score);
? >
```

程序运行结果为：

47. 4

图 13-6　程序运行结果

实训 5　PHP 数组

【实训目的】

（1）熟悉数组的定义以及初始化。

（2）掌握数组的遍历和输出。

（3）掌握数组的排序以及相关处理函数。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

（1）创建一个长度为 10 的数组，数组中的元素满足斐波拉契数列的规律。

斐波拉契数列，又称黄金分割数列，指的是这样一个数列：1，1，2，3，5，8，13，21，…在数学上，斐波拉契数列以如下递归的方法定义：$F_0=0$，$F_1=1$，$F_n=F_{n-1}+F_{n-2}$（$n\geqslant 2$，$n\in \mathbf{N}$）。特别指出：第 0 项是 0，第 1 项是第一个 1。

创建 fibo. php 文件，输入以下代码：

```
<? php
function arrFibo( $len){
 $arr[0] = 0;
 $arr[1] = 1;
for( $i=2; $i< $len; $i++){
 $arr[ $i] =  $arr[ $i-1]+ $arr[ $i-2];
}
return  $arr;
}
echo "<pre>";
print_r(arrFibo(10));
echo "</pre>";
? >
```

运行结果如图 13-7 所示。

图 13-7　程序运行结果

（2）写一个方法，将一个长度超过 10 的数组最后 5 项直接截取，不改变顺序变为前 5 项，如｛1，2，3，4，5，6，7，8，9，10｝变为｛6，7，8，9，10，1，2，3，4，5｝。

思路：先把数组截取相应的长度（array_slice），再把 2 段数组拼接（array_merge）

创建文件 arrsort. php，输入以下代码：

```
<? php
    function arrsort( $arr)
```

```
{
    $num = count( $arr);
    if( $num > 10)
    {
        //array_slice( $arr,起始位置,截取长度,保留索引(默认为 false))
        $arr_firstpart = array_slice( $arr,0, $num-5,true);
        $arr_lastpart = array_slice( $arr,( $num-5),5,true);
    }
    else
    {
        echo "数组不超过 10 个元素,请重新输入";
        exit();
    }
    //拼接
    $arr_new = array_merge( $arr_lastpart, $arr_firstpart);
    return $arr_new;
}
$arr = array("a"=>1,2,3,8,9,6,"b"=>5,-1,"c"=>8,0,7);
echo "<pre>";
print_r( $arr);
echo "<br>= = = = = 拼接后 = = = = <br><br>";
print_r(arrsort( $arr));
echo "</pre>";
? >
```

运行结果如图 13-8、图 13-9 所示。

图 13-8 拼接前

图 13-9 拼接后

（3）逆序输出数组（不能使用 rsort（）函数，不能生成新数组，使用 array_reverse（）函数会创建新的数组，所以不能使用）。

新建 arr. php 文件，输入以下代码：

```
<? php
 $arr = array("who","your","name",1,10);
 $i = "";//要替换位置的数的下标
 $j = "";//临时变量
 $k = "";//被替换位置的数的下标
 $len = count( $arr);
 $half_len = floor( $len/2);//向下取整,取整的值是循环的次数
for( $i=0; $i< $half_len; $i++)
{
   $j = $arr[ $i];
  //判断数组个数奇偶
  if( $len%2! =0)
  { //奇数
     $k = $half_len * 2- $i;
  }
  else
  {
    //偶数
     $k = $half_len * 2- $i-1;
  }
   $arr[ $i] = $arr[ $k];
   $arr[ $k] = $j;
}
echo "<pre>";
print_r( $arr);
echo "</pre>";
? >
```

程序运行结果如图 13-10 所示。

图 13-10　程序运行结果

实训 6　字符串处理与正则表达式

【实训目的】

掌握字符串处理的相关函数。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

利用字符串处理的相关函数对超长文本进行分页显示。

程序分析：首先定义两个自定义函数 unhtml（$str）和 msubstr（$str）。其中，函数 unhtml（）的功能就是将文本中的一些特殊字符转换为 HTML 标记符号，msubstr（）函数的功能就是根据设定篇幅的长度对文本进行分页截取。另外还需要对文本进行相应的计算。

每页显示的字节数（char_size）：如每页显示 500 个字节，$char_size=500。

文本总字节数（$length）：strlen（unhtml（$counter））。

总页数（page_count）：

```
$page_count=ceil( $length/ $char_size);
```

上一页的字节数：

```
msubstr( $counter,0,( $_GET['page']-1) * $char_size);
```

下一页的字节数：

```
msubstr( $counter,0, $_GET['page'] * $char_size);
```

当前页显示的文本：

```
substr( $c1,strlen( $c),strlen( $c1)-strlen( $c));
```

创建一个文件 funye.php，程序代码如下：

```
<span style="font-size:18px;">
    <table width=400 align=center border=1>
    <tr><td align=center>PHP 学习心得分享</td></tr>
    <tr><td>
    <? php
        function msubstr( $str, $start, $len){
            $tmpstr="";
            $strlen = $start + $len;
            for( $i=0; $i< $strlen; $i++){
    /* 字符串截取函数 substr()。ord()返回字符的 ASCII 码值 >0xa0 说明是个汉字,占两个字节,所以读取两个 */
                if(ord(substr( $str, $i,1))>0xa0){
```

```
                $tmpstr.=substr( $str, $i,2);
                $i++;
            }else{
                $tmpstr.=substr( $str, $i,1);
            }
        }
        return  $tmpstr;
    }
    $page=@ $_REQUEST['page'];
    if( $page=="" ){
        $page = 1;
    }
    if( $page){
//文件读取函数,相当于 fopen、fgets、fclose 三个函数的结合,将文件内容全部读完
        $counter = file_get_contents("file.txt");
        $length = strlen( $counter); //字符串长度计算函数 strlen()
        $page_count=ceil( $length/400); //ceil(float  $value)返回不小于 $value 的下一个数
        $c=msubstr( $counter,0,( $page-1) * 400);
        $c1 = msubstr( $counter,0, $page * 400);
        echo substr( $c1,strlen( $c),strlen( $c1)-strlen( $c));//输出当前页的数据
    }
?>
</td></tr></table>
<table align=center width=400>
    <tr>
        <td>页次:<? php echo  $page;? >/<? php echo  $page_count;? >页</td>
        <td>分页:
        <? php
            if( $page! =1){
                echo "<a href=funye.php? page=1>首页</a> ";
                    //? 后面的是传递的数据,这个先不用管。
                echo "<a href=funyet.php? page=".( $page-1)."">上一页</a> ";
            }
            if( $page< $page_count){
                echo "<a href=funye.php? page=".( $page+1)."">下一页</a> ";
                echo "<a href=funye.php? page=". $page_count."">尾页</a> ";
            }
        ? >
</td></tr>
</table></span>
```

程序运行结果如图 13-11 所示。

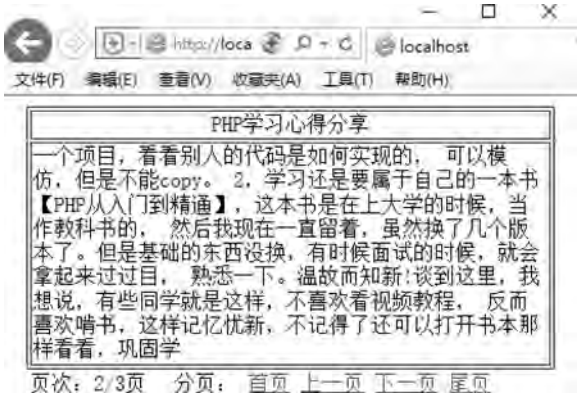


图 13-11 程序运行结果

实训 7 文件与目录访问操作

【实训目的】

（1）掌握文件与目录的常用函数。

（2）掌握全局变量 $_ FILES [ ] 的使用。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

如果做一个在线的文件管理系统的话，那么首先必须知道怎么读取目录和文件，其实这个功能几行代码就可以实现了。

例如：使用 PHP 文件与目录的常用函数，实现读取目录中的文件信息，并显示。

创建一个上传文件的页面 info. php，输入以下代码：

```
<? php
    $dir = "D:/"; //要获取的目录
    echo "* * * * * * * * * 获取目录下所有文件和文件夹 * * * * * * * * * *<hr/>";
    //先判断指定的路径是不是一个文件夹
    if (is_dir( $dir)){
    if ( $dh = opendir( $dir)){
    while (( $file = readdir( $dh))! = false){
    //文件名的全路径 包含文件名
    $filePath = $dir. $file;
```

```
//获取文件修改时间
 $fmt = filemtime( $filePath);
echo $filePath."<br/>";
}
closedir( $dh);
}
}
? >
```

运行结果如图 13-12 所示。

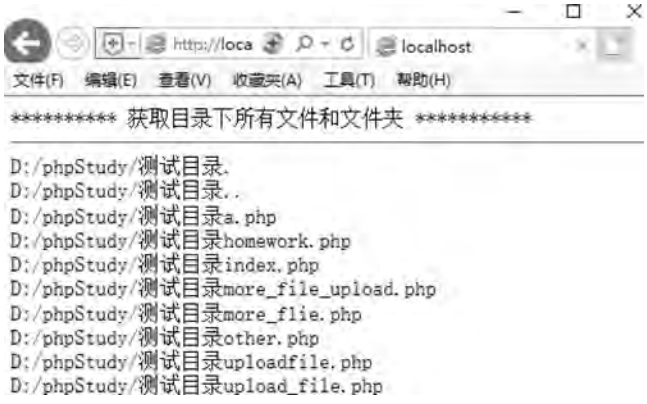


图 13-12　程序运行结果

实训 8　图形图像处理

【实训目的】

(1) 掌握文件与目录的相关函数。

(2) 掌握全局变量的使用。

(3) 掌握图形图像处理技术相关函数。

【实训环境】

在 Windows 下安装或提供以下软件：

(1) EclipsePHP Studio。

(2) phpStudy 最新版本。

【实训步骤】

在前面章节所学的文件上传系统后，要求上传图片后，并能显示缩略图。

新建文件 image_upload. php，输入以下代码：

```
<form method="post" action="" enctype="multipart/form-data">
```

```
<h3>缩略图显示上传的图片</h3>
<input type="file" name="picture"><br>
<input type="submit" name="submit" value="提交">
</form>
<? php
if(isset( $_POST['submit']))
{
    $filename= $_FILES['picture']['name'];
    $type= $_FILES['picture']['type'];
    $upfile="image/small.jpeg";
    $small_upfile="image/image_small.jpeg";
    if(! $filename)
    {
        echo "<script>alert('文件不存在!')</script>";
    }
    else if( $type! ='image/pjpeg')
    {
        echo "<script>alert('文件格式不正确!')</script>";
    }
    else
    {
        move_uploaded_file( $_FILES['picture']['tmp_name'], $upfile);
        $dst_w=150;                                    //设定缩略图的宽
        $dst_h=150;                                    //设定缩略图的高
        $src_image=imagecreatefromjpeg( $upfile);      //读取上传后的文件并创建图像
        $src_w=imagesx( $src_image);                   //获得图像的宽
        $src_h=imagesy( $src_image);                   //获得图像的高
        $dst_image=imagecreatetruecolor( $dst_w, $dst_h);  //创建新图像
        //将图像 $src_image 重新定义大小并写入新的图像 $dst_image 中
        imagecopyresized( $dst_image, $src_image,0,0,0,0, $dst_w, $dst_h, $src_w, $src_h);
        imagejpeg( $dst_image, $small_upfile);         //将新图像保存为 $small_upfile
        echo "文件上传成功,缩略图如下:<br>";
        echo "<img src= $small_upfile>";               //显示图像
        imagedestroy( $src_image);
        imagedestroy( $dst_image);
    }
}
? >
```

程序运行结果如图 13-13 所示。

图 13-13　程序运行结果

实训 9　PHP 面向对象程序设计

【实训目的】

（1）掌握类的定义。

（2）掌握类的成员方法，构造方法的定义和使用。

（3）掌握类的封装。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

利用类封装一个连接数据库的类。

（1）分析。定义一个构析函数，将外部传入的参数变量转换为类的内部变量，并在类实例化的时候调用 connect（）方法。创建 connect（）方法，其功能是利用 mysqli_connect（）函数连接 MySQL 数据库。

创建一个文件 class. php，输入以下代码：

```
<? php
    class mysql_connect
{
    private  $host;
    private  $user;
```

```
        private  $pwd;
        private  $conn;
        private  $dbname;
        public function _construct( $host, $user, $pwd, $conn, $dbname){
              $this->host= $host;
              $this->user= $user;
              $this->pwd= $pwd;
              $this->connt= $conn;
              $this->dbname= $dbname;
        }
        public function connect( ){
         $this->conn=mysqli_connect( $this->host, $this->user, $this->pwd, $this->dbname);
        }
    }
     $mysql=new mysql_connect("localhost","root","root","student");
    ? >
```

（2）已知一个类 pen，请实例化名称为“英雄牌”，颜色为“黑色”，并调用方法输出信息。输入以下代码：

```
    <? php
        header("Content-Type:text/html;charset=gb2312");
        class pen{
            public  $name;
            public  $color;
            public function attri( $p_name, $p_color) {
                 $this->name= $p_name;
                 $this->color= $p_color;
            }
            public function print_attri( ){
                 echo "名称:", $this->name,"<br/>";
                 echo "颜色:", $this->color,"<br/>";
            }
        }
         $my_pen=new pen( );
         $my_pen->attri("英雄","黑色");
         $my_pen->print_attri( );
    ? >
```

实训 10　PHP 与 Web 页面交互

【实训目的】

（1）熟悉表单的应用。

（2）熟悉标签的使用。

（3）熟悉 Cookie 的使用

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

Cookie 程序设计，用户成功后如果选择保存 Cookie，则以后再次访问时就不需要重新登录，会自动跳到到欢迎界面，如图 13-14 所示，并显示用户的访问次数和上次登录的时间。新建 cookie. php 主程序文件，输入以下代码：

```
<? php
if (@ $_COOKIE["user"]["xm"]<>""){                    //尝试获取指定的 Cookie 变量
    $visnum=intval( $_COOKIE["user"]["num"])+1;       //将原来的访问次数加 1
    $expire=intval( $_COOKIE["user"]["expire"]);      //获取有效期
    //将本次访问时间写入 Cookie
    setcookie("user[dt]",@ date("Y-m-d h:i:s"),time()+3600 * $expire);
    setcookie("user[num]", $visnum,time()+3600 * $expire); //将本次访问次数写入 Cookie
    echo"欢迎光临登录,您的用户名是:". $_COOKIE["user"]["xm"];//输出 Cookie 变量值
    echo "<br/>这是您第". $visnum."次光临本网站";
    echo "<br/>您上次访问是在". $_COOKIE["user"]["dt"];
}
else                    //没有 Cookie 则显示登录表单
    echo '<html><body>
<div style="border:1px solid #06f; background:#bbdeff">
  <form method="post" action="set_cookie.php" style="margin:4px;">
    <p>帐号: <input name="xm" type="text" size="12"></p>
    <p>密码: <input name="Pwd" type="password" size="12"></p>
    <p>保存: <select name="Save">
        <option value="-1">不保存</option>
        <option value="7">保存 1 周</option>
        <option value="30">保存 1 月</option></select>
        <input type="submit" value="登 录"></p>
```

```
    </form></div></body></html>'
? >
```

新建一个 set_cookie. php 页面，输入以下代码：

```
<? php
    if ( $_POST[ "xm" ] = = "admin" && $_POST[ "Pwd" ] = = "123" ) {
setcookie( "user[ xm ]" , $_POST[ "xm" ] ,time( ) +3600 * intval( $_POST[ 'Save' ] ) ) ;
setcookie( "user[ dt ]" ,@ date( "Y-m-d h:i:s" ) ,time( ) +3600 * $expire) ;//写入 Cookie
setcookie( "user[ num ]" ,1 ,time( ) +3600 * intval( $_POST[ 'Save' ] ) ) ;
    //保存有效期到 Cookie
setcookie( "user[ expire ]" , $_POST[ 'Save' ] ,time( ) +3600 * intval( $_POST[ 'Save' ] ) ) ;
echo $_POST[ "xm" ] . "：首次光临" ;
    //var_dump( $_COOKIE) ;
    }
else
    echo "<script>alert( '用户名或密码不对' ) ;location.href = ' cookie.php ' ;</script>" ;
? >
```

程序运行结果如图 13-14 所示。

图 13-14　程序运行结果

实训 11　MySQL 数据库

本实训主要讲解利用 phpAdmin 图形化管理工具管理数据库的操作过程。

【实训目的】

（1）熟悉 phpAdmin 的安装和配置。

（2）熟悉 SQL 常用命令。

（3）熟悉 phpAdmin 图形化管理数据库。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

（1）在 phpMyAdmin 图形化管理工具的可视化界面中进行数据库增、删、改操作。

在浏览器的地址栏中输入：localhost/phpMyAdmin/，回车，就可以进入 phpMyAdmin 主界面了。

1）创建数据库。在 phpMyAdmin 的主界面中，可以看见有“语言-Languange”的下拉框，我们可以在下拉框中选择“中文-Chinese simplified”选项，然后在上面的“服务器连接校对”框中选择 utf8_general_ci 简体中文编码格式，防止出现乱码的情况，如图 13-15 所示。

图 13-15　常规设置

当上面的设置好了，下面开始创建数据库，点击左上方的“数据库”就会出现如图 13-16 所示的界面。

图 13-16　创建数据库

然后在左边的文本框中输入自己的数据库名字，本实训输入“php. cn”，在右边的下接列表框中选择“utf8_general_ci”，然后点击“创建”按钮，在左侧栏就可以看见所创建的数据库了，如图 13-17 所示。

图 13-17　创建 php. cn 数据库成功

点击一下创建的数据库，就可以对数据库进行管理了，操作界面如图 13-18 所示。

图 13-18　php. cn 数据库的管理界面

2）修改数据库。在数据库管理界面（见图 13-18）有个“操作”按钮，点击进入修改操作数据库的页面，如图 13-19 所示。

a. 在修改页面左上方可以对当前数据库执行创建数据表的操作，在创建数据表提示信息下的两个文本框中分别输入要创建的数据表名字和字段总数，点击“执行”按钮，就进入创建数据表结构页面。

b. 在修改页面的右上方可以对数据库重命名，在“将数据库改名为:”的文本框中输入新的数据库名字，单击“执行”则修改数据库名成功。

图 13-19　修改数据库

3）删除数据库。同样是点击“操作”按钮进入修改页面，在“新建数据表”的正下面就是删除数据库，点击就可以删除数据库，如图 13-20 所示。

图 13-20　删除数据库

注意：数据库在我们日常开发中是非常重的，里面有很多数据，如果要删除一定要谨慎，一旦删除就不可恢复，建议在删除之前先备份数据库。

（2）在 phpMyAdmin 界面中使用 SQL 语句操作数据表。

单击 phpMyAdmin 主界面中的“SQL”按钮，打开 SQL 语句编辑区，输入完整的 SQL 语句，来实现数据的查询、添加、修改和删除操作。

1）使用 SQL 语句插入数据。在 SQL 语句编辑区中使用 insert 语句向数据表 php. cn 中插入数据，单击“执行”按钮，向数据表中插入一条数据。如果提交的 SQL 语句有错误，那么系统就会给出警告，提示用户修改，如图 13-21 所示。

图 13-21　插入数据

如果 SQL 语句没有错误，提交后点击左上角的“浏览”按钮，即可进入数据展示页面进行浏览，如图 13-22 所示。

图 13-22　数据展示

2）使用 SQL 语句修改数据。在 SQL 编辑区使用 update 语句修改数据信息，例如，将 id 为 100 的用户的名称修改为“PHP 个人网站”，密码修改为：“myphp”，如图 13-23 所示。

图 13-23　修改数据

3）使用 SQL 语句查询数据。在 SQL 语句编辑区使用 select 语句检索指定条件的数据信息，例如，将 id 小与 4 的信息全部显示出来，如图 13-24 所示。

图 13-24　查询数据

除了对整个表的简单查询外，还可以进行一些复杂的条件查询（使用 where 子句提交 LIKE、ORDER BY，GROUP BY 等条件查询语句）及多表查询。

4）使用 SQL 语句删除数据。在 SQL 语句编辑区使用 delete 语句检索指定条件的数据或者全部信息，比如：删除名称为 baidu. com 的信息，SQL 语句如图 13-25 所示。

图 13-25　删除数据

注意：如果 delete 语句后面没有 where 条件语句，那么就会删除指定数据表中的全部数据。

实训 12　PHP 访问数据库

【实训目的】

（1）熟悉 PHP 连接数据库。

（2）熟悉 PHP 操作数据库的函数。

【实训环境】

在 Windows 下安装或提供以下软件：

（1）EclipsePHP Studio。

（2）phpStudy 最新版本。

【实训步骤】

（1）编写程序，连接 MySQL 服务器的数据库 MyPHP。实现代码如下：

```
<? php
    global  $conn;                          //定义数据库连接标识符
    $host = "localhost";                    //MySQL 服务器地址
```

```
    $user="root";                    //MySQL 登录用户名
    $password="root";                //MySQL 登录密码
    $dbname="MyPHP";                 //所操作的数据库
    $conn=mysqli_connect( $host, $user, $password, $dbname);
    mysqli_query( $conn,"set names gb2312");
?>
```

（2）编写程序，向 student 表插入数据：学号为“201901045”，姓名为“张晓芳”，班级为“19 级计算机应用 4 班”，PHP 成绩为“87”。实现代码如下：

```
<? php
include_once("connect_mysql.php");
header("Content-Type:text/html;charset=gb2312");
 $sql="insert into student(xh,xm,bj,cj) values('201901045','张晓芳','19 级计算机应用 4 班',
87)";
 $result=mysqli_query( $conn, $sql);
if( $result)
    echo "插入记录的行数:".mysqli_affected_rows( $conn);
else
    echo"数据插入失败!";
? >
```

（3）编写程序，将 student 表中姓名为“李硕”的成绩改为 97。实现代码如下：

```
<? php
include_once("connect_mysql.php");
header("Content-Type:text/html;charset=gb2312");
 $sql="update student set cj=97 where xm='李硕'";
    $result=mysqli_query( $conn, $sql);
    if( $result)
        echo "记录修改成功!";
    else
        echo "记录修改不成功!";
? >
```

（4）编写程序，将 student 表中成绩大于 90 分的记录全部都显示出来。实现代码如下：

```
<? php
include_once("connect_mysql.php");
header("Content-Type:text/html;charset=gb2312");
 $sql="select * form student where cj>90";
 $result=mysqli_query( $conn, $sql);
while( $r=mysql_fetch_object( $result))
{
echo $r->xh,"", $r->xm,"", $r->bj,"", $r->cj,"<br/>";
}
? >
```

第 14 章

综 合 实 训

基于 PHP 和 MySQL 的简单学生成绩管理系统

本系统是在 Windows 环境下，基于 PHP 脚本语言实现的学生成绩管理系统，使用 phpStudy2018 软件，该软件集成 Apache、MySQL 数据库等软件。本系统包含学生的成绩录入、添加、删除等。由于时间有限，本系统只有简单的功能，读者可以在本系统的基础上进行相应的扩展，如增加课程信息的录入、查询、删除、更新和课程排序、成绩排名等更多的功能。

本系统只提供重要数据和实现代码。

1. 数据库的创建

数据库创建有多种方法，最主要用的方法是在 phpMyAdmin 图形化界面进行数据库、数据表的创建，另一种方法就是使用 SQL 语句进行创建。

本系统使用的数据库名称为 xscj，数据库中的两个表分别是 stu_info 和 user_info。表的详细结构信息见表 14-1、表 14-2。

表 14-1　stu_info 表的结构

列　名	数据类型	长度	是否可空	默认值	说　明
id	不定长字符型（VARCHAR）	20	√	无	主键，由英文字母、下划线或数字组成
姓名	字符型（CHAR）	8	√	无	
班级	字符型（char）	30	√		
性别	字符型（char）	2	√	1	
C 语言程序设计	整数型（INT）	4	√	0	
网络编程 PHP	整数型（INT）	4	√	0	
Flash 动画设计	整数型（INT）	4	√	0	
Java 程序设计	整数型（INT）	4	√	0	
总分	整数型（INT）	8	√	0	

表 14-2　user_info 表的结构

项目名	列　名	数据类型	长度	是否可空	默认值	说　明
用户名	username	不定长字符型（VARCHAR）	20	×	无	主键，由英文字母、下划线或数字组成
密码	password	不定长字符型（VARCHAR）	20	√	无	6~20 位字符
管理员	is_manager	字符型（char）	10	√		
性别	sex	位型（BIT）	默认值	√	1	1：男；0：女
年龄	age	整型（INT）	默认值	√	无	
邮箱	email	定长字符型（CHAR）	30	√	无	

2. 登录界面

登录界面如图 14-1 所示，主要提供登录和注册功能。

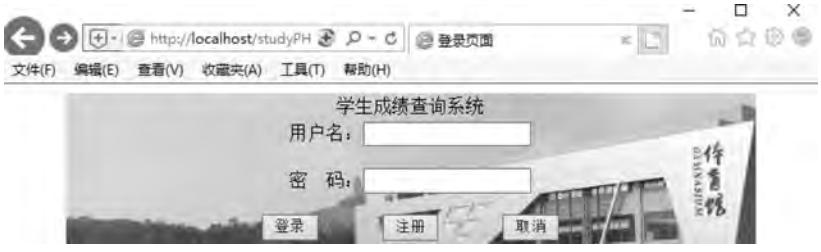


图 14-1　登录界面

新建文件名 login. php，输入以下代码：

```
<html>
<head><title>登录页面</title></head>
<body>
<table background=bgimage.jpg width=600 height=100 align=center>
<tr><td align=center>学生成绩查询系统</td></tr>
<tr><td align=center>
<form method=post action=check.php>
用户名:<input type=text name=username><br><br>
密   码:<input type=password name=password><br><br>
<input type=submit name=sumbit value="登录">      
<input type=reset name=zc value="注册" onclick="window.location='regist.php'">  
     
<input type=reset name=sumbit value="取消">
</form></td></tr>
</table>
</body>
```

```
</html>
<? PHP
include" conn.php" ;
? >
```

3. 注册页面

注册页面主要完成注册，其中有两种身份选择，一个是管理员身份，一个是普通身份(学生)。本页面主要应用了正则表达式，对各项录入信息进行有效性的检测，在本系统中，它的安全性还不高，后续需要更新修改。

新建 regist. php，输入以下代码：

```
<html>
<head>
<title>用户注册页面</title>
</head>
<body>
<form action=" " method=" post" >
<div align=" center" ><font size=" 5" color=" blue" >新用户注册</font></div>
<table width=" 370" align=" center" border=" 0" >
<tr><td width=" 80" align=" right" >用户名:</td>
    <td><input type=" text" name=" userid" ></td><td><font color=" red" > * 1-20 个字符</font></
td></tr>
<tr><td align=" right" >密码:</td>
    <td><input type=" password" name=" pwd1" size=" 21" ></td><td><font color=" red" > * 6-20 个
字符</font></td></tr>
<tr><td align=" right" >确认密码:</td>
    <td><input type=" password" name=" pwd2" size=" 21" ></td><td> </td></tr>
<tr><td align=" right" >性别:</td>
    <td><input type=" radio" name=" sex" value=" 1" >男
    <input type=" radio" name=" sex" value=" 0" >女
    </td><td> </td></tr>
<tr><td align=" right" >年龄:</td>
    <td><input type=" text" name=" age" ></td><td> </td></tr>
<tr><td align=" right" >email:</td>
    <td><input type=" text" name=" email" ></td><td> </td></tr>
<tr><td colspan=" 3" align=" center" >
<tr><td align=" right" >管理员:</td>
    <td><input type=" radio" name=" manager" value=" 1" >是
    <input type=" radio" name=" manager" value=" 0" >否
    </td><td> </td></tr>
<tr><td colspan=" 3" align=" left" ><input type=" submit" name=" Submit" value=" 提交" > 
  
```

```
<input type="reset" name="Submit2" value="重置">   
<input type="reset" name="Submit3" value="返回登录" onclick="window.location='login.php'">
</td></tr><br>
</table>
//注意:禁止使用火狐浏览器,否则无法注册。
</form>
</body>
</html>
<? php
include "conn.php";
if(isset( $_POST['Submit'] ))
{
    $userid= $_POST['userid'];
    $pwd1= $_POST['pwd1'];
    $pwd2= $_POST['pwd2'];
    $sex=@ $_POST['sex'];
    $age= $_POST['age'];
    $email= $_POST['email'];
    $manager= $_POST['manager'];
    $checkid=preg_match('/^\w{1,20} $/', $userid);//使用正则表达式检查用户名
    $checkpwd1=preg_match('/^\w{6,20} $/', $pwd1);
    $checkemail=preg_match('/^[a-zA-Z0-9_\-]+@[a-zA-Z0-9\-]+。[a-zA-Z0-9\-\.]+
$/', $email);
if(! $checkid)
        echo "<script>alert('用户名设置错误!');</script>";
    elseif(! $checkpwd1)
        echo "<script>alert('密码设置错误!');</script>";
    /*elseif(! $sex)
        echo "<script>alert('性别为必选项!');</script>";*/
    elseif( $age&&(! is_numeric( $age)))
        echo "<script>alert('年龄必须为一个数字!');</script>";
    elseif( $email&&(! $checkemail))
        echo "<script>alert('email 格式错误!');</script>";
    elseif( $pwd1! = $pwd2)
        echo "<script>alert('两次输入的密码不一致!');</script>";
    else
    {
        include "conn.php";                    //连接数据库
        $s_sql="select * from user_info where username=' $userid'";
        $s_result=mysql_query( $s_sql);
        if(mysql_num_rows( $s_result)! =0)
```

```
                echo "<script>alert('用户名已存在!');</script>";
            else
            {
                if( $age)
                {
                    if( $email)
$in_sql="insert into user_info values(' $userid',' $pwd1',' $manager', $sex, $age,' $email')";
                    else
$in_sql="insert into user_info values(' $userid',' $pwd1',' $manager', $sex, $age,NULL)";
                }
                else
                {
                    if( $email)
$in_sql="insert into user_info values( $userid, $pwd1,' $manager', $sex,NULL, $email)";
                    else
$in_sql="insert into user_info values( $userid, $pwd1,' $manager', $sex,NULL,NULL)";
                }
                 $in_result=mysql_query( $in_sql, $con);
                if(mysql_affected_rows( $con)==0)
                    echo "<script>alert('注册失败!');</script>";
                else
                {
                    //注册成功后跳转到登录页面
                    echo "<script>alert('注册成功!');location.href='login.php';</script>";
                }
            }
        }
}
?>
```

程序运行结果如图 14-2 所示。

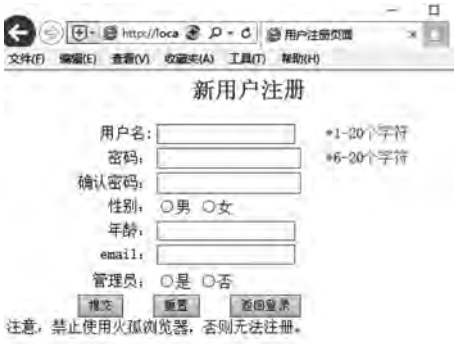


图 14-2　注册界面

4. 主程序界面

登录服务：分为管理员登录和学生登录，若登录时输入数据库中没有的账号和密码，系统会自动注册，即将新账号信息写入 user_info 中去。

学生登录只可以进行查询操作，管理员登录可以进行增删和查询操作，学生具有注册新账号的权限。管理员登录界面如图 14-3 所示。

图 14-3　主程序界面

5. 有效性检测页面

用户登录时，通过 PHP 访问 MySQL 数据库，校对密码有效性检测，新建文件 check. php，输入以下代码：

```
<title>密码验证</title>
<meta charset="gb2312">
    <? php
        function get_str( $str){
            $val = ( $_POST[ $str])? $_POST[ $str]:null;
            return $val;
        }
        $user = get_str("username");
        $pass = get_str("password");
        if( $user==null or $pass == null){? >
            <script type="text/javascript">
                alert("用户名或密码为空,请重新输入");
                window.location.href="login.php";
            </script>
```

```
            <? php
        }
        $con = mysql_connect("localhost","root","root") or die("数据库连接失败");
        if(! $con){
    ? >
    <script type = "text/javascript">
        alert("连接服务器失败");
        window.location.href = "login.php";
    </script>
    <? php
        }
        $sel = mysql_select_db("xscj", $con) or die("数据库选择失败");
        if(! $sel){
    ? >
    <script type = "text/javascript">
        alert("选择数据库失败");
        window.location.href = "login.php";
    </script>
    <? php
        }
        $sql = "select * from user_info where username = ' $user' ";
        //echo $sql."<br>";
        $info = mysql_query( $sql, $con);
        echo $info;
        $num = mysql_num_rows( $info);
        ? >
<? php
            if( $row[' password' ] = = $pass){
                if( $row[' is_manager' ] = = 1){
                    //是管理员
        ? >
        <script type = "text/javascript">
            alert("管理员登录成功");
            window.location.href = "welcome_mag.php";
        </script>
        <? php }? >
        <? php
        if( $row[' is_manager' ] ! = 1){
            //普通用户登录
            //直接进入查询界面
        ? >
```

```
            <script type = "text/javascript">
                alert("普通用户登录成功");
                window.location.href = "select_stu.php";
            </script>
            <? php }
                }
                else{
            ? >
            <script type = "text/javascript">
                alert("密码错误");
                window.loaction.href = "login.php";
            </script>
            <? php
                }
                mysql_close( $con);
? >
```

6. 系统菜单

新建页面 welcom_mag. php，本页面主要的功能是显示本系统的菜单，并且使用了 CSS 样式表对文件的控制。输入以下代码：

```
<? PHP
include"conn.php";
? >
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>欢迎界面</title>
<style>
    a{text-decoration: none;font-size:25px;font-family:微软雅黑;color:rgb(98,94,91);}
    a:link{color:#0000FF;}/*未访问的颜色 蓝色*/
    a:hover{color:#FF00FF;}/*鼠标放上去的颜色(没起作用) 紫色*/
    a:visited{color:#00FFFF;}/*访问过的颜色 蓝色*/
    a:active{color:#FF00FF;}/*选定的代码 紫色*/

</style>
</head>
<div style="font-size:25px;font-family:微软雅黑;color:rgb(0,0,0);" align="center">欢迎管理员登
录,您可以进行所有操作。</div>
<div style="height:180px;width:180px;margin:0px auto;" align="right">
    <p sytle="text-align:center;">
```

```
<img src="timg.gif" width="160" height="160" align="right" />
</p>
</div>

<div style="font-size:25px;font-family:微软雅黑;color:rgb(0,0,0);" align="center">请选择您需要的服务</div>
<div style="height:100px;width:240px;margin:0px auto;" align="center">
    <p><a href="insert.php" align="center">输入学生成绩信息</a></p>
    <p><a href="select_mag.php" align="center">查询学生信息</a></p>
    <p><a href="delete.php" align="center">删除学生信息</a></p>
</div>
<body>
</body>
</html>
```

7. 管理员信息查询

新建信息查询页面 select_mag. php，实现代码如下：

```
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>信息查询界面</title>
<style type = "text/css">
    a{text-decoration: none;font-size:25px;font-family:微软雅黑;color:rgb(98,94,91);}
    a:link{color:#0000FF;}/*未访问的颜色 蓝色*/
    a:hover{color:#FF00FF;}/*鼠标放上去的颜色(没起作用) 紫色*/
    a:visited{color:#00FFFF;}/*访问过的颜色 蓝色*/
    a:active{color:#FF00FF;}/*选定的代码 紫色*/
</style>
</head>

<body>
<div style="font-size:25px;font-family:微软雅黑;color:rgb(0,0,0);" align="center">
    <p>请输入学生的学号进行查询操作</p>
    <p>如果要查询所有学生信息请输入“所有”进行查询</p>
</div>
<div style="height:180px;width:180px;margin:0px auto;" align="right">
    <p sytle="text-align:center;">
<img src="timg.gif" width="160" height="160" align="right" />
</p>
</div>
```

```
<form action="select_opt.php" method="post" style="margin: 35px auto;padding:30px;box-shadow:
1px 1px 2px 1px #aaaaaa;border-radius: 3px;width:380px;">
    <input type="text" name="check" style = "background-color: rgb(250, 255, 189) ! important;
border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;width:240px;" placeholder
="请输入要查询学生的学号">

    <input type="submit" value="查询" style ="height : 23px;border: 1px;background-color:#
00bee7;color:#fff;width:72px;border-radius: 3px;"></form>
<div style="height:100px;width:240px;margin:0px auto;" align="center">
    <p><a href="welcome_mag.php" align="center">返回服务选择页面</a></p>
</div>
</body>
    </html>
```

程序运行结果如图 14-4 所示。

图 14-4　信息查询页面

8. 查询结果显示

新建一个查询结果显示界面 select_opt. php，代码如下：

```
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
```

```
<title>查询结果</title>
</head>
<style type = "text/css">
    a{text-decoration: none;font-size:25px;font-family:微软雅黑;color:rgb(98,94,91);}
    a:link{color:#0000FF;}/*未访问的颜色*/
    a:hover{color:#FF00FF;}/*鼠标放上去的颜色(没起作用) 紫色*/
    a:visited{color:#00FFFF;}/*访问过的颜色 蓝色*/
    a:active{color:#FF00FF;}/*选定的代码 紫色*/
</style>
<body>
    <div style="font-size:25px;font-family:微软雅黑;color:rgb(98,94,91);" align="center">查询
的结果如下</div>
    <div style="height:180px;width:180px;margin:0px auto;" align="right">
    <p sytle="text-align:center;">
<img src="timg.gif" width="160" height="160" align="right"/>
</p>
</div>
<div style="border:1px;border-color: #00bee7" align="center">
<center>
    <table style="border: solid 1px;border-color:#09F;" align ="center" width=60% >
    <tr>
        <td width=90px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">学号
</td>
        <td width=90px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">姓名
</td>
        <td width=90px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">班级
</td>
        <td width=60px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">性别
</td>
        <td width=60px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">语文
</td>
        <td width=60px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">英语
</td>
        <td width=60px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">数学
</td>
        <td width=60px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">理综
</td>
        <td width=60px style="font-size:20px;font-family:微软雅黑;color:rgb(98,94,91);">总分
</td>
    </tr>
</div>
```

```
<? php
    function get_str( $str){
        $val = ( $_POST[ $str])? $_POST[ $str]:null;
        return $val;
    }
    $sel = get_str("check");
    if( $sel == null){
? >
    <script type="text/javascript">
        alert("输入不能为空,请重新输入");
        window.location.href="select_mag.php";
    </script>
<? php
    }
    $con = mysql_connect("localhost","root","root");
    if(! $con){
        ? >
        <script type="text/javascript">
            alert("数据库连接失败");
            window.location.href="select_mag.php";
        </script>
        <? php
        }
        mysql_select_db("xscj");
        if( $sel ! = "所有"){
        $sql = "select * from stu_info where id = $sel";
        $info = mysql_query( $sql, $con);
        $num = mysql_num_rows( $info);
        if( $num){
            $row = mysql_fetch_row( $info);
            echo $row[0]. $row[1]. $row[2]. $row[3]. $row[4]. $row[5]. $row[6]. $row[7]. $
row[8];
            }
        else{
            ? >
            <script type="text/javascript">
                alert("查询失败");
                window.location.href="select_mag.php";
            </script>
            <? php
        }
```

```
            if( $row){
                echo "<tr><td width=90px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[0]."</td> <td width=90px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[1]."</td> <td width=90px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[2]."</td><td width=90px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[3]."</td><td width=60px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[4]."</td><td width=60px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[5]."</td><td width=60px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[6]."</td><td width=60px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[7]."</td><td width=60px style='font-size:20px;font-family:微软雅黑;color:rgb(98,94,
91);'>". $row[8]."</td></tr>";
            }
        }
        else{
            $sql = "select * from stu_info";
            $info = mysql_query( $sql, $con);
            $num = mysql_num_rows( $info);
            if( $num){          //查询成功
                while( $row = mysql_fetch_row( $info)){     //显示所有信息
                    echo "<tr><td width=120px style='font-size:20px;font-family:微软雅黑;color:rgb
(98,94,91);'>". $row[0]."</td> <td width=120px style='font-size:20px;font-family:微软雅黑;color:
rgb(98,94,91);'>". $row[1]."</td> <td width=120px style='font-size:20px;font-family:微软雅黑;
color:rgb(98,94,91);'>". $row[2]."</td><td width=120px style='font-size:20px;font-family:微软雅
黑;color:rgb(98,94,91);'>". $row[3]."</td><td width=90px style='font-size:20px;font-family:微软雅
黑;color:rgb(98,94,91);'>". $row[4]."</td><td width=90px style='font-size:20px;font-family:微软雅
黑;color:rgb(98,94,91);'>". $row[5]."</td><td width=90px style='font-size:20px;font-family:微软雅
黑;color:rgb(98,94,91);'>". $row[6]."</td><td width=90px style='font-size:20px;font-family:微软雅
黑;color:rgb(98,94,91);'>". $row[7]."</td><td width=90px style='font-size:20px;font-family:微软雅
黑;color:rgb(98,94,91);'>". $row[8]."</td></tr>";
                }
            }
            else{
            ? >
                <script type="text/javascript">
                    alert("查询失败");
                    window.location.href="select_mag.php";
                </script>
            <? php
            }
        }
    ? >
```

```
</table>
</center>
<div style="height:100px;width:240px;margin:0px auto;" align="center">
    <p><a href="select_mag.php" align="center">返回查询页面</a></p>
</div>
</body></html>
```

程序运行结果如图 14-5 所示。

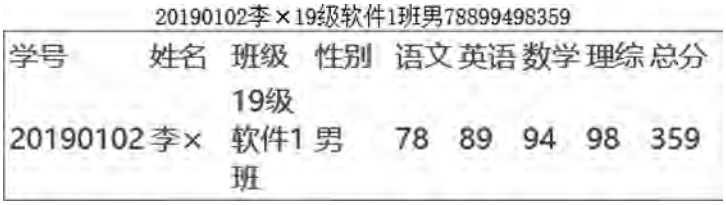


图 14-5　查询结果

9. 增添学生成绩数据

新建增加学生信息页面 insert. php，实现代码如下：

```
<html>
<head>
    <title>增添学生成绩数据</title>
    <meta charset="gb2312">
    <style type = "text/css">
    a{text-decoration: none;font-size:25px;font-family:微软雅黑;color:rgb(98,94,91);}
    a:link{color:#0000FF;}/*未访问的颜色*/
    a:hover{color:#FF00FF;}/*鼠标放上去的颜色(没起作用) 紫色*/
    a:visited{color:#00FFFF;}/*访问过的颜色 蓝色*/
    a:active{color:#FF00FF;}/*选定的代码 紫色*/
</style>
    </head>
    <body>
    <div style="font-size:25px;font-family:微软雅黑;color:rgb(0,0,0);" align="center">请填写表单以添加数据</div>
        <table align=center>
        <form action="mysql_opt.php" method="post" style="margin:35px auto;padding:30px;box-shadow:1px 1px 2px 1px #aaaaaa;
        border-radius: 3px;width:500px;">
        <ul>
            <li>
                学号:<input type="text" name="number" style = "background-color: rgb(250, 255, 189) ! important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;
```

```
width:200px;" placeholder="请输入学号(唯一)">
                </li>
                <li>
                    姓名:<input type="text" name="stuname" style = "background-color: rgb(250, 255, 189) ! important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px; width:200px;" placeholder="请输入姓名">
                </li>
                <li>
                    班级:<input type="text" name="class" style = "background-color: rgb(250, 255, 189) ! important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;width: 200px;" placeholder="请输入班级(可全使用字符)">
                </li>
                <li>
                    性别:<input type="text" name="sex" style = "background-color: rgb(250, 255, 189) ! important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;width: 200px;" placeholder="请输入性别">
                </li>
                <li>
                    C 语言程序设计:<input type="text" name="c" style = "background-color: rgb (250, 255, 189) ! important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;width:150px;" placeholder="请输入 C 语言程序设计(满分 100)">
                </li>
                <li>
                    网络编程 PHP:<input type="text" name="php" style = "background-color: rgb (250, 255, 189) ! important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;width:150px;" placeholder="请输入网络编程 PHP 成绩(满分 100)">
                </li>
                <li>
                    Flash 动画设计:<input type="text" name="flash" style = "background-color: rgb (250, 255, 189) ! important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;width:150px;" placeholder="请输入 Flash 动画设计成绩(满分 100)">
                </li>
                <li>
                    Java 程序设计:<input type="text" name="java" style = "background-color: rgb (250, 255, 189) ! important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;width:150px;" placeholder="请输入 Java 程序设计成绩(满分 100)">
            </li>

            </ul>
            <input type="submit" value="提交" style ="height : 26px;border:1px;background-color:#00bee7;color:#fff;width:72px;border-radius: 3px;">
```

```
    </form></table>
    </div>
    <div style="height:100px;width:240px;margin:0px auto;" align="center">
    <p><a href="welcome_mag.php" align="center">返回服务选择页面</a></p>
</div>
    </body>
</html>
```

程序运行结果如图 14-6 所示。

图 14-6　增加学生信息

10. 数据库后台操作

新建页面 mysql_opt. php，输入以下代码：

```
<html>
<head>
    <title>数据库后台操作</title>
    <meta charset="gb2312">
    </head>
    <body>
        <? php
            //这个页面主要是对插入进行处理
            function get_str( $str){
                $val = ( $_POST[ $str])? $_POST[ $str]:null;
                return $val;
            }
            $num = get_str("number");//接收学号
```

```
$name = get_str("stuname");//接收姓名
$class = get_str("class");//接收班级
$sex = get_str("sex");//接收性别
$c = get_str("c");//接收 C 语言程序成绩
$php = get_str("php");//接收网络编程 PHP 成绩
$flash = get_str("flash");//接收 Flash 动画设计成绩
$java = get_str("java");//接收 Java 程序设计成绩
if( $num==0 or $name==null or $class == null or $sex==null or $c==0 or $php
==0 or $flash==0 or $java==0){? >
    <script type="text/javascript">
        alert("数据输入有误,请重新输入");
        window.location.href="insert.php";
    </script>
    <? php
}
$sum = $c + $php + $flash + $java;//计算总分
$con = mysql_connect("localhost","root","root");
if(! $con){
    ? >
    <script type="text/javascript">
        alert("数据库连接失败");
        window.location.href="insert.php";
    </script>
    <? php
}
$sql = "insert into stuinfo (id,姓名,班级,性别,C 语言程序设计,网络编程 PHP,Flash 动画设计,
Java 程序设计,总分)
values( $num,' $name',' $class',' $sex', $c, $php, $flash, $java, $sum)";
mysql_select_db("xscj");
$info = mysql_query( $sql, $con);
if( $info){
    ? >
    <script type="text/javascript">
        alert("写入成功");
        window.location.href="insert.php";
    </script>
    <? php
}
else{
    ? >
    <script type="text/javascript">
```

```
                    alert("写入失败");
                    window.location.href="insert.php";
                </script>
                <? php
            }
            mysql_close( $con);
        ? >
    </body>
</html>
```

11. 删除信息页面

新建删除信息页面 delete. html，只要输入学生学号就可以删除。输入以下代码：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"/>
<title>删除信息</title>
<style>
    a{text-decoration: none;font-size:25px;font-family:微软雅黑;color:rgb(0,0,0);}
    a:link{color:#0000FF;}/*未访问的颜色*/
    a:hover{color:#FF00FF;}/*鼠标放上去的颜色(没起作用) 紫色*/
    a:visited{color:#00FFFF;}/*访问过的颜色 蓝色*/
    a:active{color:#FF00FF;}/*选定的代码 紫色*/

</style>
</head>
<body>
    <div style="font-size:25px;font-family:微软雅黑;color:rgb(0,0,0);" align="center">请输入学生的学号以删除学生信息</div>
    <div style="height:180px;width:180px;margin:0px auto;" align="right">
    <p sytle="text-align:center;">
<img src="timg.gif" width="160" height="160" align="right"/>
</p>
</div><div style="border:1px;border-color: #00bee7" align="center">
<form action="delete_opt.php" method="post" style="margin: 35px auto;padding:30px;box-shadow: 1px 1px 2px 1px #aaaaaa;border-radius: 3px;width:380px;">
    <input type="text" name="delete" class ="" style = "background-color: rgb(250, 255, 189) !important; border-radius: 3px;box-shadow: none; color: rgb(0, 0, 0);line-height: 26px;width:200px;" placeholder="请输入要删除学生信息的学号">

    <input type="submit" value="删除" style ="height : 22px;border: 1px;background-color:#00bee7;color:#fff;width:72px;border-radius: 3px;">
</form>
```

```
    <div style="height:100px;width:240px;margin:0px auto;" align="center">
    <p><a href="welcome_mag.php" align="center">返回服务选择页面</a></p>
</div>
    </body>
</html>
```

12. 在数据库中实现删除

新建页面 delete_opt.php，输入以下代码：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"/>
<title>删除操作后台执行页面</title>
</head>
<? php
    function get_str( $str){
        $val = ( $_POST[ $str])? $_POST[ $str]:null;
        return $val;
    }
    $number = get_str("delete");

    $con = mysql_connect("localhost","root","root");
    if(! $con){
        ? >
        <script type="text/javascript">
            alert("数据库连接失败");
            window.location.href="delete.php";
        </script>
<? php
    }
    $sql = "select * from stu_info where id = $number";
    mysql_select_db("xscj");
    $info = mysql_query( $sql, $con);
    $num = mysql_num_rows( $info);
    if(! $num){
        ? >
    <script type="text/javascript">
        alert("找不到该同学,请重新输入学号");
        window.location.href="delete.php";
    </script>
    <? php }
    else{
        //找到了该同学,进行删除操作
```

```
        $sql = "delete from stu_info where id = $number";
        $info = mysql_query( $sql, $con);
        if( $info){
            //删除成功
            ? >
            <script type="text/javascript">
                alert("删除成功");
                window.location.href="delete.php";
            </script>
            <? php
        }
        else{
            //删除失败
            ? >
            <script type="text/javascript">
                alert("删除失败");
                window.location.href="delete.php";
            </script>
            <? php
        }
    }
? >
```

13. 数据库连接

新建页面 conn. php，输入以下代码：

```
<? php
 $server="localhost";
 $user="root";
 $passwd="root";
 $database="xscj";
 $con=mysql_connect( $server, $user, $passwd) or die("数据库连接失败");
mysql_select_db( $database, $con) or die("数据库选择失败");
? >
```

至此，本系统基本设计完毕，还有一些页面代码没有提供，一些相应的扩展功能读者可以自行进行设计实现。

参 考 文 献

[1] 郑阿奇. PHP 实用教程 [M]. 2 版. 北京：电子工业出版社，2014.

[2] 唐四薪. PHP 动态网站开发 [M]. 北京：清华大学出版社，2015.

[3] 黄迎久，石炜. PHP 动态网页设计教程 [M]. 北京：清华大学出版社，2017.

[4] 陈明忠，杨杰涌. PHP 动态网站开发案例教程 [M]. 北京：清华大学出版社，2017.

[5] 阮云兰，钟诚，张磊. PHP Web 应用开发案例教程 [M]. 上海：上海交通大学出版社，2017.

[6] 高洛峰. 细说 PHP [M]. 2 版. 北京：电子工业出版社，2015.