项目 1

PHP入门与环境搭建

学习目标

- 熟悉PHP语言的特点，理解PHP的工作流程。
- 掌握PHP开发环境的搭建，学会服务器的基本配置。

技能要点

- PHP概述。
- PHP程序的工作流程。
- PHP开发环境的搭建。
- 常用代码编辑工具。

1.1 项目描述

PHP（Hypertext Preprocessor，超文本预处理器，追溯到最初，应该被称为“Personal Home Page”，即个人主页）是一种服务器端、跨平台、面向对象、HTML嵌入式的脚本语言。本项目将简单介绍PHP语言、PHP的工作流程、PHP开发环境的搭建等，主要目的是让读者对PHP语言有一个整体了解。

1.2 涉及知识

PHP是一种HTML内嵌式语言，一种在服务器端执行的嵌入HTML文档的脚本语言，语言风格类似于C语言，现在被很多网站编程人员广泛运用。PHP的独特语法混合了C、Java、Perl及PHP自创新的语法，可以比CGI或者Perl更快速地执行动态网页。与其他编程语言相比，使用PHP制作的动态页面是将程序嵌入到HTML文档中去执行，执行效率比完全生成HTML标记的CGI高许多。与同样是嵌入HTML文档的脚本语言JavaScript相比，PHP在服务器端执行，充分利用了服务器的性能。PHP执行引擎还可以将用户经常访问的PHP程序驻留在内存中，其他用户再一次访问这个程序时不需要重新编译程序，直接执行内存中的代码就可以了，这也是PHP高效率的原因之一。PHP非常强大，可以实现所有CGI或者JavaScript的功能，而且支持几乎所有流行的数据库及操作系统。

1.2.1 Web开发

Web开发是一个表示网页或网站编写过程的广义术语。网页使用HTML、CSS和JavaScript编写。这些页面可以是类似于文档的简单文本和图形，也可以是交互式的或显示变化的信息。交互式服务器页面的编写略微复杂一些，但可以实现更丰富的网站效果。如今的大多数页面都是交互式的，并提供了购物车、动态可视化甚至复杂的社交网络等现代在线服务。

通俗地说，Web开发就是通常所说的制作网站。它分为网页部分和逻辑部分，也就是前台和后台。前台负责与用户的交互、数据显示（前台页面利用HTML作为载体来显示数据，使用CSS控制样式，编写JavaScript脚本实现复杂交互）；后台负责编写处理这些逻辑的程序。Web开发可以使用PHP、C#、Python、Java等语言。

1.2.2 PHP概述

PHP是一种脚本语言，从本质上说，也就是解释型语言，不需要编译，但需要有相应的脚本引擎来解释执行。

PHP是一种运行于后端服务器的脚本语言，开源且免费，可镶嵌于HTML页面中解析

共存，动态创建输出内容，是构建网页最为省时、简单的解析性脚本语言。自7.0版本发布后，PHP的应用变得更加广泛。

1.2.3 PHP的版本

1995年初，PHP 1.0诞生，Rasmus Lerdof发明了PHP，这是一套简单的Perl脚本，用于跟踪访问者的信息。这个时候的PHP只是一个小工具，被称为“Personal Home Page Tool”（个人主页小工具）。

1995年6月，PHP 2.0诞生，Rasmus Lerdof用C语言重新开发这个工具，取代了最初的Perl程序。这个用C语言编写的新工具的最大特色就是可以访问数据库，可以让用户简单地开发动态Web程序。这个用C语言编写的工具又被称为“PHP/FI”，它已经具有了今天的PHP的一些基本功能。

1998年6月，PHP 3.0诞生，在正式发布之前，PHP 3.0已经过9个月的公开测试。Andi Gutmans和Zeev Suraski加入了PHP开发项目组。这是两位以色列工程师，他们在使用PHP/FI时发现了PHP的一些缺点，然后决定重写PHP的解析器。注意，在这时，PHP就不再被称为“Personal Home Page”了，而是被改称为“PHP：Hypertext Preprocessor”。PHP 3.0是最像现在使用的PHP的第一个版本，这个重写的解释器也是后来Zend的雏形。PHP 3.0最强大的功能是它的可扩展性。它在提供给第三方开发者数据库、协议和API的基础结构之外，还吸引了大量的开发人员加入并提交新的模块。

2000年5月22日，PHP 4.0发布了。许多人认为PHP 4.0的发布是这种语言在企业级开发环境下的正式亮相，这一观点也由于PHP的迅速普及得到了佐证。仅仅在发布后的几个月内，Netcraft（http://www.netcraft.com/）就有约360万个以上的站点安装了PHP。

PHP 5.0是PHP语言发展历程中的一个分水岭。前面的几个主要版本已经增加了许多库，PHP 5.0则在功能上又进行了许多改进，并且增加了成熟的编程语言架构才具有的一些特性。

之后的PHP 6.0具有以下一些特性。

（1）增加了本地的Unicode支持，使构建和维护多语言应用程序变得更加容易。

（2）已经进行了大量有关安全性的改进，基于这些改进，可以显著遏制安全相关问题的泛滥。这些问题其实不能归咎于语言，而应归咎于只会东拼西凑的没有经验的程序员。

（3）增加了许多新的语法特征，其中最突出的就是64位整数类型、经过“改造”的用于多维数组的foreach循环构造，以及对于标签的break的支持。

目前的PHP 7.0具有以下一些特性。

（1）抽象语法树。之前的PHP版本、PHP代码在语法解析阶段直接生成ZendVM指令，也就是在zend_language_parser.y中直接生成opline指令，使编译器和执行器耦合在一起。PHP 7.0是先生成抽象语法树，然后将抽象语法树编译成ZendVM指令，使PHP的编译和执行隔离开。

（2）Native TLS。PHP 7.0使用Native TLS（线程局部存储）保存线程的资源池，也就是通过_ _thread保存一个全局变量，这样该变量就被线程共享了，不同线程的修改不会相互影响。

（3）指定函数参数、返回值类型。

此外，PHP 7.0还具有以下一些新增特性。

1. zval 结构体的变化

```
struct_zval_struct{}
typedef union_zvalue_value{}
```

PHP 7.0将计数器 refcount__gc 放到了内存对象zval中。

2. 异常处理

PHP 5.0中直接抛出一些fatal error的错误，PHP 7.0改为异常抛出，继承throwable可以得到。

3. hashtable 的变化

hashtable即哈希表，也被称为“散列表”，是PHP数组的内部实现结构，也是PHP内核中使用很频繁的一个结构。函数符号表、类符号表、常量符号表都是通过hashtable实现的。PHP 7.0的hashtable从72byte减小到32byte，数组元素bucket结构也从72byte减小到32byte。

4. 执行器

execute_data、opline采用寄存器变量存储，执行器的调度函数为execute_ex。这个函数负责执行PHP代码编译生成的ZendVM指令，在执行期间会频繁地调用execute_data、opline两个变量。在PHP 5.0中，这两个变量是通过参数传递给各指令handler的；在PHP 7.0中，使用寄存器存储，避免了参数出栈、入栈的操作，同时寄存器相对于内存的访问更快，这一优化使PHP的性能提升了5%。

5. 新的参数解析方式

在保留原参数解析方式zend_parse_parameters( )的同时，PHP 7.0提供了另外一种高效的参数解析方式。

1.2.4 PHP的工作流程

PHP是运行于服务器端的脚本语言，实现了数据库与网页之间的数据交互。一个完整的PHP网站系统由以下几部分组成。

- 操作系统：网站运行服务器所使用的操作系统。PHP具有良好的跨平台性，不要求操作系统的特定性，支持Windows和Linux等操作系统。
- Web服务器：当在一台计算机中安装操作系统后，还需要安装Web服务器才能进行HTTP访问。常见的Web服务器软件有Apache、Nginx、IIS等，它们都具有跨平台的特性。
- 数据库系统：实现系统中数据的存储，用于网站数据的存储和管理。PHP支持多种数据库，包括MySQL、SQL Server、Oracle及DB2等。
- PHP包：实现对PHP脚本文件的解析、访问数据库等，是运行PHP代码所必须的软件。
- 浏览器：浏览网页。PHP在发送给浏览器的时候已经被解析器解析成其他代码了（PHP脚本是在服务器端运行的），因此，通过浏览器看到的是经过PHP处理后

的HTML结果。

图1-1完整展示了用户通过浏览器访问PHP网站系统的全过程。通过图例演示，可以更直观地了解PHP的工作流程。

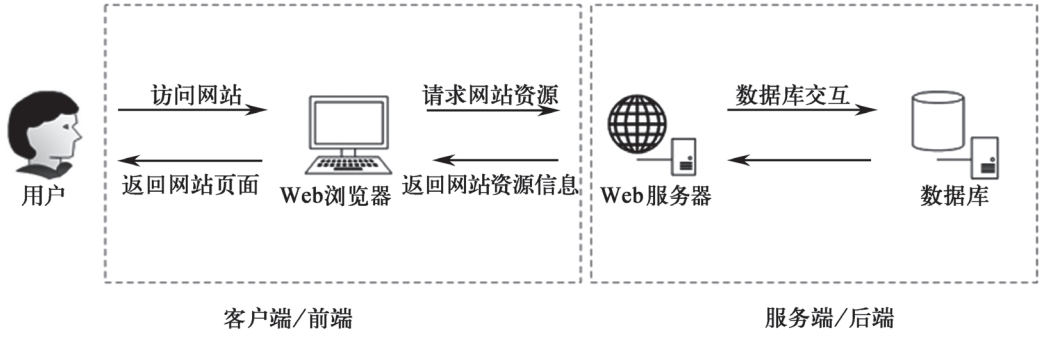

图 1-1　PHP的工作流程

1.3 任务实现

1.3.1 搭建PHP开发环境

刚开始接触PHP时，搭建PHP开发环境较为复杂，需要单独安装和配置Apache、PHP及MySQL。在Windows系统下开发，可以选择 phpStudy、WAMP（Windows + Apache + MySQL + PHP）、AppServ等集成安装环境，一次性安装，无需配置即可使用，非常方便、实用。

1. 安装 phpStudy

步骤01　首先从官方网站（https://www.xp.cn/）下载 phpStudy的安装程序。下面以Windows 10（64位）系统、phpStudy 2018为例，讲解phpStudy的安装步骤。

步骤02　phpStudy安装文件的压缩包下载完毕，对该压缩包进行解压缩，然后双击phpStudy2018.exe 安装文件，弹出如图1-2所示的对话框，将默认安装路径“C:\ phpStudy”修改为“D:\phpStudy”，单击“是”按钮，解压进度如图1-3所示。

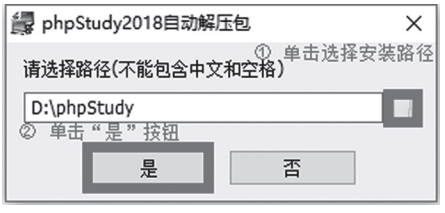

图 1-2　解压对话框

图 1-3　解压进度

步骤03　解压完成，进入phpStudy的启动界面，启动完成后的效果如图1-4所示。单击启动界面中的“启动”按钮，如有防火墙开启，在Apache服务和MySQL服务启动中会提示是否信任Apache HTTP Server、mysqld运行，如图1-5、图1-6所示，勾选“专用网络，例如家庭或工作网络”复选框，并单击“允许访问”按钮，成功之后，即完成了 phpStudy的安装

操作，启动完成界面如图1-7所示，在任务栏的托盘中增加了phpStudy图标和桌面快捷方式。

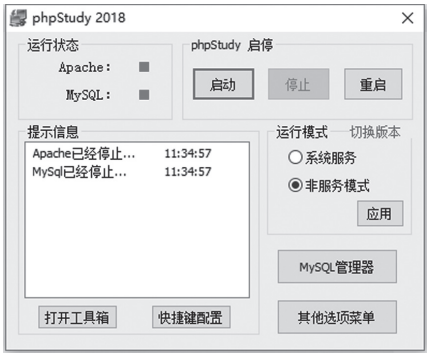

图 1-4　启动完成后的效果

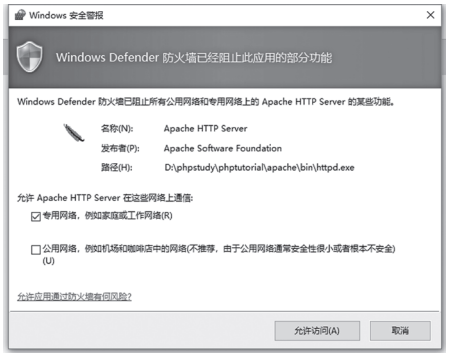

图 1-5　Apache网络安全提示框

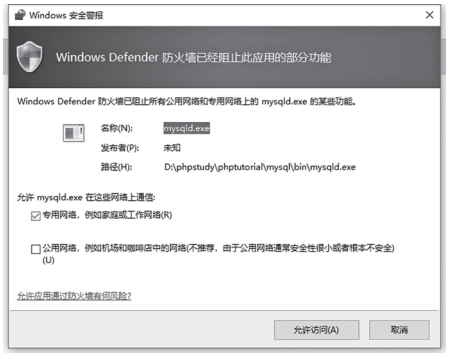

图 1-6　MySQL主程序安全提示框

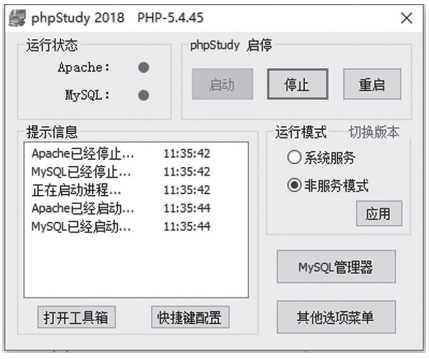

图 1-7　启动完成界面

步骤04　打开浏览器，在地址栏中输入“http://localhost/l.php”或者“http://127.0.0.1/l.php”，然后按Enter键，如果出现如图1-8所示的页面，说明phpStudy安装成功。

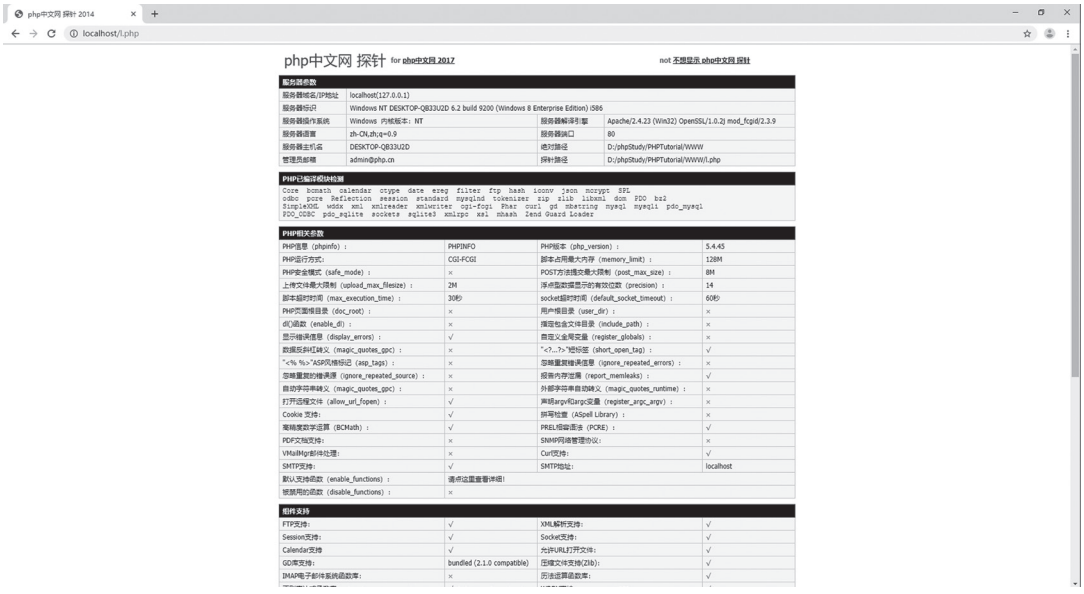

图 1-8　phpStudy安装成功运行页面

步骤05　由于phpStudy默认是使用PHP 5.4.45版本，如果需要修改，则单击“切换版本”，选择自己想用的版本即可，如图1-9、图1-10所示。

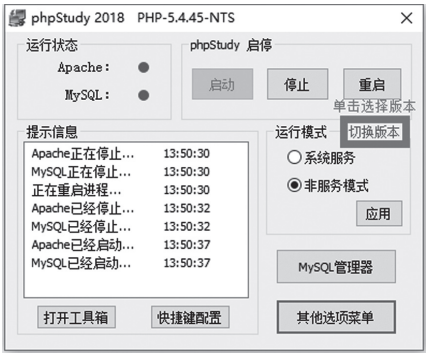

图 1-9　单击“切换版本”

图 1-10　切换版本后的界面

注意　如果提示“没有安装VC9运行库”，则需要到微软官方网站下载。PHP 5.5、PHP 5.6是用VC11编译的，使用时必须安装VC11运行库；PHP 7.0、PHP 7.1是用VC14编译的，使用时必须安装VC14运行库。

phpStudy启动失败时的解决方法如下。

（1）失败原因：防火墙拦截。

为减少出错，安装路径不得有汉字。如果有防火墙开启，会提示是否信任Apache HTTP Server、mysqld运行，请全部选择允许。

（2）失败原因：80端口已经被其他程序占用，如IIS等程序。

由于端口问题而无法启动时，请在phpStudy界面中执行 “其他选项菜单”→“环境端口检测”→“环境端口检测”→“检测端口”→“尝试强制关闭相关进程并启动”操作，如图1-11所示。

注意　如果强制关闭相关进程并在启动后仍无法启动 phpStudy，请检测端口被哪一个进程所占用，然后手动关闭该进程。

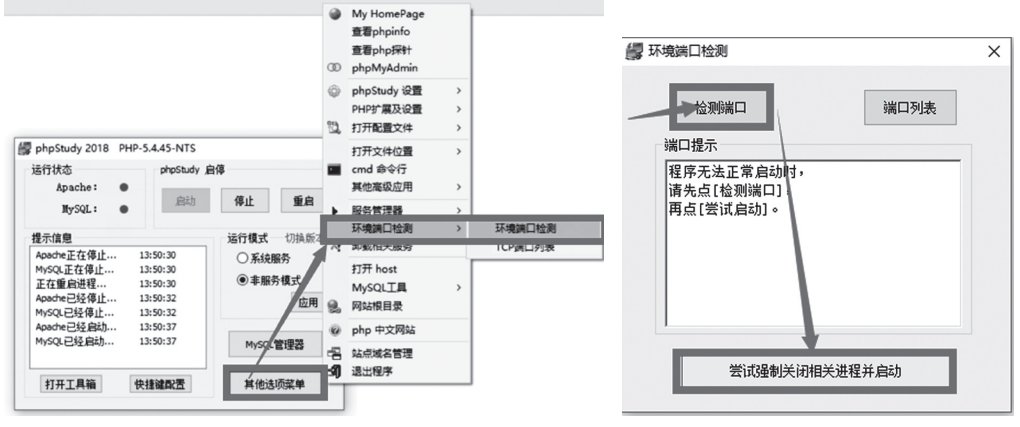

图 1-11　phpStudy检测端口

2. 启动与停止 PHP 服务器

PHP服务器主要包括Apache服务器和MySQL服务器。重新启动计算机后，在默认状态下，Apache服务和MySQL服务是停止的。下面介绍在 phpStudy中启动与停止这两种服务器的方法。

1）实现开机自动启动服务

在 phpStudy的启动界面中，只需单击“系统服务”单选按钮，然后单击“应用”按钮，即可实现开机自动启动服务的功能，如图1-12所示。

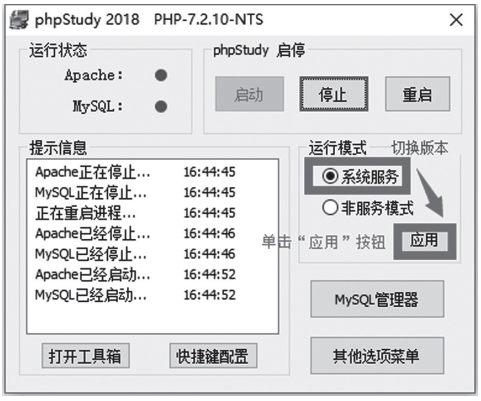

图 1-12　开机自动启动服务设置

2）启动和停止服务器

双击 phpStudy快捷方式图标，打开 phpStudy，界面如图1-13所示，单击“启动”按钮即可同时启动Apache服务和MySQL服务，启动后的状态如图1-14所示。如果想要停止Apache服务和MySQL服务，只需要单击图1-14中的“停止”按钮即可。另外，单击图1-14中的“重启”按钮，即可重启Apache和MySQL两种服务。

图 1-13　phpStudy的打开界面

图 1-14　phpStudy启动服务

1.3.2 扩展设置和开发环境关键配置

1. 开启 PHP 扩展设置

在开发某些项目时，会使用PHP扩展库中的扩展。通常情况下，如果要开启某个扩

展，需要修改php.ini文件或者进行可视化设置。具体方法如下。

（1）修改php.ini文件：用记事本或者其他编辑器打开php.ini文件（X:\phpStudy\PHPTutorial\php\php-A\php.ini，其中“X”为自选安装盘符，“A”为自选当前PHP版本），找到需要打开的扩展，将其前面的“;”去掉，保存文件并重启Apache，即可打开该扩展，如图1-15所示。

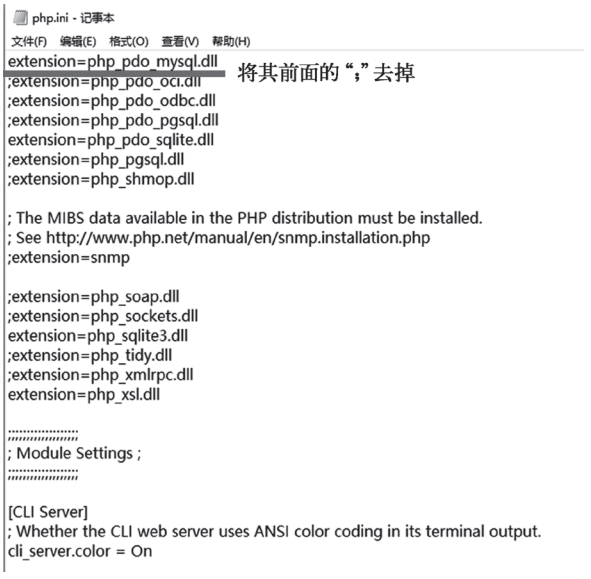

图 1-15　打开扩展

（2）使用phpStudy可视化开启扩展：操作过程变得非常简单，只需执行“其他选项菜单”→“PHP扩展及设置”→“PHP扩展”操作，然后选择相应的扩展即可，如图1-16所示。

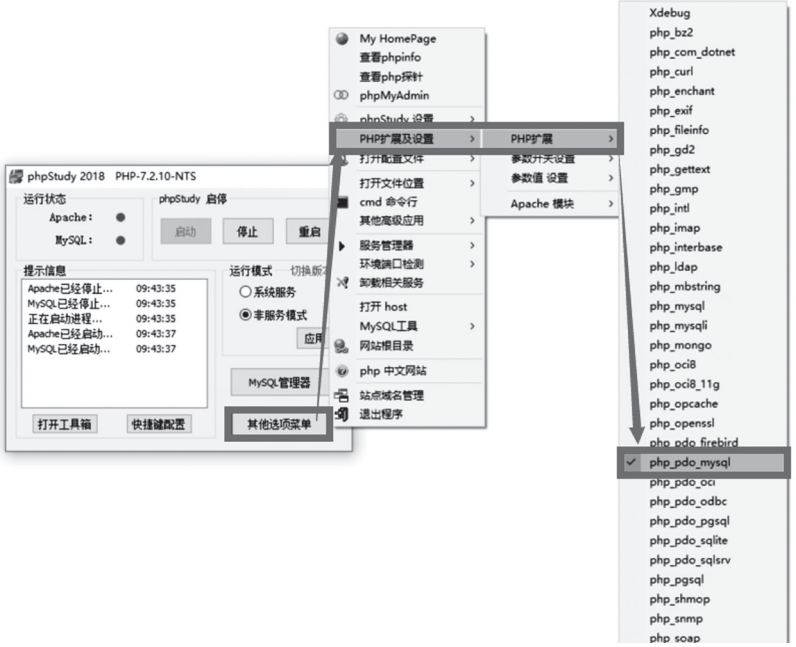

图 1-16　扩展设置

2. PHP 开发环境关键配置

1）修改Apache服务端口号

phpStudy是一个PHP调试环境的程序集成包。它集成了最新的Apache＋PHP＋MySQL＋phpMyAdmin＋zendOptimizer。安装成功后，Apache服务的端口号默认为80。有的初学者会遇到这样的问题，Apache服务器运行几秒后会停止，其中最有可能的原因是默认的80端口号被占用。此时需要修改Apache服务的端口号，可以通过以下操作完成。

步骤01　执行“其他选项菜单”→“打开配置文件”操作，选择“httpd.conf”，将默认的端口号修改为没有被占用的任意端口号，在此修改为“89”，如图1-17所示。

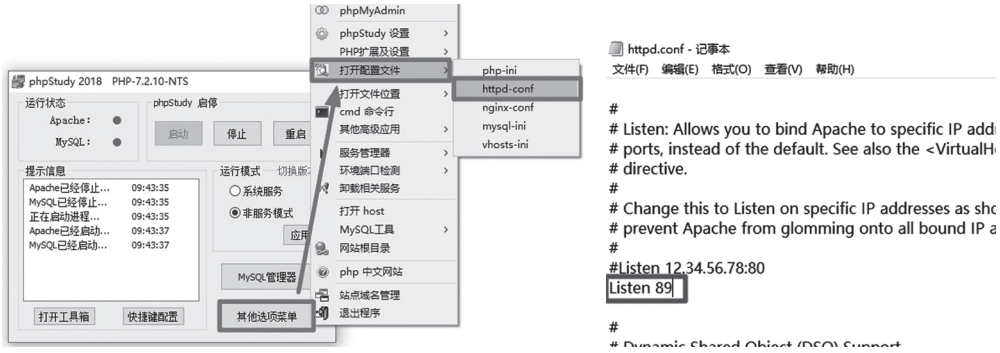

图 1-17　修改端口

步骤02　保存httpd.conf配置文件，重新启动Apache服务器，使新的配置生效。以后访问Apache服务时，需要在浏览器的地址栏中加上Apache服务的端口号（如http://localhost:89/。若端口号没有修改，则直接访问http://localhost/。本书相关讲解都是基于没有修改端口号）。

2）设置网站起始页面

Apache服务器允许用户自定义网站的起始页及其优先级，方法如下。

步骤01　打开httpd.conf配置文件，查找关键字“DirectoryIndex”，在“DirectoryIndex”的后面就是网站的起始页及优先级，如图1-18所示。由图可见，默认的网站起始页及优先级为index.html、index.php、index.htm、l.php。Apache的默认显示页为index.html。

步骤02　在浏览器的地址栏中输入“http://localhost/”时，Apache会首先查找访问服务器主目录下的index.html文件。如果没有index.html，系统会自动依次查找index.php、index.htm、l.php作为默认首页。

图 1-18　设置网站起始页

3）设置Apache服务器主目录

默认情况下，浏览器访问的是“X:\phpStudy\PHPTutorial\WWW”目录下的文件，WWW目录被称为“Apache服务器的主目录”。用户也可以自定义Apache服务器的主目录，方法如下。

步骤01　打开httpd.conf配置文件，查找关键字“DocumentRoot”，如图1-19所示。

步骤02　修改httpd.conf配置文件。例如，设置目录“D:\WWW”为Apache服务器的主目录，如图1-20所示。

步骤03　保存修改的配置文件“httpd.conf”后，必须重启Apache，配置文件才生效。在浏览器的访问地址栏中输入“http://localhost/l.php”，访问的就是Apache服务器主目录“D:\WWW”下的l.php文件。

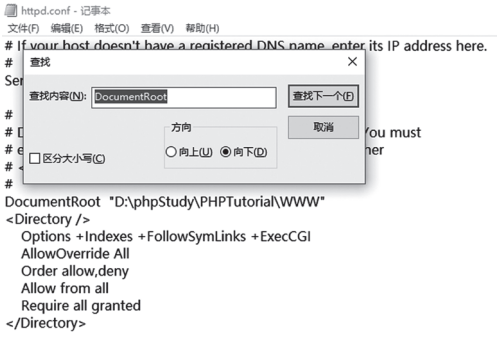

图 1-19　查找关键字

图 1-20　修改Apache服务器的主目录

4）PHP的其他常用配置

（1）语言的相关配置。

- engine：设置PHP引擎是否可用，默认值为On。若将其设置为Off，则无法使用PHP。

  配置示例：

```
engine = On
```

- short_open_tag：是否允许PHP脚本使用短开放标记，将“<?php ?>”改为“<? ?>”。但这个语法与XML相同，在某些情况下可能会导致问题，因此，一般建议关闭该项。

  配置示例：

```
short_open_tag = Off
```

- asp_tags：是否支持ASP风格的脚本定界，即“<% %>”。

  配置示例：

```
asp_tags = On
```

（2）错误信息的相关配置。

- log_errors：PHP错误报告日志功能的开关。

  配置示例：

```
log_errors = On      //打开PHP错误报告的日志功能
```

- error_log：PHP错误报告日志文件的路径。

  配置示例：

```
error_log = "D:\php_errors.log"
```

1.3.3 常用代码编辑工具

常用代码编辑工具应具备的主要功能如下。

（1）强有力的开发环境：通过完全的PHP支持、编码分析器、编码组合功能、语法检索、项目管理器、编码编辑器、绘图调试器（向导），可以提高生产力。

（2）超强的智能编码：具备新的和更优秀的分析和优化工具，就像PHP编码检测器。

（3）PHP的标准记录工具，PHP文档记录器：非常容易记录PHP代码、程序应用和方案。

（4）FTP和SFTP组合简化配置：使开发者从远程服务器上安全、灵活地上传和下载项目文件。

下面介绍几款常用的代码编辑工具。

1. Dreamweaver

Dreamweaver是一款专业的网站开发编辑器。从MX版本开始，Dreamweaver就支持PHP＋MySQL的可视化开发，对于初学者而言，确实是比较好的选择。因为如果是一般性开发，几乎是可以不用一行代码也可以写出一个程序，而且都是所见即所得，所包含的特征包括语法加亮、函数补全、形参提示等。不过Dreamweaver生成的代码比较复杂，安全性一般，在手写方面的方便程度也一般，在调试环境方面的表现不尽如人意，不太适合比较复杂的编程，但对于初学者是再好不过了。

下载地址：http://www.adobe.com/

本书所介绍的网页和实例都是使用Dreamweaver CS6编辑的。

2. Zend Studio

这是Zend Technologies开发的PHP语言集成开发环境（Integrated Development Environment，IDE），也支持HTML和js标签，但只对PHP语言提供调试支持。因为是同一家公司的产品，所以提供的Zend Framework方面的支持比其他软件好。

下载地址：http://www.zendstudio.net/zend-studio-all-in-one-download/

3. Eclipse

Eclipse是一款支持各种应用程序开发工具的编辑器，为程序设计员提供了许多强大的功能，以一种友好的集成开发环境为各种类型的用户提供了一系列针对Web开发的可用工具。它包括适用于各种语言、向导和内置应用程序以简化开发的源代码和图形编辑器，PHPEclipse是Eclipse的一个插件，提供了包括PHP语法分析、运行、调

试等功能的集成开发环境。它基于Eclipse的插件机制，即插即用，配置和使用都非常方便。

下载地址：http://www.eclipse.org

4. PhpStorm

PhpStorm是一款由JetBrains公司推出的商业PHP集成开发工具，被誉为最好用的PHP IDE。它是一种轻量级且便捷的PHP IDE，旨在提高用户效率，可深刻理解用户的编码，提供智能代码补全、快速导航，以及即时错误检查。

下载地址：

https://intellij-support.jetbrains.com/hc/en-us/community/topics/200367219-PhpStorm

完成前面内容的学习，下面动手实践一下：搭建环境测试成功后，如果将根目录的路径名称改为中文，会是怎样的结果？

项目 2

在线相册

学习目标

- 了解PHP的语法基础、标记、变量及常量，熟悉PHP语言的特点。
- 掌握PHP的数据类型，熟悉PHP的运算符及优先级的运用。
- PHP程序应遵循的编码规范。
- PHP的数据输出。

技能要点

- PHP的标记及注释。
- PHP的数据类型。
- PHP的编码规范。

2.1 项目描述

随着生活水平的逐步提高，旅游摄影成为人们必不可少的放松项目。虽然照片最后会存入计算机里，但是时间一长，照片一多，这些可以勾起回忆的“资料”就会显得杂乱无章，既不方便欣赏，也不方便管理。在线相册具有欣赏和传播方便、界面美观等特点，可以很好地管理存入的照片，不失为欣赏、保存照片的最佳相册工具。在线相册系统平台因此应运而生，实现效果如图2-1所示。

图 2-1　在线相册

利用PHP可将通过表单输入的相册名称创建并显示在页面中，在QQ空间或其他类似空间中也有这样的功能。从图2-1可以看出，实现这个在线相册，首先要清楚项目最终要呈现的效果，并用DIV＋CSS实现最终页面的静态页面效果。在操作时需要注意以下几点。

（1）将最终页面的静态页面效果事先制作完成。

（2）提交表单的方式。

（3）利用PHP获取表单值的方法。

2.2 涉及知识

2.2.1 PHP需要标记符

PHP需要标记符吗？带着这样的问题，正式踏上PHP的旅途。答案是肯定的，PHP需要标记符。由于PHP是嵌入式脚本语言，PHP标记符能够让Web服务器识别PHP代码的开始和结束，开始和结束标记之间的所有文本都会被解释为PHP代码。PHP提供了4种标记风格。如果要使用简短风格或者ASP风格（由于和ASP、JSP中的标记冲突，不推荐使用），就要在php.ini文件中将 `Short_open_tag=OFF` 或 `Asp_tags=OFF` 代码段中的

“OFF”改为“ON”，然后保存php.ini文件并重启Apache服务。

1. XML 标记风格（标准标记，推荐使用）

```
<?php
   echo "这是XML标记的风格,标准风格";
?>
```

此标记是本书使用的标记风格。在实际开发中推荐开始标记顶格写。如果一个文件是纯PHP代码，可以不写结束标记。

2. 脚本标记风格

```
<script language="php">
   echo '这是脚本风格的标记';
</script>
```

3. 简短标记风格

```
<?echo '这是简短风格的标记';?>
```

4. ASP 标记风格

```
<%
   echo '这是ASP风格的标记';
%>
```

2.2.2 PHP的注释及应用

在网站的开发过程中，为了便于阅读代码、后期维护，以及编写某行代码或功能模块，最好在代码的尾部或者上方添加注释以进行解释、说明，例如说明代码或函数的用途、时间、作者等。在执行程序时，注释部分会被解释器忽略，不会影响程序的执行。

PHP支持如下3种风格的程序注释。

1. 单行注释（//）

后方注释如下。

```
<?php
   echo 'php旅程'; //这是单行注释（标记后的内容不会被输出）
?>
```

上方注释如下。

```
<?php
   require'inc/init.php';
    //判断相册是否存在
    if ($id && !$data) {
  exit('相册不存在! ');
    }
?>
```

2. 多行注释（/*…*/）

块注释如下。

```
<?php
    /* 这是
       多行的
   注释 */
   echo '只有这个会显示';
?>
```

函数注释如下。

```
/**
 * 保存错误信息
 * @param string $str 错误信息
 * @return string 错误信息
 */
function errors($str = null)
{
   static $errors = null;
   return $str ? ($errors = $str) : $errors;
}
```

多行注释不能嵌套。注释中不能出现“?>”，否则解释器会认为PHP脚本结束了。

3. Shell 风格注释（#）

```
<?php
   echo '这是shell风格的注释';  #这句内容看不到
?>
```

2.2.3 PHP语句和语句块

PHP程序由一条或多条PHP语句构成，每条语句都以英文分号“;”结束。在书写PHP代码时，一般一条PHP语句占用一行；如果多条PHP语句之间存在某种联系，例如条件判断、循环语句、函数等，可以使用“{”和“}”将这些PHP语句包含起来形成一个语句块，语句块一般不会单独使用。

```
<?php
function errors($str = null)
{
```

```
    static $errors = null;
    return $str ? ($errors = $str) : $errors;
}
?>
```

2.2.4 标识符与关键字

在开发过程中，经常需要在程序中定义变量名、函数名、类名、方法名等一些标记名称，这些标记名称就是所谓的标识符。在PHP中，定义标识符要遵循一定的规则，具体规则如下。

（1）标识符可以由一个或多个字符组成，必须以字母或下划线开头。此外，标识符只能由字母、数字、下划线字符，以及127～255的其他ASCII字符组成。

（2）标识符区分大、小写，函数例外。因此，变量$name不同于变量$Name、$nAME或$NAme。

（3）标识符可以是任意长度，这样程序中就能通过标识符名准确地描述其用途。

（4）标识符名不能与任何PHP预定义的关键字相同。

在开发过程中还会运用关键字。这些关键字在 PHP 中被事先定义好并具有特殊的意义。它们是语言结构的一部分，不能使用它们其中的任何一个作为常量、方法名或是类名。虽然可以将它们作为变量名使用，但这样容易导致混淆，因此不建议使用。这些关键字也被称为“保留字”。关键字如表2-1所示。

注意 从PHP 7.0.0开始，这些关键字允许被用作类的属性、常量及类的方法名，或者接口名和traints名，除了class不能被用作常量名。

表 2-1 关键字

__halt_compiler()	abstract	and	array()	as
break	callable (as of PHP 5.4)	case	catch	class
clone	const	continue	declare	default
die()	do	echo	else	elseif
empty()	enddeclare	endfor	endforeach	endif
endswitch	endwhile	eval()	exit()	extends
final	finally (从PHP 5.5开始)	for	foreach	function
global	goto (从PHP 5.3开始)	if	implements	include
include_once	instanceof	insteadof (从PHP 5.4开始)	interface	isset()

续表

list()	namespace（从PHP 5.3开始）	new	or	print
private	protected	public	require	require_once
return	static	switch	throw	trait（从PHP 5.4开始）
try	unset()	use	var	while
xor	yield（从PHP 5.5开始）			

2.2.5 变量与常量

在代数学科中，使用字母（例如x）来保存值（例如5）；而在编程中，这里的“x”就是变量。变量是由“$”符号和变量名组成的，其值可以改变，其命名规则与标识符相同。变量为开发人员提供了一个有名称的内容存储区，程序可以通过变量名对内存存储区进行读、写操作，也可以理解为变量是用于存储可变数据的“容器”。

与其他语言不同，PHP是弱类型语言，因此，变量在使用前不需要先声明就可以直接赋值使用。在PHP中，变量分直接赋值、传值赋值、引用赋值3种。具体如下。

1. 直接赋值

```
<?php
    $host='127.0.0.1';    //定义变量$host,并赋值为127.0.0.1
    $dbuser='root';   //定义变量$dbuser,并赋值为root
?>
```

从上面的例子可以看出，直接赋值就是用“=”直接将值赋给变量。若要输出变量的值，可以使用echo，如echo $host。

注意 美元符号“$”是变量的标识符，所有变量都是以“$”符开头的。无论是声明变量还是调用变量，都应该使用“$”符。

2. 传值赋值

传值赋值是使用“=”将一个变量的值赋给另一个变量。“变量间的赋值”是指赋值后两个变量使用各自的内存，互不干扰。

```
<?php
 $host='127.0.0.1';          //为变量$host赋值127.0.0.1
 $host1=$host;               //使用$host初始化$host1
 $host="192.168.1.1";    //改变变量$host的值192.168.1.1
?>
```

3. 引用赋值

“引用赋值”是指用不同的名字访问同一个变量内容，当改变其中一个变量的值时，另一个变量的值也随之发生变化。引用赋值将一个“&”符号简单地加到将要赋值的变量前。具体使用如下。

```
<?php
 $host='127.0.0.1';          //定义变量$host并赋值127.0.0.1
 $local=&$host;        //定义变量$local,并将$host值引用赋值给$local
 $host="192.168.1.1";      //变量$host重新赋值为192.168.1.1
 echo $local;            //输出$local的值,其结果为192.168.1.1
?>
```

在学习数学时会遇到圆周率π，它是固定不变的。在开发过程中，如果同样要定义一个固定不变的量，可以使用变量吗？除了变量，实际在PHP中还可以使用常量来保存数据。常量一旦被定义，就不能被修改或者重新定义，脚本在运行过程中始终保持值不变，这时就用常量来保存。常量的具体定义和使用如下所示。

1）define()函数

语法格式如下。

```
define(name,value,case_insensitive)
```

- name：必要参数，规定常量的名称。
- value：必要参数，规定常量的值。
- case_insensitive：可选参数，规定常量的名称是否对大、小写敏感。若设置为true，则对大、小写不敏感；默认是 false（大、小写敏感）。

如果需要输出常量，直接用constant()函数，并设置常量名称参数项即可。

```
<?php
  define('DBNAME','album');       //定义常量DBNAME并赋值album
  echo DBNAME;             //使用echo输出DBNAME,其结果为album
 echo constant('DBNAME'); //使用constant函数获取常量输出DBNAME,其结果为album
?>
```

2）const关键字

```
<? php
  const PAI=3.14;          //定义名称为PAI的常量,赋值为3.14
  echo  PAI;             //输出结果为3.14
?>
```

除去以上自定义变量和常量外，PHP提供一些常用的系统常量（如表2-2所示）和全局变量（如表2-3所示）。

表 2-2　系统常量

常 量 名	用　　途	注　　意
__FILE__	当前PHP文件的相对路径	前后都是双下划线
__LINE__	当前PHP文件中所在的行号	前后都是双下划线
__CLASS__	当前类名，只对类起作用	前后都是双下划线
PHP_VERSION	PHP版本号	大写
PHP_OS	当前操作系统类型	大写
M_PI	圆周率常量值	大写
M_E	科学常数e	大写
M_LOG2E	代表$\log_2 e$，以2为底e的对数	大写
M_LN2	2的自然对数	大写
M_LN10	10的自然对数	大写
E_ERROR	最近的错误之处	大写
E_WARNING	最近的警告之处	大写
E_PARSE	剖析语法有潜在问题之处	大写
__METHOD__	表示类方法名，例如A::test	大写

表 2-3　全局变量

变 量 名	用　　途
$_SERVER	返回服务器的相关信息，返回一个数组
$_GET	所有GET请求过来的参数
$_POST	所有POST过来的参数
$_COOKIE	所有HTTP提交过来的cookie
$_FILES	所有HTTP提交过来的文件
$_ENV	当前的执行环境信息
$_REQUEST	相当于$_POST、$_GET、$_COOKIE提交过来的数据，因此，这个变量不值得信任
$_SESSION	session会话变量

2.2.6 PHP的数据类型

在计算机中，以位（0或1）表示数据。数据是计算机操作的对象，每一个数据都有其类型，具有相同数据类型的数据才能进行运算操作。PHP的数据类型可以归纳为标量数据类型、复合数据类型和特殊数据类型3种。

1. 标量数据类型

标量数据类型是数据结构中最基本的单元，只能存储一个数据。PHP中的标量数据类

型包括4种，如表2-4所示。

表 2-4　标量数据类型

类　型	描　述
Boolean（布尔型）	布尔型是最简单的数据类型，只有两个值 false（假） 和true（真）
String（字符串型）	字符串就是连续的字符序列，可以是计算机能表示的一切字符的集合
Integer（整型）	整型数据类型只能包含整数，这些数据类型可以是负整数、正整数或者零
Float（浮点型）	浮点型类型用于存储数字，和整型不同的是，浮点型可以有小数点

- 布尔型（Boolean）：布尔型数据类型保存一个 true值或者 false值，其中true和false是PHP的内部关键字。设定一个布尔型的变量，只需将 true 或者false赋值给变量即可。布尔型数据类型通常用于逻辑判断，是PHP中最常用的数据类型之一。示例如下。

```
<?php
  $flag=true;          //定义一个变量$flag并赋值为true
  $result=false;       //定义一个变量$result并赋值为false
?>
```

注意：在PHP中不是只有false值才为假，在0、0.0、“0”、空白字符串（“”）、只声明没有赋值的数组等这些特殊情况下，boolean值也被认为是false。

- 字符串型（String）：字符串是由连续的字母、数字或者字符组成的字符序列。简单来说，就是想表达的一切让其他人看到的字符。在PHP中，通常使用单引号或双引号表示字符串。示例如下。

```
<?php
  $name='张三';
  $sex='男';
  $str='个人信息';
  echo $name.'的'.$str.'性别$sex<br>';
  echo $name.'的'.$str."<br>性别$sex";
  echo $name.'的'.$str."<br>性别:$sex<br>年龄";
?>
```

从上述代码的运行结果可以看出，在双引号字符串中变量$sex会被解析为变量值“男”，而在单引号字符串中输出原样。在上述代码中，“.”是字符串连接符，“<br>”是换行标签，“echo”是PHP的输出语句，将文本内容显示在浏览器中。双引号字符串还支持换行符“\n”、制表符“\t”等转义符的使用，单引号字符串只支持“'”和“\”的转义。其中，转义字符如表2-5所示。

表 2-5　转义字符

转义字符	含　义	转义字符	含　义
\n	换行	\\	反斜杠
\r	回车	\$	美元符号
\t	水平制表符	\"	双引号

从 echo $name.'的'.$str."
性别:$sex
年龄"; 这行代码的最终执行结果可以看出，性别后面并没有出现$sex变量的值和“年龄”二字，原因是在双引号字符串中输出变量时，会出现变量名界定不明确的问题，在这种情况下就可以使用英文“{}”对变量进行界定，建议在编程中使用。示例如下。

```
<?php
   $name='张三';
   $sex='男';
   $str='个人信息';
   echo $name.'的'.$str.'性别$sex<br>';
   echo $name.'的'.$str." <br>性别$sex";
   echo $name.'的'.$str."<br>性别:{$sex}年龄";
?>
```

PHP还有另外一种创建字符串的方法，就是使用定界符“<<<”。示例如下。

```
<?php
$str=<<<"ABC"
……
ABC;
echo $str;
?>
```

注意　是在“<<<”之后提供一个标识符，然后是字符串，最后用统一的标识符结束字符串。

- 整型（Integer）：整型数据类型就是通常所说的整数，它只能是整数，生活中的123或者－123，年龄20，都表示整型。这些都是十进制，还可以写为八进制、十六进制。如果使用八进制表示，数字前面必须加“0”；如果使用十六进制表示，数字前面需要加“0x”。整型数据类型支持的最大整数与平台的系统环境有关，一般是$\pm 2^{31}$。如果超过此限制，整型数据类型将转换为浮点型数据类型。示例如下。

```
<?php
   $m=0123;      //八进制
   $m2=123;      //十进制
```

```
    $m3=0x123;    //十六进制
    echo "八进制的结果是：{$m}<br>";
    echo "十进制的结果是：{$m2}<br>";
    echo "十六进制的结果是：{$m3}";
?>
```

- 浮点型（Float）：浮点型（浮点数，单精度数，双精度数或实数），也就是通常所说的小数，可以用小数点法或者科学记数法表示。使用科学记数法时可以使用小写的“e”，也可以使用大写的“E”。示例如下。

```
<?php
  $num_float = 1.234;     //小数点法
  $num_float = 1.2e3;     //科学记数法,小写e
  $num_float = 7.0E-10;      //科学记数法,大写E
?>
```

2. 复合数据类型

复合数据类型将多个简单数据类型组合在一起，存储在一个变量中，包括数组和对象两种，如表2-6所示。

表 2-6　复合数据类型

类　型	描　述
Array（数组）	一组数据的集合
Object（对象）	对象是类的实例。使用关键字new来创建

- 数组（Array）：数组是一个能在单个变量中存储多个值的特殊变量。数组是一组数据的集合，将一系列数据组织起来，形成一个可操作的整体。数组中的每个数据都被称为“一个元素”，元素包括键名（即索引）和值两部分。元素的键名可以由数字或字符串组成，元素的值可以是多种数据类型（标量数据，数组，对象，资源，以及PHP支持的其他语法结构等）。定义$arr_ext变量作为数组名，并为数组赋值。示例如下。

```
<?php
     $arr_ext=['jpg', 'jpeg', 'png'];          //第一种
     $arr_ext=array('jpg', 'jpeg', 'png');      //第二种
?>
```

上面代码中的键名为数字，“jpg”“jpeg”“png”为元素的值。

- 对象（Object）：要创建一个对象，首先要创建一个类，类需要用class关键字定义。创建好类后，便可以使用new关键字实例化类的对象，然后访问对象的属性、方法等。编程语言用到的方法有面向过程和面向对象两种。

3. 特殊数据类型

除上面介绍的两大数据类型外，PHP还支持用在特殊方面的数据类型，主要包括资源

类型和空值类型。

- 资源类型（resource）：资源是由专门的函数来建立和使用的。在使用资源时要及时释放不需要的资源，如果忘记释放，系统就会自动回收，以避免内存消耗殆尽。
- 空值类型（NULL）：空值类型只有一个值NULL，并不表示空格、零。它表示没有值。

2.2.7 PHP运算符

1. 算术运算符

算术运算符是用来处理加、减、乘、除运算的符号，也是最简单和最常用的运算符，如表2-7所示。

表 2-7 算术运算符

运 算 符	名 称	表 达 式	描 述	实 例	结 果
+	加	x+y	x和y的和	2+2	4
-	减	x-y	x和y的差	5-2	3
*	乘	x*y	x和y的积	5*2	10
/	除	x/y	x和y的商	15/5	3
%	模（除法的余数）	x%y	x除以y的余数	5%2 10%8 10%2	1 2 0
-	取反	-x	x取反	-2	-2
.	拼接	a.b	连接两个字符串	"Hi"."Ha"	HiHa

注意 在进行四则混合运算时，运算顺序依然遵循数学中的“先乘除后加减”原则；在进行取模运算时，运算结果的正负取决于被模数（%左侧的数）的符号，与模数（%右侧的数）的符号无关。

2. 赋值运算符

赋值运算符是一个二元运算符，它意味着左操作数被设置为右侧表达式的值。也就是说，“$x = 5”中x的值是5。具体如表2-8所示。

表 2-8 赋值运算符

运 算 符	表 达 式	等 同 于	描 述
=	x=y	x=y	左操作数x被设置为右侧表达式y的值
+=	x+=y	x=x+y	左操作数x被设置为x+y的值
-=	x-=y	x=x-y	左操作数x被设置为x-y的值
*=	x*=y	x=x*y	左操作数x被设置为x*y的值

续表

运 算 符	表 达 式	等 同 于	描　　述
/=	x/=y	x=x/y	左操作数x被设置为x/y的值
%=	x%=y	x=x%y	左操作数x被设置为x%y的值
.=	a.=b	a=a.b	左操作数a被设置为a.b（两个字符串连接）的值

3. 递增 / 递减运算符

PHP中递增/递减运算符也称自增/自减运算符，可以看作一种特定形式的复合赋值运算，如表2-9所示。

表 2-9　递增/递减运算符

运 算 符	表 达 式	等 同 于	描　　述
++	++x	预递增	x加1，然后返回x
++	x++	后递增	返回x，然后x加1
--	--x	预递减	x减1，然后返回x
--	x--	后递减	返回x，然后x减1

4. 比较运算符

比较运算符用来对两个变量或表达式进行比较，其结果是布尔类型的值true或false，如表2-10所示。

表 2-10　比较运算符

运 算 符	名　　称	表 达 式	描　　述	实　　例
==	等于	m==n	如果m等于n，则返回true	6==8返回false
===	恒等	m===n	如果m等于n，且它们类型相同，则返回true	8===" 8 "返回false
!=	不等于	m!=n	如果m不等于n，则返回true	6!=8返回true
<>	不等于	m<>n	如果m不等于n，则返回true	6<>8返回true
!==	不恒等	m!==n	如果m不等于n，或它们类型不相同，则返回true	6!=="6"返回true
>	大于	m>n	如果m大于n，则返回true	6>8返回false
<	小于	m<n	如果m小于n，则返回true	6<8返回true
>=	大于等于	m>=n	如果m大于或者等于n，则返回true	6>=8返回false
<=	小于等于	m<=n	如果m小于或者等于n，则返回true	6<=8返回true

5. 逻辑运算符

主要用于程序开发中的逻辑判断，其返回值类型为布尔类型，如表2-11所示。

表 2-11　逻辑运算符

运 算 符	名　　称	表 达 式	描　　述	实　　例
and	与	m and n	如果m和n都为true，则返回true	m=8 n=5 (m<9 and n>3) 返回true
or	或	m or n	如果m和n至少有一个为true，则返回true	m=8 n=5 (m==6 or n==5) 返回true
xor	异或	m xor n	如果m和n有且仅有一个为true，则返回true	m=8 n=5 (m==8 xor n==5) 返回false
&&	与	m&&n	如果m和n都为true，则返回true，与and相同，但优先级较高	m=8 n=5 (m<9 && n>3) 返回true
||	或	m||n	如果m和n至少有一个为true，则返回true，与or相同，但优先级较高	m=8 n=6 (m==5 || n==5) 返回false
!	非	!m	如果m不为true，则返回true	m=8 n=5 !(m==n) 返回true

虽然“&&”“||”与“and”“or”的功能相同，但前者比后者的优先级别高。

6. 运算符的优先级

在一个表达式中含有多个运算符时，要明确表达式中各个运算符参与运算的先后顺序，这种顺序被称为“运算符的优先级”。表2-12按照优先级从高到低的顺序列出了运算符，同一行中的运算符具有相同的优先级，此时它们的结合方向决定求值顺序。

注意　“左 =”表示从左到右，“右 =”表示从右到左。在表达式中还有一个优先级最高的运算符，即圆括号“()”。它可以提升其内运算符的优先级，通常能够增加代码的可读性。

表 2-12　运算符优先级

结合方向	运 算 符	附加信息
无	clone new	clone和new
左	[ ]	array()
右	++　--　~　(int)　(float)　(string)　(array)　(object)　(bool)　@	类型和递增／递减
无	instanceof	类型
右	!	逻辑运算符

续表

结合方向	运 算 符	附加信息
左	* / %	算术运算符
左	+ - .	算术运算符和字符串运算符
左	<< >>	位运算符
无	== != === !== <>	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

2.2.8 流程控制

1. 条件控制语句

在生活中总会遇到需要进行判断和决策的情况，程序也是一样。例如，用户使用微信登录，可以用微信扫一扫；使用QQ登录，可以输入QQ号和密码；使用微博登录，可以输入微博名称和密码；使用手机号登录，则输入手机号和密码。这类判断，按照条件选择执行不同的代码片段，就是程序中的条件控制语句。条件控制语句主要有if、if...else、if...elseif...else和switch。

- if语句：if语句是最简单的条件判断语句，它对某段程序的执行附加一个条件。如果条件成立，就执行这段程序；否则就跳过这段程序，去执行后面的程序。语法格式如下。

```
if(expr) {
    //满足条件要执行的代码块
}
```

如果表达式expr的值为true，则执行“{}”内的代码块；否则就跳过该条语句往下执行。如果执行语句只有一条，“{}”可省略。

```
If(条件)
    //满足条件要执行的代码块
```

- if...else语句：有时程序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，这时可以使用if...else语句。语法格式如下。

```
if (expr)
{
  //条件成立时执行的代码一;
}
else
{
  //条件不成立时执行的代码二;
}
```

如果表达式expr的值为true，则执行代码一；否则就执行代码二。如果执行语句只有一条，“{}”可省略。

- if...elseif...else 语句：If...elseif...else语句也是if语句的一种衍生，其作用是根据不同的条件执行不同的结果，类似于多个if...else语句嵌套。语法格式参见下例。

```
<?php
   $score=90;                    //定义变量成绩
   if ($score>0 && $score<60) {          //通过判断成绩返回值给出评价
      echo '成绩为不及格! ';
   } elseif ($score>=60 && $score<70)  { //通过判断成绩返回值给出评价
      echo '成绩为及格! ';
   } elseif ($score>=70 && $score<85)  { //通过判断成绩返回值给出评价
      echo '成绩为中等! ';
   } elseif ($score>=85 $& $score<=100)  { //通过判断成绩返回值给出评价
      echo '成绩为优秀! ';
   } else {//以上所有范围都不符合,给出提示
      echo  '成绩输入错误! ';
   }
?>
```

- switch语句：switch 语句和前面讲到的if...elseif...else语句类似，也是根据不同的条件执行不同的语句。和if...elseif...else 语句的不同点在于，switch常用于对不同的值进行判断，然后作出响应。语法格式参见下例。

```
<?php
   //提交按钮
   $action = input('post', 'action', 's');
```

```
    switch ($action){
        case 'new'://创建相册
            $name = input('post', 'new_name', 's');
            if(newalbum($id,$name)>1){
                //创建成功重定向
                header("location:index.php?id=".$id);
            };
        break;
        case 'upload'://上传图片
            $file=input($_FILES, 'upload', 'a');
            upload($id,$file);
        break;
        case 'del'://删除相册
            delete($t_id);
        break;
        case 'pic_cover'://设为封面
            picture_cover($t_id, $id);
        break;
        case 'pic_del'://删除图片
            picture_delete($t_id);
        break;
        default:
            echo '参数不对';
        break;
}
?>
```

通过上述代码可以看出switch的具体用法。利用PHP提供的自定义input函数来接收提交过来的值，再通过switch来判断具体执行什么流程。

注意

switch语句首先对一个简单的表达式 $action（通常是变量）进行一次计算。将表达式的值与结构中每个 case 的值进行比较。如果存在匹配，则执行与 case 关联的代码块。执行代码后，使用 break阻止代码跳入下一个 case 中继续执行。default 语句主要用于不存在匹配（即没有 case 为真）时执行。

2. 循环控制语句

在实际应用中经常会遇到一些并不复杂但需要反复多次处理的操作，使用顺序结构是很难实现的，也比较烦琐。因此，PHP提供了循环语句来实现其功能。

循环控制语句是指能够按照一定的条件重复执行某段功能代码的代码结构，分为while、do...while、for循环、foreach循环。

- while语句：while 语句的执行流程很简单，只要while循环条件的值为true就重复执行循环体中的语句。语法格式如下。

```
while（判断条件）{
    执行循环体语句;
}
```

- do...while语句：while语句的另一种表示形式，与while语句非常相似，只是do...while语句在循环的底部检测循环表达式，而不是在循环的顶部检测。do...while语句先执行语句，然后再对条件进行判断，如果条件值为false，则跳出循环。由此可以看出，该语句的循环体至少被执行一次。语法格式如下。

```
do{
    //执行循环体语句
}while(判断条件);
```

- for语句：for循环是一种在程序执行前就判断条件表达式是否为真的循环语句。如果条件为假（false），循环语句就不会执行。for语句适合重复执行预订次数的循环。语法格式如下。

```
for(expr1;expr2;expr3){
    //执行循环体语句
}
```

在for循环的括号内有3个表达式，其含义如下。

expr1表达式：初始表达式，用于变量的初始化。

expr2表达式：循环条件表达式，布尔类型的值，也可被称为“判定式”。

expr3表达式：循环后操作表达式，用于调整变量的值，也可被称为“更新表达式”。

- foreach语句：foreach 循环结构是遍历数组时常用的方法。foreach 仅能够应用于数组和对象，如果尝试应用于其他数据类型的变量或者未初始化的变量将发出错误信息。foreach 有以下两种语法格式。

第一种格式：

```
foreach (array_expression as $value){
    statement
}
```

此格式用法遍历 array_expression 数组时，每次循环将数组的值赋给 $value。

第二种格式：

```
foreach (array_expression as $key => $value){
    statement
}
```

此格式用法不仅将数组值赋给$value，还将键名赋给$key。

2.2.9 函数

把一段可以实现指定功能的代码封装在函数内，直接调用函数即可实现指定的功能。

函数可以分为以下几类：系统内置函数，自定义函数，变量函数。

1. 系统内置函数（PHP 常用自带函数）

（1）chunk_split：将字符串分割成小块。

语法格式如下。

```
chunk_split ( string $body [, int $chunklen = 76 [, string $end =
"\r\n" ]] ) : string
```

参数说明如下。

- $body：要分割的字符。
- $chunklen：分割的尺寸。
- $end：行尾序列符号。

（2）echo：输出一个或多个字符串。

语法格式如下。

```
echo ( string $arg1 [, string $... ] ) : void
```

示例如下。

```
echo "Hello World";
```

参数说明如下。

- $arg1：必填参数。一个或多个要输出的字符串。

（3）explode：使用一个字符串分割另一个字符串。

语法格式如下。

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

该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。

参数说明如下。

- $delimiter：指定分隔字符。
- $string：指定字符串。
- $limit：如果设置了 limit 参数并且是正数，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的剩余部分。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 limit 是0，则会被当作1。

（4）implode：将一个一维数组的值转化为字符串。

语法格式如下。

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

参数说明如下。

- $glue：默认为空的字符串。
- $pieces：想要转换的数组。

（5）htmlspecialchars：将指定的字符串预定义的字符（如表2-13所示）转换为 HTML 实体。

语法格式如下。

```
htmlspecialchars ( string $string [, int $flags = ENT_COMPAT | ENT_HTML401 [, string $encoding =ini_get(<<default_charset>>) [, bool $double_encode = TRUE ]]] ) : string
```

参数说明如下。

- $string：必需参数。规定要转换的字符串。
- $flags：可选参数。规定如何处理引号、无效的编码，以及使用哪种文档类型。
- $encoding：可选参数。一个规定了要使用的字符集的字符串。
- $double_encode：可选参数。布尔值，规定了是否编码已存在的 HTML 实体。默认为true，将对每个实体进行转换；为false则不会对已存在的 HTML 实体进行编码。

表 2-13　预定义的字符

原 字 符	替换后的实体
& （& 符号）	&amp;
" （双引号）	"，除非设置了 ENT_NOQUOTES
' （单引号）	设置了 ENT_QUOTES 后，'（如果是 ENT_HTML401），或者 '（如果是 ENT_XML1、 ENT_XHTML 或 ENT_HTML5）
< （小于）	&lt;
> （大于）	&gt;

（6）htmlspecialchars_decode：将特殊的 HTML 实体转换回普通字符。

语法格式如下。

```
htmlspecialchars_decode ( string $string [, int $flags = ENT_COMPAT | ENT_HTML401 ] ) : string
```

此函数与 htmlspecialchars( ) 刚好相反。它将特殊的HTML实体转换回普通字符。

参数说明如下。

- $string：特殊的HTML实体。
- $flags：可选参数。规定如何处理引号以及使用哪种文档类型。

  可用的引号类型如下。

  ENT_COMPAT：默认。仅解码双引号。

  ENT_ QUOTES：解码双引号和单引号。

  ENT_ NOQUOTES：不解码任何引号。

  规定使用的文档类型的附加flags 如下。

  ENT_ HTML401：默认。作为HTML 4.01处理代码。

  ENT_ HTML5：作为HTML 5处理代码。

  ENT_ XML1：作为XML 1处理代码。

ENT_ XHTML：作为XHTML处理代码。

（7）ltrim：删除字符串开头的空白字符（或其他字符）。

语法格式如下。

```
ltrim ( string $str [, string $character_mask ] ) : string
```

参数说明如下。

- $str：输入的字符串。
- $character_mask：通过参数 character_mask，可以指定想要删除的字符列表。具体做法是，简单地列出想要删除的所有字符。

（8）rtrim：删除字符串末端的空白字符（或者其他字符）。

语法格式如下。

```
rtrim ( string $str [, string $character_mask ] ) : string
```

参数说明如下。

- $str：输入的字符串。
- $character_mask：通过参数 character_mask，可以指定想要删除的字符列表。具体做法是，简单地列出想要删除的全部字符。

（9）trim：删除字符串首尾处的空白字符（或者其他字符）。

语法格式如下。

```
trim ( string $str [, string $character_mask = "<< \t\n\r\0\x0B>>" ]): string
```

参数说明如下。

- $str ：待处理的字符串。
- $character_mask ：可选参数，过滤字符也可由 character_mask 参数指定。一般要列出所有希望过滤的字符。

（10）md5：返回字符串的 MD5 散列值（32位）。

语法格式如下。

```
md5 ( string $str [, bool $raw_output = FALSE ] ) : string
```

参数说明如下。

- $str：原始字符串。
- $raw_output：如果可选的 raw_output 被设置为 true，那么 MD5 报文摘要将以16字节长度的原始二进制格式返回。

（11）money_format：数字格式转化成货币字符串。

语法格式如下。

```
money_format ( string $format , float $number ) : string
```

参数说明如下。

- $format：字符串由单个 % 字符、可选的标记（flags）、可选的字段宽度、可选的左侧精度、可选的右侧精度、必选的单个转化字符、必选的标记（flags）组成。可选多个标记，如表2-14所示。

表 2-14　可选多个标记

标　记	释　义
=f	字符：=，并紧跟一个字符（单字节）f，用于数字填充。默认的填充字符是空格
^	禁用分组字符（例如金额中的逗号，在本地区域设置 locale 中定义）
+ or (	正负数字的格式。使用+，将使用区域设置（locale）中相当于+和-的符号。 如果使用(，负数将被圆括号围绕。不设置的话，默认为+
!	不输出货币符号（例如 ¥）
-	有这个符号时，将使字段左对齐（填充在右边），默认是相反的，是右对齐的（填充在左边）
字段宽度w	十进制数值字符串的宽度。字段将右对齐，除非使用了-标记。默认值为 0
左侧精度#n	小数字符（例如小数点）前的最大位数（n）。常用于同一列中的格式对齐。如果位数小于n，则使用填充字符填满。如果实际位数大于n，此设置将被忽略 如果没用^标识禁用分组，分组分隔符会在添加填充字符之前插入（如果有的话）。分组分隔符不会应用到填充字符里，哪怕填充字符是个数字 为了保证对齐，出现在之前或者之后的字符，都会填充必要的空格，保证正、负情况下长度都一样
右侧精度.p	小数点后的一段数字（p）。如果p的值是0（零），小数点右侧的数值将被删除。如果不使用这个标记，默认显示取决于当前的区域设置。小数点后指定位数的数字，四舍五入格式化
转化符i	被格式化为国际货币格式（例如，locale是USA：USD 1 234.56）
转化符n	被格式化为国家货币格式（例如，locale是de_DE：EU1.234 56）
转化符%	返回字符%
%	返回字符%

- $number：需要格式化的数字。

（12）number_format：以千位分隔符方式格式化一个数字。

语法格式如下。

```
number_format(float $number[,int $decimals = 0,$dec_point,$thousands_
sep]) : string
```

参数说明如下。

- $number：想要格式化的数字。
- $decimals：想要保留的小数位数。
- $dec_point：指定小数点显示的字符。
- $thousands_sep：指定千位分隔符显示的字符。

（13）str_replace：字符串替换。

语法格式如下。

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

该函数返回一个字符串或者数组。如果 search 和 replace 都是数组，它们的值将会被依次处理。

参数说明如下。

- $search：查找的目标值，也就是needle。一个数组可以指定多个目标。
- $replace：search 的替换值。一个数组可以被用来指定多重替换。
- $subject：执行替换的数组或者字符串，也就是 haystack。如果 subject 是一个数组，替换操作将遍历整个 subject，返回值也将是一个数组。
- $count：如果被指定，该值将被设置为替换发生的次数。

（14）str_ireplace：str_replace( ) 的忽略大、小写版本。

语法格式如下。

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

该函数返回一个字符串或者数组。该字符串或数组是将subject中的全部search都替换为replace（忽略大、小写）之后的结果。如果没有一些特殊的替换规则，应该使用该函数替换带有i修正符的preg_replace( )函数。

如果search和replace为数组，那么str_replace( )将对subject作二者的映射替换。如果replace值的个数少于search的个数，多余的替换将使用空字符串来进行。如果search是一个数组而replace是一个字符串，那么search中每个元素的替换将始终使用这个字符串。如果search或replace是数组，它们的元素将从头到尾一个个处理。

参数说明如下。

- $search：要搜索的值，就像是 needle。可以使用 array提供多个 needle。
- $replace：替换的新字符。
- $subject：要被搜索和替换的字符串或数组，就像是haystack。如果subject是一个数组，替换操作将遍历整个subject，并且也将返回一个数组。
- $count：如果被设定，则会设置执行替换的次数。

（15）str_pad：使用另一个字符串填充字符串为指定长度。

语法格式如下。

```
    str_pad ( string $input , int $pad_length [, string $pad_string = " "
[, int $pad_type = STR_PAD_RIGHT ]]) : string
```

该函数返回input被从左端、右端或者同时两端填充到指定长度后的结果。如果可选的pad_string参数没有被指定，input将被空格字符填充，否则它将被pad_string填充到指定长度。

参数说明如下。

- $input：输入的字符串。
- $pad_length：如果pad_length的值是负数，小于或者等于输入字符串的长度，不会发生任何填充，并会返回input。
- $pad_string：如果填充字符的长度不能被pad_string整除，那么pad_string可能会被缩短。

- $pad_type：可选的pad_type参数的可能值为STR_PAD_RIGHT，STR_PAD_LEFT或STR_PAD_BOTH。如果没有指定pad_type，则默认它是STR_PAD_RIGHT。

（16）str_repeat：重复一个字符串。

语法格式如下。

```
str_repeat ( string $input , int $multiplier ) : string
```

返回input重复multiplier次后的结果。

参数说明如下。

- $input：待操作的字符串。
- $multiplier：input被重复的次数。multiplier必须大于等于0。如果multiplier被设置为0，函数返回空字符串。

（17）str_shuffle：随机打乱一个字符串。

语法格式如下。

```
str_shuffle ( string $str ) : string
```

str_shuffle( )函数打乱一个字符串，使用任何一种可能的排序方案。本函数并不会生成安全加密的值，不应用于加密。若需要安全加密的值，考虑使用openssl_random_pseudo_bytes( )。

参数说明如下。

- $str：输入字符串。

（18）str_split：将字符串转换为数组。

语法格式如下。

```
str_split ( string $string [, int $split_length = 1 ] ) : array
```

将一个字符串转换为数组。

参数说明如下。

- $string：输入的字符串。
- $split_length：每一段的长度。

（19）strcasecmp：二进制安全比较字符串（不区分大小写）。

语法格式如下。

```
strcasecmp ( string $str1 , string $str2 ) : int
```

二进制安全比较字符串（不区分大小写）。

参数说明如下。

- $str1 ：第一个字符串。
- $str2 ：第二个字符串。

（20）strlen：获取字符串长度。

语法格式如下。

```
strlen ( string $string ) : int
```

返回给定的字符串 string 的长度。

参数说明如下。

- $string：需要计算长度的字符串。

（21）strrev：反转字符串。

语法格式如下。

```
strrev ( string $string ) : string
```

返回string反转后的字符串。

参数说明如下。

- $string：待反转的原始字符串。

（22）strripos：计算指定字符串在目标字符串中最后一次出现的位置（不区分大、小写）。

语法格式如下。

```
strripos ( string $haystack , string $needle [, int $offset = 0 ] ) : int
```

以不区分大、小写的方式查找指定字符串在目标字符串中最后一次出现的位置。与strrpos()不同，strripos()不区分大、小写。

参数说明如下。

- $haystack：在此字符串中进行查找。
- $needle：needle可以是一个单字符或者多字符的字符串。
- $offset：可选参数。规定开始搜索的位置。

字符串位置从 0 开始，不是从 1 开始。该函数如果成功，则返回参数$needle在参数$haystack中所在位置，否则返回 false。

（23）strrpos：计算指定字符串在目标字符串中最后一次出现的位置。

语法格式如下。

```
strrpos ( string $haystack , string $needle [, int $offset = 0 ] ) : int
```

返回字符串haystack中needle最后一次出现的数字位置。注意，在PHP 4.0中，needle只能为单个字符。如果needle被指定为一个字符串，那么将仅使用第一个字符。

参数说明如下。

- $haystack：在此字符串中进行查找。
- $needle：如果needle不是一个字符串，它将被转换为整型并被视为字符的顺序值。
- $offset：可选参数。规定开始搜索的位置。

字符串位置从 0 开始，不是从 1 开始。该函数如果成功，则返回参数$needle在参数$haystack中所在位置，否则返回 false。

（24）strstr：查找字符串的首次出现。

语法格式如下。

```
strstr ( string $haystack , mixed $needle [, bool $before_needle =
FALSE ] ) : string
```

返回haystack字符串，从needle第一次出现的位置开始且到haystack结尾的字符串。该函数区分大、小写。如果想要不区分大、小写，请使用stristr( )。如果仅想确定needle是否

存在于haystack中，则使用速度更快、耗费内存更少的strpos()函数。

参数说明如下。

- $haystack：输入的字符串。
- $needle：如果needle不是一个字符串，那么它将被转化为整型并且作为字符的序号来使用。
- $before_needle：若为true，strstr()将返回needle在haystack中的位置之前的部分。

（25）strtolower：将字符串转化为小写。

语法格式如下。

```
strtolower ( string $string ) : string
```

将string中所有的字母字符转换为小写并返回。“字母”与当前所在区域有关。例如，在默认的“C”区域，字符 umlaut-A（ä）不会被转换。

参数说明如下。

- $string：输入的字符串。

（26）strtoupper：将字符串转化为大写。

语法格式如下。

```
strtoupper ( string $string ) : string
```

将string中所有的字母字符转换为大写并返回。“字母”与当前所在区域有关。例如，在默认的“C”区域，字符 umlaut-a（ä）不会被转换。

参数说明如下。

- $string：输入的字符串。

（27）substr_count：计算字符串出现的次数。

语法格式如下。

```
substr_count ( string $haystack , string $needle [, int $offset = 0
[, int $length ]] ) : int
```

substr_count( )返回子字符串needle在字符串haystack中出现的次数。注意，needle区分大、小写。该函数不会计算重叠字符串。

参数说明如下。

- $haystack：在此字符串中进行搜索。
- $needle：要搜索的字符串。
- $offset：开始计数的偏移位置。如果是负数，则从字符的末尾开始统计。
- $length：指定偏移位置之后的最大搜索长度。如果偏移量加上这个长度的和大于haystack的总长度，则打印警告信息。负数的长度length是从haystack的末尾开始统计的。

（28）substr：返回字符串的子字符串。

语法格式如下。

```
substr ( string $string , int $start [, int $length ] ) : string
```

返回字符串string由start和length参数指定的子字符串。

参数说明如下。

- $string：输入的字符串。必须至少有一个字符。
- $start：如果start是非负数，返回的字符串将从string的start位置开始。字符串位置是从0开始计算的。例如，在字符串“abcdef”中，在位置0的字符是“a”，在位置2的字符是“c”等。如果start是负数，返回的字符串将从string结尾处向前数第-start个字符开始。如果string的长度小于start，将返回false。

（29）ucwords：将字符串中每个单词的首字母转换为大写。

语法格式如下。

```
ucwords ( string $string [, string $delimiter = " \t\r\n\f\v" ] ) : string
```

将string中每个单词的首字符（如果首字符是字母）转换为大写字母，并返回这个字符串。这里单词的定义是紧跟在delimiter参数（默认：空格符、制表符、换行符、回车符、水平线以及竖线）之后的子字符串。

参数说明如下。

- $string：输入的字符串。
- $delimiter：可选参数，包含单词分隔字符。

（30）date_default_timezone_set：设定用于一个脚本中所有日期时间函数的默认时区。

语法格式如下。

```
date_default_timezone_set ( string $timezone_identifier ) : bool
```

参数说明如下。

- $timezone_identifier：表示国家区域。

（31）date_default_timezone_get：取得一个脚本中所有日期时间函数所使用的默认时区。

语法格式如下。

```
date_default_timezone_get () : string
```

（32）date：格式化一个本地时间/日期。

语法格式如下。

```
date ( string $format [, int $timestamp ] ) : string
```

参数说明如下。

- $format：必需参数。规定时间戳的格式，如表2-15所示。
- $timestamp：可选参数。规定时间戳，默认是当前的日期和时间。

表 2-15　format参数可填值

format字符	说　明	返回值示例
日	—	—
d	月份中的第几天，有前导零的2位数字	01到31
D	星期中的第几天，文本表示，3个字母	Mon到Sun
j	月份中的第几天，没有前导零	1到31

续表

format字符	说　明	返回值示例
l（"L"的小写字母）	星期几，完整的文本格式	Sunday到Saturday
N	ISO-8601格式数字表示的星期中的第几天（PHP 5.1.0新加）	1（表示星期一）到7（表示星期天）
S	每月天数后面的英文后缀，2个字符	st，nd，rd或者th。可以和j一起用
w	星期中的第几天，数字表示	0（表示星期天）到6（表示星期六）
z	年份中的第几天	0到365
星期	—	—
W	ISO-8601格式年份中的第几周，每周从星期一开始（PHP 4.1.0新加）	例如，42（当年的第42周）
月	—	—
F	月份，完整的文本格式，例如January或者March	January到December
m	数字表示的月份，有前导零	01到12
M	3个字母缩写表示的月份	Jan到Dec
n	数字表示的月份，没有前导零	1到12
t	指定的月份有几天	28到31
年	—	—
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，如果ISO的星期数（W）属于前一年或下一年，则用那一年（PHP 5.1.0新加）	Examples:1999or2003
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
时间	—	—
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到999
g	小时，12小时格式，没有前导零	1到12
G	小时，24小时格式，没有前导零	0到23
h	小时，12小时格式，有前导零	01到12

续表

format字符	说　明	返回值示例
H	小时，24小时格式，有前导零	00到23
i	有前导零的分钟数	00到59>
s	秒数，有前导零	00到59>
u	毫秒（PHP 5.2.2新加）。需要注意的是，date()函数总是返回000000，因为它只接受integer参数，而DateTime::format()才支持毫秒	示例：654321
时区	—	—
e	时区标识（PHP 5.1.0新加）	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	与世界时（GMT）相差的小时数	例如，+0200
P	与世界时（GMT）的差别，小时和分钟之间由冒号分隔（PHP 5.1.3新加）	例如，+02:00
T	本机所在的时区	例如，EST，MDT（在Windows下为完整文本格式，如“Eastern Standard Time”，中文版会显示“中国标准时间”）
Z	时差偏移量的秒数。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到43200
完整的日期/时间	—	—
c	ISO-8601 格式的日期（PHP 5.0.0新加）	2004-02-12T15:19:21+00:00
r	RFC 822格式的日期	例如，Thu，21 Dec 2000 16:01:07 +0200
U	从UNIX纪元（January 1 1970 00:00:00 GMT）开始至今的秒数	参见time()

（33）microtime：返回当前 UNIX 时间戳和微秒数。

语法格式如下。

```
microtime ([ bool $get_as_float ] ) : mixed
```

参数说明如下。

- $get_as_float：可选参数。当设置为true时，规定函数应该返回浮点数，否则返回字符串。默认为 false。

microtime( )为当前UNIX时间戳及微秒数。本函数仅在支持 gettimeofday( )系统调用的操作系统下可用。如果调用时不带可选参数，本函数以“msec sec”的格式返回一个字符串。其中，sec是自 UNIX纪元（世界时，1970 年 1 月 1 日，00:00:00）起到现在的秒数，msec 是

微秒部分。字符串的两部分都是以“秒”为单位返回的。如果给出 get_as_float 参数并且其值等价于 true，microtime( ) 将返回一个浮点数。

（34）time：返回当前的 UNIX 时间戳。

语法格式如下。

```
time () : int
```

返回自从 UNIX 纪元（世界时，1970 年 1 月 1 日，00:00:00）到当前时间的秒数。

（35）file_exists：检查文件或目录是否存在。

语法格式如下。

```
file_exists ( string $filename ) : bool
```

检查文件或目录是否存在。

参数说明如下。

- $filename：文件或目录的路径。在 Windows 中要用 //computername/share/filename 或者 \\computername\share\filename 来检查网络中的共享文件。

（36）mkdir：新建目录。

语法格式如下。

```
mkdir ( string $pathname [, int $mode = 0777 [, bool $recursive =
false [, resource $context ]]] ) : bool
```

尝试新建一个由 pathname 指定的目录。

参数说明如下。

- $pathname：目录的路径。
- $mode：mode 默认是 0777，意味着最大可能的访问权。mode 在 Windows 下被忽略。

注意　如果想用八进制数指定模式（也就是说，该数应以零开头），模式也会被当前的 umask 修改，可以用 umask() 来改变。

- $recursive：允许递归创建由 pathname 所指定的多级嵌套目录。
- $context：在 PHP 5.0.0 中增加了对上下文（Context）的支持。

（37）rmdir：删除目录。

语法格式如下。

```
rmdir ( string $dirname [, resource $context ] ) : bool
```

尝试删除 dirname 所指定的目录。 该目录必须是空的，而且要有相应的权限。 删除失败时会产生一个 E_WARNING 级别的错误。

参数说明如下。

- $dirname：目录的路径。
- $context：在 PHP 5.0.0 中增加了对上下文（context）的支持。

（38）rename：重命名一个文件或目录。

语法格式如下。

```
rename ( string $oldname , string $newname [, resource $context ] ) : bool
```

尝试将 oldname 重命名为 newname。

参数说明如下。

- $oldname：用于 oldname 中的封装协议必须和用于 newname 中的相匹配。
- $newname：新的名字。
- $context：在 PHP 5.0.0 中增加了对上下文（context）的支持。

（39）fopen：打开文件或者 URL。

语法格式如下。

```
fopen ( string $filename , string $mode [, bool $use_include_path =
false [, resource $context ]] ) : resource
```

参数说明如下。

- $filename：文件路径。
- $mode：操作权限，如表2-16所示。

表 2-16　操作权限

mode	说　明
'r'	只读方式打开，将文件指针指向文件头
'r+'	读写方式打开，将文件指针指向文件头
'w'	写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在，则尝试创建之
'w+'	读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在，则尝试创建之
'a'	写入方式打开，将文件指针指向文件末尾。如果文件不存在，则尝试创建之
'a+'	读写方式打开，将文件指针指向文件末尾。如果文件不存在，则尝试创建之
'x'	创建并以写入方式打开，将文件指针指向文件头。如果文件已存在，则fopen()调用失败并返回false，并生成一条E_WARNING级别的错误信息。如果文件不存在，则尝试创建之。这和为底层的open(2)系统调用指定O_EXCL|O_CREAT标记是等价的
'x+'	创建并以读写方式打开，其他的行为与'x'一样

fopen( ) 将 filename 指定的名字资源绑定到一个流上。

（40）file_get_contents ： 将整个文件读入一个字符串。

语法格式如下。

```
file_get_contents ( string $filename [, bool $use_include_path = fals
e [, resource $context [, int $offset = -1 [, int $maxlen ]]]] ) : string
```

和 file( ) 一样，只除了 file_get_contents( ) 把文件读入一个字符串。在参数 offset 所指定

的位置开始读取长度为 maxlen 的内容。如果失败，file_get_contents( ) 将返回 false。file_get_contents( ) 函数是用来将文件的内容读入到一个字符串中的首选方法。如果操作系统支持，还会使用内存映射技术来增强性能。

参数说明如下。

- $filename：要读取的文件的名称。

（41）file_put_contents ： 将一个字符串写入文件。

语法格式如下。

```
file_put_contents ( string $filename , mixed $data [, int $flags =
0 [, resource $context ]] ) : int
```

与依次调用 fopen( )、fwrite( ) 及 fclose( ) 功能相同。

参数说明如下。

- $filename：要被写入数据的文件名。
- $data：要写入的数据。类型可以是 string、array 或者是 stream 资源。如果将 data 指定为 stream 资源，stream 中所保存的缓存数据将被写入到指定文件中，这种用法类似于使用stream_copy_to_stream( ) 函数。参数 data 可以是数组（但不能为多维数组），这就相当于 `file_put_contents($filename, join('', $array))`。
- $flags：flags 的值可以是表2-17中 flag 使用 OR (|) 运算符进行的组合。
- $context：一个 context 资源。

表 2-17　flag设置值

flag值	描　　述
FILE_USE_INCLUDE_PATH	在 include目录里搜索 filename
FILE_APPEND	如果文件 filename已经存在，追加数据而不是覆盖
LOCK_EX	在写入时获得一个独占锁

（42）fclose：关闭一个已打开的文件指针。

语法格式如下。

```
fclose ( resource $handle ) : bool
```

将 handle 指向的文件关闭。

参数说明如下。

- $handle：文件指针必须有效，并且是通过 fopen( ) 或 fsockopen( ) 成功打开的。

以上是PHP内置的字符串函数、时间函数和文件函数。

2. 自定义函数

语法格式如下。

```
function function_name ([$arg_1],[$arg_2], ... ,[$arg_n]){
```

```
    fun_body;
[return arg_n;]
}
```

1）函数的调用

```
<?php
      /*  声明自定义函数  */
      function example($num){
    return "$num * $num = "$num * $num;
       }
       echo example(10);
?>
```

2）自定义函数的参数

参数传递方式如图2-2所示。

图 2-2　参数传递方式

按值传递如下。

```
<?php
function example( $m ){
       $m = $m * 5 + 10;
       echo "在函数内：\$m = ".$m;
}
$m = 1;
example( $m ) ;
echo "<p>在函数外 \$m = $m <p>" ;
?>
```

按引用传递如下。

```
<?php
function example( &$m ){
       $m = $m * 5 + 10;
       echo "在函数内：\$m = ".$m;
}
$m = 1;
example( $m ) ;
echo "<p>在函数外：\$m = $m <p>" ;
?>
```

默认参数如下。

```
<?php
function values($price,$tax=""){
       $price=$price+($price*$tax);
       echo "价格:$price<br>";
}
values(100,0.25);
values(100);
?>
```

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，函数将可能出错。

自定义函数通常将返回值传递给调用者的方式是使用return语句。

```
<?php
function values($price,$tax=0.65){
       $price=$price+($price*$tax);
       return $price;
}
echo values(100);
?>
```

变量的作用域如表2-18所示。

表 2-18　变量的作用域

作 用 域	说　明
全局变量	被定义在所有函数以外的变量，其作用域是整个PHP文件，但是在用户自定义函数内部是不可用的。要想在用户自定义函数内部使用全局变量，就要使用global关键词声明，或者通过使用全局数组$globals进行访问
局部变量	在函数的内部定义的变量，这些变量只限于在函数内部使用，在函数外部不能被使用
静态变量	能够在函数调用结束后仍保留变量值，当再次回到其作用域时，又可以继续使用原来的值。一般变量在函数调用结束后，其存储的数据值将被清除，所占的内存空间将被释放。使用静态变量时，先要用关键字static声明变量，需要把关键字static放在要定义的变量之前

3. 变量函数

可以将不同的函数名称赋给同一个变量，赋给变量哪个函数名，在程序中使用变量名并在后面加上圆括号时就执行哪个函数。

大多数函数都可以将函数名赋值给变量，形成变量函数。但变量函数不能用于语言结构，例如echo()、print()、unset()、isset()、empty()、include()、require()，以及类似的语句。

2.2.10 PHP文件的引用

PHP中文件引用的语句包括include语句、require语句。

1. include 语句

语法格式如下。

```
void include(string filename);
```

参数说明如下。

- filename：文件路径。

included.php文件代码如下。

```
<?php
$bookname = "PHP开发实战宝典";
echo "这是被引用的文件";
?>
```

index.php文件代码如下。

```
<?php
include("included.php");
echo "<br />".$bookname;
?>
```

2. require 语句

require 语句的使用方法与 include 语句类似，都是实现对外部文件的引用。语法格式如下。

```
void require(string filename);
```

3. include 语句和 require 语句的比较

include语句和require语句的比较如下。

（1）在使用require语句调用文件时，如果调用的文件没找到，require语句会输出错误信息，并且立即终止脚本的处理；include语句在没有找到文件时则会输出警告，不会终止脚本的处理。

（2）使用require语句调用文件时，只要程序一执行，就会立刻调用外部文件；使用include语句调用外部文件时，只有程序执行到该语句时才会调用外部文件。

4. include_once 语句和 require_once 语句

include_once语句的语法格式如下。

```
void include_once (string filename);
```

require_once语句的语法格式如下。

```
void require_once (string filename);
```

2.2.11 数组

通常在变量中保存的是单个数据，而在数组中保存的是多个变量的集合。使用数组的目的就是将多个相互关联的数据组织在一起形成一个整体，作为一个单元使用。变量和数组如图2-3所示。

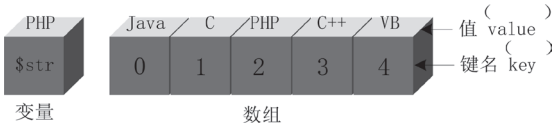

图 2-3　变量和数组

在PHP中将数组分为一维数组、二维数组和多维数组。无论是一维还是多维，都可以统一将数组分为两种，即数字索引数组（indexed array）和关联数组（associative array）。

- 数字索引数组：下标（键名）由数字组成，默认从0开始。

```
$arr_int = array ("PHP入门与实战","C#入门与实战","VB入门与实战");
```

- 关联数组：关联数组的键名可以是数字和字符串混合的形式。

```
$arr_string = array ("PHP"=>"PHP入门与实战","Java"=>"Java入门与实战",
"C#"=>"C#入门与实战");
```

1. 数组创建

通过数组标识符 [] 创建数组。在PHP中一种比较灵活的数组声明方式是通过数组标识符 [] 直接为数组元素赋值。语法格式如下。

```
$arr[key] = value;
$arr[] = value;
```

使用array()函数创建数组。语法格式如下。

```
array array ( [mixed ...])
```

参数mixed的格式为“key => value”，多个参数之间用英文逗号分开。

（1）数组中的索引（key）可以是字符串或数字。

（2）数组中的各数据元素的数据类型可以不同，也可以是数组类型。

2. 数组遍历

（1）使用foreach循环遍历数组。

（2）通过数组函数list( )和each( )遍历数组：list( )函数将数组中的值赋给一些变量。each( )函数返回数组中当前指针位置的键名和对应的值，并向前移动数组指针。

3. 数组输出

print_r()函数，语法格式如下。

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

var_dump()函数，语法格式如下。

```
void var_dump( mixed expression [,mixed expression [,…]])
```

4. 数组函数

（1）array_change_key_case：将数组中的所有键名修改为全大写或小写。

语法格式如下。

```
array_change_key_case ( array $array [, int $case = CASE_LOWER ] ) : array
```

array_change_key_case( ) 将 array 数组中的所有键名改为全小写或大写。本函数不改变数字索引。

参数说明如下。

- $array：需要操作的数组。
- $case：可以在这里使用两个常量，即CASE_UPPER 或 CASE_LOWER（默认值）。

```
<?php
$array = array("Fir" => 1, "Sec" => 4);
print_r(array_change_key_case($array, CASE_UPPER));
?>
```

输出：

```
Array
(
    [FIR] => 1
    [SEC] => 4
)
```

（2）array_chunk：将一个数组分割成多个。

语法格式如下。

```
array_chunk ( array $array , int $size [, bool $preserve_keys = false ] ) : array
```

将一个数组分割成多个数组，其中每个数组的单元数目由 size 决定。最后一个数组的单元数目可能会少于 size 个。

参数说明如下。

- $array：需要操作的数组。
- $size：每个数组的单元数目。
- $preserve_keys：设置为 true，可以使 PHP 保留输入数组中原来的键名。如果指定为 false，那么每个结果数组将用从零开始的新数字索引。默认值是 false。

```
<?php
$array = array('aa', 'bb', 'cc', 'dd', 'ee');
print_r(array_chunk($array, 2));
?>
```

输出：

```
Array
(
```

```
    [0] => Array
        (
            [0] => aa
            [1] => bb
        )
    [1] => Array
        (
            [0] => cc
            [1] => dd
        )
    [2] => Array
        (
            [0] => ee
        )
)
```

（3）array_column：返回数组中指定的一列。

语法格式如下。

```
array_column ( array $input , mixed $column_key [, mixed $index_
key = null ] ) : array
```

array_column( ) 返回input数组中键值为column_key的列。如果指定了可选参数index_key，那么input数组中这一列的值将作为返回数组中对应值的键。

参数说明如下。

- $input：需要取出数组列的多维数组。如果提供的是包含一组对象的数组，则只有public属性会被直接取出。为了也能取出 private 和 protected 属性，类必须实现 __get( ) 和 __isset( ) 魔术方法。
- $column_key：需要返回值的列，它可以是索引数组的列索引，或者是关联数组的列的键，也可以是属性名。可以是NULL，此时将返回整个数组（配合index_key参数重置数组键时非常管用）。
- $index_key：作为返回数组的索引/键的列，可以是该列的整数索引，或者字符串键值。

```
<?php
$rec = array(
    array(
        'id' => 1,
        'first_name' => 'John',
        'last_name' => 'Doe',
    ),
```

```
        array(
            'id' => 2,
            'first_name' => 'Sally',
            'last_name' => 'Smith',
        ),
        array(
            'id' => 3,
            'first_name' => 'Jane',
            'last_name' => 'Jones',
        ),
        array(
            'id' => 4,
            'first_name' => 'Peter',
            'last_name' => 'Doe',
        )

);
$names = array_column($rec, 'first_name');
print_r($names);
?>
```

输出：

```
Array
(
    [0] => John
    [1] => Sally
    [2] => Jane
    [3] => Peter
)
```

（4）array_combine：创建一个数组，用一个数组的值作为其键名，另一个数组的值作为其值。

语法格式如下。

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

返回一个 array，用来自 keys 数组的值作为键名，来自 values 数组的值作为相应的值。

参数说明如下。

- $keys：将被作为新数组的键。非法的值会被转换为字符串类型（string）。
- $values：将被作为 array 的值。

```
<?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
)
```

（5）array_count_values：统计数组中所有的值。

语法格式如下。

```
array_count_values ( array $array ) : array
```

array_count_values( ) 返回一个数组。数组的键是 array 里单元的值；数组的值是 array 单元的值出现的次数。

参数说明如下。

- $array：统计这个数组的值。

```
<?php
$array = array(1, "hello", 1, "world", "hello");
print_r(array_count_values($array));
?>
```

输出：

```
Array
(
   [1] => 2
   [hello] => 2
   [world] => 1
)
```

（6）array_diff_assoc ：带索引检查计算数组的差集。

语法格式如下。

```
array_diff_assoc ( array $array1 , array $array2 [, array $... ] ) : array
```

array_diff_assoc( ) 返回一个数组，该数组包括所有在 array1 中但是不在任何其他参数数组中的值。注意，与 array_diff( ) 不同的是，array_diff_assoc( ) 的键名也用于比较。

参数说明如下。

- $array1：从这个数组进行比较。

- $array2：被比较的数组。
- $...：更多被比较的数组。

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "green", "yellow", "red");
$result = array_diff_assoc($array1, $array2);
print_r($result);
?>
```

输出：

```
Array
(
    [b] => brown
    [c] => blue
    [0] => red
)
```

（7）array_diff_key：使用键名比较计算数组的差集。

语法格式如下。

```
array_diff_key ( array $array1 , array $array2 [, array $... ] ) : array
```

用 array1 中的键名与 array2 进行比较，返回不同键名的项。本函数和 array_diff( ) 相同，只除了比较是根据键名而不是值来进行的。

参数说明如下。

- $array1：从这个数组进行比较。
- $array2：针对此数组进行比较。
- $...：更多被比较的数组。

```
<?php
$array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3, 'purple' => 4);
$array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7, 'cyan' => 8);
var_dump(array_diff_key($array1, $array2));
?>
```

输出：

```
array(2) {
  ["red"]=>
  int(2)
  ["purple"]=>
  int(4)
}
```

（8）array_diff：计算数组的差集。

语法格式如下。

```
array_diff ( array $array1 , array $array2 [, array $... ] ) : array
```

对比 array1 和其他一个或者多个数组，返回在 array1 中但是不在其他array 中的值。

参数说明如下。

- $array1：要被对比的数组。
- $array2：和这个数组进行比较。
- $...：更多被比较的数组。

```
<?php
$array1 = array("a" => "green", "red", "blue", "red");
$array2 = array("b" => "green", "yellow", "red");
$result = array_diff($array1, $array2);
print_r($result);
?>
```

与在第一个参数 $array1 中多次出现的值一样处理，输出结果为：

```
Array
(
    [1] => blue
)
```

（9）array_fill_keys：使用指定的键和值填充数组。

语法格式如下。

```
array_fill_keys ( array $keys , mixed $value ) : array
```

使用 value 参数的值作为值，使用 keys 数组的值作为键来填充一个数组。

参数说明如下。

- $keys：使用该数组的值作为键。非法值将被转换为字符串。
- $value：填充使用的值。

```
<?php
$keys = array('foo', 5, 10, 'bar');
$a = array_fill_keys($keys, 'banana');
print_r($a);
?>
```

输出：

```
Array
(
    [foo] => banana
    [5] => banana
```

```
    [10] => banana
    [bar] => banana
)
```

（10）array_fill：用给定的值填充数组。

语法格式如下。

```
array_fill ( int $start_index , int $num , mixed $value ) : array
```

array_fill( ) 使用 value 参数的值将一个数组填充 num 个条目，键名由第一个参数$start_index 指定并作为返回数组的开始索引值。

参数说明如下。

- $start_index：返回的数组的第一个索引值。如果 start_index 是负数，那么返回的数组的第一个索引将会是 start_index ，而后面索引则从0开始。
- $num：插入元素的数量，必须大于或等于0。
- $value：用来填充的值。

```
<?php
$a = array_fill(5, 6, apple);
$b = array_fill(-2, 4, 'pear');
print_r($a);
print_r($b);
?>
```

输出：

```
Array
(
    [5]  => apple
    [6]  => apple
    [7]  => apple
    [8]  => apple
    [9]  => apple
    [10] => apple
)
Array
(
    [-2] => pear
    [0] => pear
    [1] => pear
    [2] => pear
)
```

（11）array_flip：交换数组中的键和值。

语法格式如下。

```
array_flip ( array $array ) : array
```

array_flip( ) 返回一个反转后的 array。例如， array 中的键名变成了值，而 array 中的值成了键名。

注意 array 中的值需要能够作为合法的键名（例如，需要是 integer 或者 string）。如果类型不对，将出现一个警告，并且有问题的键/值对将不会出现在结果里。如果同一个值出现多次，则最后一个键名将作为它的值，其他键会被丢弃。

参数说明如下。

- $array：要交换键/值对的数组。

```
<?php
$input = array("oranges", "apples", "pears");
$flipped = array_flip($input);print_r($flipped);
?>
```

输出：

```
Array
(
    [oranges] => 0
    [apples] => 1
    [pears] => 2
)
```

（12）array_key_exists ：检查数组里是否有指定的键名或索引。

语法格式如下。

```
array_key_exists ( mixed $key , array $array ) : bool
```

数组里有键 key 时，array_key_exists( ) 返回 true。 key 可以是任何能作为数组索引的值。

参数说明如下。

- $key：要检查的键。
- $array：一个数组，包含待检查的键。

```
<?php
$search_array = array('first' => 1, 'second' => 4);
if (array_key_exists('first', $search_array)) {
    echo "The 'first' element is in the array";
}
?>
```

（13）array_keys：返回数组中部分的或所有的键名。

语法格式如下。

```
array_keys ( array $array [, mixed $search_value = null [, bool $strict = false ]] ) : array
```

array_keys( ) 返回 input 数组中的数字或者字符串的键名。如果指定了可选参数 search_value，则只返回该值的键名；否则，input 数组中的所有键名都会被返回。

参数说明如下。

- $array：一个数组，包含要返回的键。
- $search_value：如果指定了这个参数，则只有包含这些值的键才会返回。
- $strict：判断在搜索的时候是否该使用严格的比较（===）。

```
<?php
$array = array(0 => 100, "color" => "red");
print_r(array_keys($array));
?>
```

输出：

```
Array
(
    [0] => 0
    [1] => color
)
```

（14）array_values：返回数组中所有的值。

语法格式如下。

```
array_values ( array $array ) : array
```

array_values( ) 返回 input 数组中所有的值并为其建立数字索引。

参数说明如下。

- $array：数组。

```
<?php
$array = array("size" => "XL", "color" => "gold");
print_r(array_values($array));
?>
```

输出：

```
Array
(
    [0] => XL
    [1] => gold
)
```

（15）array_merge：合并一个或多个数组。

语法格式如下。

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

array_merge( ) 将一个或多个数组的单元合并起来，一个数组中的值附加在前一个数组的后面，返回作为结果的数组。如果输入的数组中有相同的字符串键名，则该键名后面的值将覆盖前一个值；如果数组中包含数字键名，则后面的值将不会覆盖原来的值，而是附加到后面；如果只给了一个数组并且该数组是数字索引的，则键名会以连续方式重新索引。

参数说明如下。

- $array1：要合并的第一个数组。
- $...：要合并的数组列表。

```
<?php
$array1 = array("color" => "red", 2, 4);
$array2 = array ("a", "b", "color" => "green", "shape" => "trapezoid", 4);
$result = array_merge($array1, $array2);print_r($result);
?>
```

输出：

```
Array
(
    [color] => green
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
    [shape] => trapezoid
    [4] => 4
)
```

（16）array_multisort：对多个数组或多维数组进行排序。

语法格式如下。

```
array_multisort ( array &$array1 [, mixed $array1_sort_order = SORT_
ASC [, mixed $array1_sort_flags = SORT_REGULAR [, mixed $... ]]] ) : bool
```

array_multisort( ) 可以用来一次对多个数组进行排序，或者根据某一维或多维对多维数组进行排序。关联（string）键名保持不变，但数字键名会被重新索引。

参数说明如下。

- $array1：要排序的 array。
- $array1_sort_order：之前 array 参数要排列的顺序。SORT_ASC 按照上升顺序排序，SORT_DESC 按照下降顺序排序。此参数可以和 array1_sort_flags 互换，也可

以完全删除，默认是 SORT_ASC 。

- $array1_sort_flags：为 array 参数设定选项。排序类型标志如表2-19所示。

表 2-19　排序类型标志

标　志	描　述
SORT_REGULAR	将项目按照通常方法比较（不修改类型）
SORT_NUMERIC	按照数字大小比较
SORT_STRING	按照字符串比较
SORT_LOCALE_STRING	根据当前的本地化设置，按照字符串比较。它会使用locale信息，可以通过 setlocale() 修改此信息
SORT_NATURAL	字符串采用“自然排序”，类似 natsort()
SORT_FLAG_CASE	可以组合（按位或OR）SORT_STRING 或者 SORT_NATURAL大、小写不敏感的方式排序字符串 参数可以和array1_sort_order交换或者省略，默认情况下是SORT_REGULAR
...	可选的选项，可提供更多数组，跟随在 sort order 和 sort flag之后。提供的数组和之前的数组要有相同数量的元素。换言之，排序是按字典顺序排列的

```
<?php
$ar1 = array(10, 100, 100, 0);
$ar2 = array(1, 3, 2, 4);
array_multisort($ar1, $ar2);

var_dump($ar1);
var_dump($ar2);
?>
```

输出：

```
array(4) {
  [0]=> int(0)
  [1]=> int(10)
  [2]=> int(100)
  [3]=> int(100)
}
array(4) {
  [0]=> int(4)
  [1]=> int(1)
  [2]=> int(2)
  [3]=> int(3)
}
```

（17）array_pad：以指定长度将一个值填充进数组。

语法格式如下。

```
array_pad ( array $array , int $size , mixed $value ) : array
```

array_pad( )返回 array 的一个副本，并用 value 将其填补到 size 指定的长度。如果 size 为正，则填补到数组的右侧；如果 size 为负，则从左侧开始填补。如果 size 的绝对值小于或等于 array 数组的长度，则没有任何填补。有可能一次最多填补 1 048 576 个单元。

参数说明如下。

- $array：需要被填充的原始数组。
- $size：新数组的长度。
- $value：将被填充的值。只有在 array 的现有长度小于 size 的长度时才有效。

```
<?php
$input = array(12, 10, 9);
$result = array_pad($input, 5, 0);
$result = array_pad($input, -7, -1);
$result = array_pad($input, 2, "noop");
?>
```

（18）array_pop：弹出数组最后一个单元（出栈）。

语法格式如下。

```
array_pop ( array $array ) : mixed
```

array_pop( ) 弹出并返回 array 数组的最后一个单元，并将数组 array 的长度减一。

使用此函数后会重置（reset( )）array 指针。

参数说明如下。

- $array：需要弹出栈的数组。

```
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_pop($stack);
print_r($stack);
?>
```

输出：

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
)
```

（19）array_product：计算数组中所有值的乘积。

语法格式如下。

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

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

参数说明如下。

- $array：一个数组。

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

输出：

```
product(a) = 384
product(array()) = 1
```

（20）array_push：将一个或多个单元压入数组的末尾（入栈）。

语法格式如下。

```
array_push ( array $array , mixed $value1 [, mixed $... ] ) : int
```

array_push( ) 将 array 当成一个栈，并将传入的变量压入 array 的末尾。array 的长度将根据入栈变量的数目增加。具体运行原理如下。

```
<?php
$array[] = $var;
?>
```

对每个传入的值重复以上动作。

如果用 array_push( ) 为数组增加一个单元，还不如用 $array[] = ，因为这样没有调用函数的额外负担。

如果第一个参数不是数组，array_push( ) 将发出一条警告。这和 $var[] 的行为不同，后者会新建一个数组。

参数说明如下。

- $array：输入的数组。
- $value1：要压入 array 末尾的第一个值。

```
<?php
$stack = array("orange", "banana");
```

```
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

输出：

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
    [3] => raspberry
)
```

（21）array_rand：从数组中随机地取出一个或多个单元。

语法格式如下。

```
array_rand ( array $array [, int $num = 1 ] ) : mixed
```

从数组中随机地取出一个或多个单元，并返回随机条目的一个或多个键。它使用伪随机数产生算法，所以不适合密码学场景。

参数说明如下。

- $array：输入的数组。
- $num：指明想取出的单元数。

```
<?php
$input = array("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$rand_keys = array_rand($input, 2);
echo $input[$rand_keys[0]] . "\n";
echo $input[$rand_keys[1]] . "\n";
?>
```

（22）array_replace：使用传递的数组替换第一个数组的元素。

语法格式如下。

```
array_replace ( array $array1 [, array $... ] ) : array
```

array_replace( )函数是使用后面数组的值替换第一个数组的值，如果一个键存在于第一个数组，同时也存在于第二个数组，它的值将被第二个数组中的值替换。如果一个键存在于第二个数组，但是不存在于第一个数组，则会在第一个数组中创建这个元素。如果一个键仅存在于第一个数组，它将保持不变。如果传递了多个替换数组，它们将被按顺序依次处理，后面的数组将覆盖之前的值。array_replace( ) 是非递归的，它将第一个数组的值进行替换而不管第二个数组中是什么类型。

参数说明如下。

- $array1：替换该数组的值。

- $...：包含要提取元素的数组。 后面的数组里的值会覆盖前面的值。

```
<?php
$base = array("orange", "banana", "apple", "raspberry");
$replacements = array(0 => "pineapple", 4 => "cherry");
$replacements2 = array(0 =>  "grape");
$basket = array_replace($base, $replacements, $replacements2);
print_r($basket);
?>
```

输出：

```
Array
(
    [0] => grape
    [1] => banana
    [2] => apple
    [3] => raspberry
    [4] => cherry
)
```

（23）array_search ：在数组中搜索给定的值，如果成功则返回首个相应的键名。

语法格式如下。

```
array_search ( mixed $needle , array $haystack [, bool $strict = false ] ) : mixed
```

参数说明如下。

- $needle：搜索的值。

如果 needle 是字符串，则比较以区分大、小写的方式进行。

- $haystack：传递一个数组。
- $strict：如果可选的第三个参数 strict 为 true，则 array_search( ) 将在 haystack 中检查完全相同的元素。 这意味着同样严格比较 haystack 里 needle 的 类型，并且对象需是同一个实例。

```
<?php
$array = array(0 => 'blue', 1 => 'red', 2 => 'green', 3 => 'red');
$key = array_search('green', $array); // $key = 2;
$key = array_search('red', $array);   // $key = 1;
?>
```

（24）array_shift：将数组开头的单元移出数组。

语法格式如下。

```
array_shift ( array $array ) : mixed
```

array_shift( ) 将 array 的第一个单元移出并作为结果返回，将 array 的长度减1并将所有其他单元向前移动一位。所有的数字键名将改为从零开始计数，文字键名将不变。

使用此函数后会重置（reset()）array 指针。

参数说明如下。

- $array：输入的数组。

```
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_shift($stack);
print_r($stack);
?>
```

输出：

```
Array
(
    [0] => banana
    [1] => apple
    [2] => raspberry
)
```

（25）array_slice：从数组中取出一段。

语法格式如下。

```
array_slice ( array $array , int $offset [, int $length = NULL [, bool $preserve_keys = false ]] ) : array
```

array_slice( ) 返回根据 offset 和 length 参数所指定的 array 数组中的一段序列。

参数说明如下。

- $array：输入的数组。
- $offset：如果参数 $offset 值为非负，则序列将从参数 $array 的值（偏移量）开始。如果参数 $offset 值为负，则序列将从参数 $array 中距离末端数值为-offset的地方开始。
- $length：如果给出 length 并且为正，则序列中将具有这么多的单元。如果给出 length 并且为负，则序列将终止在距离数组末端这么远的地方。如果省略，则序列将从 offset 开始一直到 array 的末端。
- $preserve_keys：array_slice( ) 默认会重新排序并重置数组的数字索引，可以通过将 preserve_keys 设置为 true 来改变此行为。

```
<?php
$input = array("a", "b", "c", "d", "e");
$output = array_slice($input, 2);        //返回"c" "d" "e"
$output = array_slice($input, -2, 1);    //返回"d"
$output = array_slice($input, 0, 3);     //返回"a" "b" "c"
print_r(array_slice($input, 2, -1));
print_r(array_slice($input, 2, -1, true));
?>
```

输出：

```
Array
(
    [0] => c
    [1] => d
)
Array
(
    [2] => c
    [3] => d
)
```

（26）array_splice：去掉数组中的某一部分并用其他值取代。

语法格式如下。

```
array_splice ( array &$input , int $offset [, int $length = count($input) [, mixed $replacement = array() ]] ) : array
```

将 input 数组中由 offset 和 length 指定的单元去掉，如果提供了 replacement 参数，则用其中的单元取代。

参数说明如下。

- $input：输入的数组。
- $offset：如果 offset 为正，则从 input 数组中该值指定的偏移量开始移除；如果 offset 为负，则从 input 末尾倒数该值绝对值指定的偏移量开始移除。
- $length：如果省略 length，则移除数组中从 offset 到结尾的所有部分。如果指定 length 并且为正值，则移除这么多单元。如果指定 length 并且为负值，则移除从 offset 到数组末尾倒数-length 为止中间所有的单元。如果设置 length 为零，不会移除单元。

当给出了 replacement，要移除从 offset 到数组末尾的所有单元时，用 count($input) 作为 length。

- $replacement：如果给出了 replacement 数组，则被移除的单元被此数组中的单元替代。如果 offset 和 length 的组合结果是不会移除任何值，则 replacement 数组中的单元将被插入到 offset 指定的位置，注意替换数组中的键名不保留。如果用来替换 replacement 的只有一个单元，那么不需要给它加上 array()，除非该单元本身就是一个数组、一个对象或者 NULL。

```
<?php
$input = array("red", "green", "blue", "yellow");
array_splice($input, 2);

$input = array("red", "green", "blue", "yellow");
array_splice($input, 1, -1);

$input = array("red", "green", "blue", "yellow");
array_splice($input, 1, count($input), "orange");

$input = array("red", "green", "blue", "yellow");
array_splice($input, -1, 1, array("black", "maroon"));

$input = array("red", "green", "blue", "yellow");
array_splice($input, 3, 0, "purple");
?>
```

（27）array_sum：对数组中所有值求和。

语法格式如下。

```
array_sum ( array $array ) : number
```

array_sum() 将数组中的所有值相加，并返回结果。

参数说明如下。

- $array：输入的数组。

```
<?php
$a = array(2, 4, 6, 8);
echo "sum(a) = " . array_sum($a) . "\n";
$b = array("a" => 1.2, "b" => 2.3, "c" => 3.4);
echo "sum(b) = " . array_sum($b) . "\n";
?>
```

输出：

```
sum(a) = 20
sum(b) = 6.9
```

（28）array_unique：移除数组中重复的值。

语法格式如下。

```
array_unique ( array $array [, int $sort_flags = SORT_STRING ] ) : array
```

array_unique( ) 接受 array 作为输入并返回没有重复值的新数组。注意键名保持不变。array_unique( ) 先将值作为字符串排序，然后对每个值只保留第一个遇到的键名，然后忽略后面所有的键名。这并不意味着在未排序的 array 中同一个值第一个出现的键名会被保留。

注意　当且仅当 (string) $elem1 === (string) $elem2 时，两个单元被认为相同。例如，字符串表达一样时会使用首个元素。

参数说明如下。

- $array：输入的数组。
- $sort_flags：第二个可选参数sort_flags 可用于修改排序方法。排序类型标记如表2-20所示。

表 2-20　排序类型标记

排序类型标记	描　　述
SORT_REGULAR	按照通常方法比较（不修改类型）
SORT_NUMERIC	按照数字形式比较
SORT_STRING	按照字符串形式比较
SORT_LOCALE_STRING	根据当前的本地化设置，按照字符串比较

```
<?php
$arr = array("a" => "green", "red", "b" => "green", "blue", "red");
$result = array_unique($arr);
print_r($result);
?>
```

输出：

```
Array
(
    [a] => green
    [0] => red
    [1] => blue
)
```

（29）arsort：对数组进行逆向排序并保持索引关系。

语法格式如下。

```
arsort ( array $array [, int $sort_flags = SORT_REGULAR ] ) : bool
```

本函数对关联数组按照键值进行逆向排序，主要用于对那些单元顺序很重要的关联数组进行排序。如果成功返回 true，否则返回false。

参数说明如下。

- $array：输入的数组。
- $sort_flags：可以用可选的参数 sort_flags 改变排序的行为。

```
<?php
  $fruit = array("d" => "lemon", "a" => "orange", "b" => "banana","c"
=> "apple");
  asort($fruit);
  foreach ($fruit as $key => $val) {
    echo "$key = $val".'<br/>';
  }
?>
```

输出：

```
a = orange
d = lemon
b = banana
c = apple
```

（30）asort：对数组进行排序并保持索引关系。

语法格式如下。

```
asort ( array &$array [, int $sort_flags = SORT_REGULAR ] ) : bool
```

本函数对关联数组以默认的设定方式按照键值进行排序，主要用于对那些单元顺序很重要的关联数组进行排序。如果成功返回 true，否则返回false。

参数说明如下。

- $array：输入的数组。
- $sort_flags：可以用可选的参数 sort_flags 改变排序的行为。

```
<?php
  $fruit= array("d" => "lemon", "a" => "orange", "b" => "banana", "c"=
> "apple");
  asort($fruit);
  foreach ($fruit as $key => $val) {
     echo "$key = $val".'<br/>';
  }
?>
```

输出：

```
c = apple
b = banana
d = lemon
a = orange
```

（31）compact：建立一个数组，包括变量名和它们的值。

语法格式如下。

```
compact ( mixed $var1 [, mixed $var2... ] ) : array
```

参数说明如下。

- var1：必需。可以是带有变量名的字符串，或者是一个变量数组。
- var2，...：可选。可以是带有变量名的字符串，或者是一个变量数组。允许多个参数。

```
<?php
  $city  = "ChongQing";
  $state = "CA";
  $event = "SIGGRAPH";
  $location_vars = array("city", "state");
  $result = compact("event", "nothing_here", $location_vars);
  print_r($result);
?>
```

输出：

```
Array
(
    [event] => SIGGRAPH
    [city] => ChongQing
    [state] => CA
)
```

（32）count：计算数组中的单元数目，或对象中的属性个数。

语法格式如下。

```
count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] ) : int
```

参数说明如下。

- $array：必需参数。规定要计数的数组。
- $mode：可选参数。规定函数的模式。默认为0，不计算多维数组中的所有元素；1为递归地计算数组中元素的数目（计算多维数组中的所有元素）。

```
<?php
    $arr = array(1,2,3,4,5);
    echo count($arr);
?>
```

输出：

```
    5
```

（33）in_array：检查数组中是否存在某个值。

语法格式如下。

```
in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] ) : bool
```

参数说明如下。

- $needle：待搜索的值。

如果 needle 是字符串，则比较是区分大、小写的。

- $haystack：待搜索的数组。
- $strict：如果第三个参数 strict 的值为 true ，则 in_array( ) 函数还会检查 needle 的类型是否和 haystack 中的相同。

```
<?php
$sys = array("Mac", "NT", "Irix", "Linux");
if (in_array("Irix", $sys)) {
    echo "Got Irix";}
if (in_array("mac", $sys)) {
    echo "Got mac";
}
?>
```

第二个条件false，因为 in_array( ) 是区分大、小写的，显示为：

```
Got Irix
```

（34）krsort：对数组按照键名逆向排序。

语法格式如下。

```
krsort ( array $array [, int $sort_flags = SORT_REGULAR ] ) : bool
```

对数组按照键名逆向排序，保留键名到数据的关联。

参数说明如下。

- $array：输入的数组。
- $sort_flags：可选参数，可以用 sort_flags 改变排序的行为。

```
<?php
   $fruit = array("d"=>"lemon", "a"=>"orange", "b"=>"banana","c"=
>"apple");
   krsort($fruit);
   foreach ($fruit as $key => $val) {
      echo "$key = $val".'<br/>';
   }
?>
```

输出：

```
d = lemon
```

```
c = apple
b = banana
a = orange
```

（35）ksort：对数组按照键名排序。

语法格式如下。

```
ksort ( array $array [, int $sort_flags = SORT_REGULAR ] ) : bool
```

对数组按照键名排序，保留键名到数据的关联。本函数主要用于关联数组。

参数说明如下。

- $array：输入的数组。
- $sort_flags：可选参数，可以用 sort_flags 改变排序的行为。

```
<?php
  $fruit = array("d"=>"lemon", "a"=>"orange", "b"=>"banana","c"=
>"apple");
  ksort($fruit);
  foreach ($fruit as $key => $val) {
      echo "$key = $val".'<br/>';
  }
?>
```

输出：

```
a = orange
b = banana
c = apple
d = lemon
```

（36）range：根据范围创建数组，包含指定的元素。

语法格式如下。

```
range ( mixed $start , mixed $end [, number $step = 1 ] ) : array
```

参数说明如下。

- $start：序列的第一个值。
- $end：序列结束于 end 的值。
- $step：如果设置了步长 step，会被作为单元之间的步进值，step 应该为正值。不设置step，则默认为 1。

```
<?php
  // array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,13)
  foreach (range(0, 13) as $num) {
      echo $num;
  }
?>
```

（37）rsort：对数组逆向排序。

语法格式如下。

```
rsort ( array $array [, int $sort_flags = SORT_REGULAR ] ) : bool
```

参数说明如下。

- $array：输入的数组。
- $sort_flags：可选参数，可以用 sort_flags 改变排序的行为。

```
<?php
  $fruit = array("orange", "banana", "apple");
  rsort($fruit);
  foreach ($fruit as $key => $val) {
      echo "$key = $val".'<br/>';
  }
?>
```

输出：

```
0 = orange
1 = banana
2 = apple
```

（38）shuffle：打乱数组。

语法格式如下。

```
shuffle ( array $array ) : bool
```

示例如下。

```
<?php
  $num = range(1, 20);
  shuffle($num);
  foreach ($num as $number) {
      echo "$number ";
  }
?>
```

由于输出结果每次不同，查看结果可复制代码执行。

（39）sort：对数组排序。

语法格式如下。

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

参数说明如下。

- $array：要排序的数组。
- $sort_flags：可选参数，sort_flags 可以用以下值改变排序的方式。排序类型标记如表2-21所示。

表 2-21　排序类型标记

排序类型	描　　述
SORT_REGULAR	正常比较单元（不改变类型）
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（locale）设置把单元当作字符串比较，可以用setlocale()改变
SORT_NATURAL	和natsort()类似，对每个单元以“自然的顺序”对字符串进行排序。PHP 5.4.0 中的新增功能
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
  $fruit = array("lemon", "orange", "banana", "apple");
  sort($fruit);
  foreach ($fruit as $key => $val) {
      echo "fruits[" . $key . "] = " . $val . '<br/>';
  }
?>
```

输出：

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

（40）usort：使用用户自定义的比较函数对数组中的值进行排序。

语法格式如下。

```
usort ( array $array , callable $value_compare_func ) : bool
```

参数说明如下。

- $array：一个数组。
- $value_compare_func：可选参数。调用一个自定义比较函数的函数名，此参数值为一个字符串。

如果此函数的第一个参数小于、等于或大于第二个参数时，该比较函数必须返回一个小于、等于或大于0的整数。

```
<?php
  function cmp($a, $b){
    if ($a == $b) {
```

```
            return 0;
        }
        return ($a < $b) ? -1 : 1;
    }
    $a = array(3, 2, 5, 6, 1);
    usort($a, "cmp");
    foreach ($a as $key => $value) {
        echo "$key: $value\n";
    }
    ?>
```

输出：

```
0: 1
1: 2
2: 3
3: 5
4: 6
```

2.3 任务实现

2.3.1 创建相册

利用Dreamweaver创建项目2，在目录project2下新建一个文件index.php，保留该文件中的代码，利用PHP代码可以嵌入到HTML页面中的特性，具体文件内容如下。

```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
        <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
        <title>无标题文档</title>
        </head>
        <body>
        </body>
    </html>
```

也可以将已经事先做好的静态界面复制到此文件内，替换index.php中原有的文件内容。最终此页面的代码如下。

```
<!DOCTYPE html>
<html>
    <head>
      <meta name="viewport" content="width=device-width,initial-
scale=1.0,maximum-scale=1.0,user-scalable=no"  charset="utf-8"
/>
       <title><?=$title?>--在线相册</title>
       <link rel="stylesheet" type="text/css" href="../css/
bootstrap.min.css" />
       <link rel="stylesheet" type="text/css" href="../css/main.
css" />
    </head>
    <body>
       <div class="body">
          <nav class="navbar navbar-expand-md bg-dark navbar-dark">
             <i class="fa fa-chrome fa-spin fa-lg fa-inverse"></i>
  
             <a class="navbar-brand" href="#">在线相册</a>
          </nav>
          <div class="container">
             <div class="row">
                <div class="col-lg-2 col-md-2" >
                   <p><a href="#" class="btn btn-info">相册管理</a></p>
                   <p><a  href="javascript:void(0)"  class="btn  btn-
outline-info">图片管理</a></p>
                </div>
                <div class="col-lg-10 col-md-10 col-sm-10 list" >
                   <div class="row">
                      <h3 class="col-lg-6 col-md-6 col-sm-6">相册管理</h3>
                      <div class="input-group" >
                         <form  method="post" action="init.php" >
                            <input  type="hidden"  name="action"  value=
"new" />
                            <input  type="text"  name="new_name"
 placeholder="输入相册名称" required  />
                            <input type="submit" value="创建相册" class=
"sub_add"  name="sub_add" />
                         </form>
```

```
                        <form method="post" enctype="multipart/form-
data">
                            <input type="hidden"  name="action"
value="upload" />
                            <input type="file" name="upload" required
/>
                            <input type="submit" value="上传图片" />
                        </form>
                    </div>
                </div>
                <div class="container">
                    <div class="top-nav"><a href="index.php">首页</a>
                        <i></i> <a href="#">相册名称</a>
                    </div>
                </div>
                <div class="album">
                    <!-- 相册列表 -->
                    <div class="album-list">
                        <div class="list-content">
                            <a href="#"><img src="./covers/nopic.
jpg"></a>
                            <div class="list-desc"><p><a href="#">相
册名称</a></p></div>
                        </div>
                </div>
                <div class="album-list">
                    <div class="list-content">
                            <a href="#"><img src="./covers/nopic.
jpg"></a>
                              <div class="list-desc"><p><a href="#">
相册名称</a></p></div>
                        </div>
                  </div>
                  <div class="album-list">
                     <div class="list-content">
                            <a href="#"><img src="./covers/nopic.
jpg"></a>
```

```
                        <div class="list-desc"><p><a href="#">相册名称
</a></p></div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </body>
</html>
```

注意

以上代码不一一赘述。如果有看不懂的读者，可以先回顾一下HTML综合布局知识。通过<form>标签创建一个Web表单，并通过method属性指定表单的提交方式为post，通过action属性指定表单提交的后台接收页面为init.php，action属性也可以不填写，如果不填写，则表单提交到本页。在表单中，通过表单空间来接收用户填写的相册名称数据，并通过控件的name属性设置表单提交的字段。

若要查看index.php的运行效果，则必须将project2的整个项目通过项目1介绍的环境来运行，具体操作步骤如下。

步骤01　打开并运行phpStudy集成环境，单击“其他选项菜单”按钮，在弹出的菜单中选择“站点域名管理”命令，在弹出的对话框中填入网站域名、网站目录、网站端口，第二域名可选填，如图2-4所示。

步骤02　单击“新增”按钮，单击“保存设置并生成配置文件”按钮，等待apache重新启动完成。

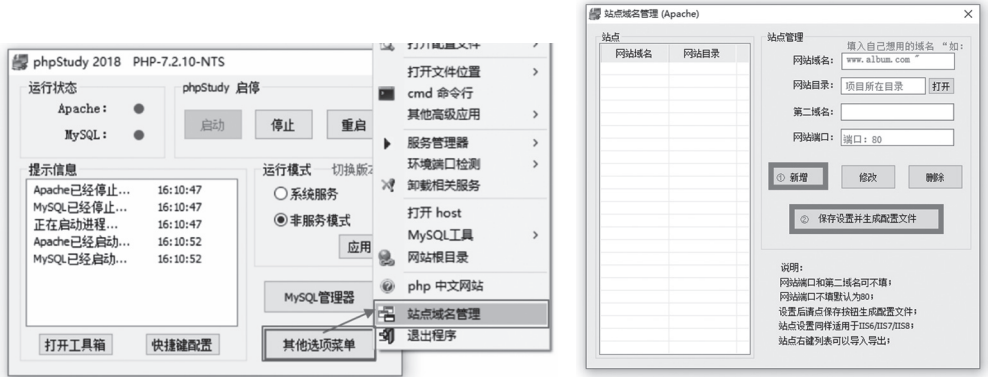

图 2-4　配置本地站点域名

步骤03　单击phpStudy主界面中的“其他选项菜单”按钮，在弹出的菜单中选择“打开host”命令，在host文件末尾追加一行“127.0.0.1　　www.album.com”，保存，如图2-5所示。

图 2-5 打开并修改host文件

步骤04 打开任意浏览器，输入域名“www.album.com\index.php”即可预览项目页。利用PHP获取表单值，具体实现代码如下。

```
<?php
    $new_name  =  $_POST['new_name'];
?>
```

在项目中采用了自定义函数来实现，具体函数代码如下。

```
<?php
    /**
    * 接收输入的函数
    * @param array $method 输入的数组（可用字符串get和post来表示）
    * @param string $name 从数组中取出的变量名
    * @param string $type 表示类型的字符串
    * @param mixed $default 变量不存在时使用的默认值
    * @return mixed 返回的结果
    */
    function input($method, $name, $type = 's', $default = '')
    {
        switch ($method) {
            case 'get': $method = $_GET;
                break;
            case 'post': $method = $_POST;
                break;
        }
        $data = isset($method[$name]) ? $method[$name] : $default;
        switch ($type) {
            case 's': return is_string($data) ? $data : $default;
            case 'd': return (int) $data;
            case 'a': return is_array($data) ? $data : [];
```

```
            default: trigger_error('不存在的过滤类型"' . $type . '"');
        }
}
?>
```

利用上面的函数可以获取表单（文本框、按钮等表单元素）的值。要使PHP能够获取表单提交的数据，还要在之前的表单上加上代码 action="init.php" ，这样才能规定index.php表单数据提交到init.php后端获取其值。在项目2的目录project2下新建一个文件init.php，具体代码如下。

```
<?php
    //提交按钮
    if(!empty($_POST)){
      $action = input('post', 'action', 's');
      //获取文本框的值
      $new_name = input('post', 'new_name', 's');
      echo $action.'<br>';
      echo $new_name;
}
//没有表单提交时继续执行
?>
```

开发项目时只要是需要获取表单数据，都是运用类似的方式获取的。代码中的empty（）函数用于判断数据是否提交。如果$_POST数组中有数据，则使用input自定义函数获取表单提交的数据，并用echo输出提交的内容。查看效果可以通过浏览器访问www.album.com\index.php并单击“创建相册”按钮。

2.3.2 相册显示

通过2.3.1任务页面可以看到相册以列表方式显示，在index.php的运行中可以看到相册列表效果。分析“相册列表”代码如下。

```
<div class="album">
    <!-- 相册列表 -->
    <div class="album-list">
        <div class="list-content">
          <a href="#"><img src="./covers/nopic.jpg"></a>
          <div class="list-desc"><p><a href="#">相册名称</a></p></div>
        </div>
  </div>
  <div class="album-list">
      <div class="list-content">
```

```
        <a href="#"><img src="./covers/nopic.jpg"></a>
        <div class="list-desc"><p><a href="#">相册名称</a></p></div>
      </div>
  </div>
  <div class="album-list">
     <div class="list-content">
        <a href="#"><img src="./covers/nopic.jpg"></a>
        <div class="list-desc"><p><a href="#">相册名称</a></p></div>
     </div>
  </div>
</div>
```

从上面代码分析可知，此列表是可以通过循环代码得到的，`$list['album']`是后端代码构造的数据源，其存放的值为一个数组。有了这样的构造，可以更好地实现其功能。

$list数据源：

```
Array
(
    [album] => Array
        (
            [0] => Array
                (
                    [id] => 53
                    [name] => 美食
                    [cover] =>
                    [num] => 0
                )
            [1] => Array
                (
                    [id] => 52
                    [name] => 校园
                    [cover] =>
                    [num] => 0
                )
            [2] => Array
                (
                    [id] => 51
                    [name] => 房产
                    [cover] =>
```

```
                [num] => 0
            )
        [3] => Array
            (
                [id] => 50
                [name] => 汽车
                [cover] =>
                [num] => 0
            )
        [4] => Array
            (
                [id] => 49
                [name] => 游戏
                [cover] =>
                [num] => 0
            )
)
```

循环输出代码：

```
<?php foreach ($list['album'] as $v){ ?>
  <div class="album-list">
  <div class="list-content">
    <a href="?id=<?=$v['id']?>"><img src="./covers/<?=$v['cover']
?: 'nopic.jpg'?>"></a>
    <div class="list-desc"><p><a href="?id=<?=$v['id']?>"><?=ht
mlspecialchars($v['name'])?></a> (<?=$v['num']?>)</p></div>
  </div>
  </div>
<?php } ?>
```

代码中的foreach用于遍历数组，“?:”是三元运算符，htmlspecialchars( ) 函数是把预定义的字符转换为 HTML 实体。

2.3.3 照片上传

在Web开发过程中经常会涉及图片的上传，例如头像设置、信息封面设置、产品图片展示等。本任务在相册管理中也是非常核心的功能，通过本任务来讲解PHP对上传文件的接收与处理等相关知识。具体要求如下。

（1）在HTML页面中创建一个上传图片的表单。

（2）判断上传图片文件的类型。
（3）判断文件是否上传成功。
（4）图片上传成功并显示。

1. 编写照片上传表单

在项目中编写页面文件index.php，用于实现表单文件的上传，关键代码如下。

```
<form method="post" enctype="multipart/form-data">
   <input type="file" name="upload" required />
   <input type="submit" value="上传图片" />
</form>
```

在上述代码里，<form>表单省略action属性，表示提交到当前URL地址；method属性为表单提交方式，文件上传需设置为post，并需设置表单的MIME类型属性enctype值为multipart/form-data。

2. PHP 接收上传文件

```
<?php
    $upload_name = $_FILES['upload'];
    //利用<pre>标签使输出的内容含有空格和换行
    echo '<pre>';
    print_r($_FILES); //输出获取上传文件的信息
    echo '</pre>';
?>
```

上述代码是PHP接收表单提交过来的文件，全局变量$_FILES数组是获取文件信息，由于是数组，利用内置函数print_r()输出获取的上传文件信息，如下所示。

```
Array
(
    [upload] => Array
        (
            [name] => canvans.png
            [type] => image/png
            [tmp_name] => C:\Windows\phpD2F9.tmp
            [error] => 0
            [size] => 231
        )
)
```

可以看出，$_FILES数组中有一个upload元素，upload是表单中上传文件的<input>标签的name属性值。upload元素是一个数组，数组元素error值为0，表明文件上传成功，该过程中没有出现错误。name元素表示图片的名称，type元素表示上传文件的类型，size元素为上传文件的大小。每一个上传文件都有name、type、size、tmp_name等信

息。文件上传后，这些文件相关信息存在FILES这个数组变量。`$_FILES["upload"]["name"]`相当于一个多维数组的访问，FILES先获取表单名称为upload的input上传的文件数据，然后再访问name、type、size、error等数据。`$_FILES["upload"]["error"]`用来表示文件上传的情况。`$_FILES ["upload"]["error"]=0`，表示文件正常上传；`$_FILES["upload"]["error"]>0`，表示文件没能正常上传；`$_FILES["upload"]["error"]=1`表示上传文件超过服务器限定的值，例如超过服务器空间大小；`$_FILES["upload"]["error"]=2`表示超过浏览器限定上传的值；`$_FILES["upload"]["error"]=3`表示文件只有部分被上传；`$_FILES["upload"]["error"]=4`表示没有文件上传。`$_FILES["upload"]["error"]`还可以为5、6、7、8，这里不作深究，只需知道大于0时就意味着文件上传出错。

从本内容后，所有代码都放在项目中的inc\init.php、inc\album.php和inc\function.php文件内。

如果能获取到上传文件的一系列属性，就可以判断文件大小是否满足规定大小、文件是否上传成功、是否存在临时文件夹、是否写入成功等情况。

3. 上传失败获取错误信息

如果上传失败，可以通过`$_FILES['upload']['error']`获取错误信息，将错误信息显示给用户，这里在代码中利用switch语句来编写，代码位于inc/function.php文件内。具体代码如下。

```
<?php
    /**
    * 检查上传文件
    * @param array $file 上传文件的 $_FILES['xx'] 数组
    * @return string 检查通过返回true,否则返回错误信息
    */
    function upload_check($file)
    {
      $error = isset($file['error']) ? $file['error'] : UPLOAD_ERR_NO_FILE;
      switch ($error) {
         case UPLOAD_ERR_OK:
            return is_uploaded_file($file['tmp_name']) ?: '非法文件';
         case UPLOAD_ERR_INI_SIZE:
            return '文件大小超过了服务器设置的限制！';
         case UPLOAD_ERR_FORM_SIZE:
            return '文件大小超过了表单设置的限制！';
         case UPLOAD_ERR_PARTIAL:
```

```
                return '文件只有部分被上传！';
            case UPLOAD_ERR_NO_FILE:
                return '没有文件被上传！';
            case UPLOAD_ERR_NO_TMP_DIR:
                return '上传文件临时目录不存在！';
            case UPLOAD_ERR_CANT_WRITE:
                return '文件写入失败！';
            default:
                return '未知错误';
        }
    }
    ?>
```

4. 判断上传文件类型

在开发中一般都会判断用户上传的文件类型是否满足要求。本案例制作的是相册，上传文件的类型为图片，则应判断上传的文件是否是设定的图片类型。具体代码如下。

```
    <?php
        // 允许上传的图片扩展
        $arr_ext=['jpg', 'jpeg', 'png'];
        if (!in_array(strtolower($ext), $arr_ext)) {
              return errors('文件上传失败：只允许扩展名：'. implode(', ',
$arr_ext));
        }
    ?>
```

注意 代码中$arr_ext 保存允许的上传图片的类型。strtolower( )函数是将$ext传入的值转换为小写。in_array( )函数搜索$arr_ext数组中是否存在$ext。若文件类型满足，则可以保存文件操作。

5. 保存上传文件并生成缩略图

```
        // 创建原图保存目录
        $upload_dir = "./uploads/$new_dir";
        if(!is_dir($upload_dir) && !mkdir($upload_dir, 0777, true)) {
            return errors('文件上传失败：无法创建保存目录！');
        }
        // 创建缩略图保存目录
        $thumb_dir = "./thumbs/$new_dir";
        if(!is_dir($thumb_dir) && !mkdir($thumb_dir, 0777, true)) {
```

```
            return errors('文件上传失败：无法创建缩略图保存目录！');
        }
        //保存上传文件（将上传文件从临时目录移动到项目指定的目录下）
        if(!move_uploaded_file($file['tmp_name'], "$upload_dir/$new_name"))
        {
            return errors('文件上传失败：无法保存文件！');
        }
        //创建缩略图
    thumb($upload_dir.'/'.$new_name, $thumb_dir.'/'.$new_name,
$thumb_size);
```

代码中is_dir(file) 函数检查指定的文件是否是目录。如果文件名存在并且为目录，则返回 true。如果 file 是一个相对路径，则按照当前工作目录检查其相对路径。mkdir( ) 函数用于创建目录，若成功则返回true，否则返回false，其语法为 `mkdir(path,mode,recursive,context)`，具体参数详解见表2-22所示。thumb为自定义函数，详见项目的inc/function.php文件，此函数内用到的函数都是PHP提供的操作图片相关的内置函数，用法详见PHP官方提供的帮助手册。

```
    /**
     * 生成缩略图
     * @param string $file 原图的路径
     * @param string $save 缩略图的保存路径
     * @param int $limit 缩略图的边长（像素）
     * @return bool 成功返回true,失败返回false
     */
    function thumb($file, $save, $limit)
    {
        $func = [
            'image/png' => function ($file, $img = null) {
                return $img ?imagepng($img,$file): imagecreatefrompng
($file);
            },
            'image/jpeg' => function ($file, $img = null) {
                return $img ? imagejpeg($img, $file, 100) :
imagecreatefromjpeg($file);
            }
        ];
        /*通过getimagesize()获取图像信息,该函数参数接收图片文件路径,返回图像
信息数组,其中图像信息数组中前两个元素保存了图片的宽度和高度*/
```

```
        $info = getimagesize($file);
        /*通过list()接收索引数组,将元素依次赋值给变量$width和$height,则得到上传图片的宽度和高度*/
        list($width, $height) = $info;
        $mime = $info['mime'];
        if (!in_array($mime, ['image/png', 'image/jpeg'])) {
            trigger_error('创建缩略图失败，不支持的图片类型。', E_USER_WARNING);
            return false;
        }
        $img = $func[$mime]($file);
        //宽度大于高度
        if ($width > $height) {
            $size = $height;
            $x = (int) (($width - $height) / 2);
            $y = 0;
        } else {
            $size = $width;
            $x = 0;
            $y = (int) (($height - $width) / 2);
        }
        /*绘制缩略图的画布资源,imagecreatetruecolor()用于创建画布,参数为画布的宽度和高度*/
        $thumb = imagecreatetruecolor($limit, $limit);
        // imagecopyresampled()处理图片缩放
        imagecopyresampled($thumb, $img, 0, 0, $x, $y, $limit, $limit, $size, $size);
        return $func[$mime]($save, $thumb);
    }
```

mkdir参数表如表2-22所示。

表 2-22　mkdir参数表

参　数	描　述
path	必需。规定要创建的目录名称
mode	可选。规定权限。默认是0777（允许全局访问） mode参数由以下四个数字组成 （1）第一个数字通常是0 （2）第二个数字规定所有者的权限 （3）第三个数字规定所有者所属的用户组的权限 （4）第四个数字规定其他所有人的权限

续表

参　数	描　述
	可能的值（如需设置多个权限，请对下面的数字进行总计）如下 （1）1 = 执行权限 （2）2 = 写权限 （3）4 = 读权限
recursive	可选。规定是否设置递归模式（PHP 5.0.0中新增）
context	可选。规定文件句柄的环境。context是一套可以修改流的行为的选项（PHP 5.0.0中新增）

由于本项目数据被保存到数据库中，所有图片上传成功后记录到数据库，数据库相关知识作为本项目的扩展内容，在此项目暂不细讲。

6. 相册展示

用户创建好相册后显示到页面中，若没设置相册封面，则显示一张默认的图片。控制显示默认图片的代码使用三元运算符来完成。关键代码如下。

```
<img src="./covers/<?=$v['cover'] ?: 'nopic.jpg'?>">
```

注意　代码中<?=$v['cover'] ?: 'nopic.jpg'?>适用于在html代码中内嵌PHP代码。./covers/为图片所在路径。

将创建的相册存储到数据库，以列表呈现，前端代码如下。

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,initial-scale=1.0,maximum-scale=1.0,user-scalable=no" charset="utf-8" />
<title><?=$title?>--在线相册</title>
<link rel="stylesheet" type="text/css" href="../css/bootstrap.min.css" />
<link rel="stylesheet" type="text/css" href="../css/main.css" />
</head>

<body>
<div class="body">
    <nav class="navbar navbar-expand-md bg-dark navbar-dark">
        <i class="fa fa-chrome fa-spin fa-lg fa-inverse"></i>  
        <a class="navbar-brand" href="index.php">在线相册</a>
    </nav>
```

```
      <div class="container">
           <div class="row">
                <div class="col-lg-2 col-md-2" >
                     <p><a href="index.php" class="btn btn-info">相册
管理</a></p>
                     <p><a href="javascript:void(0)" class="btn btn-
outline-info">图片管理</a></p>
                </div>
                <div class="col-lg-10 col-md-10 col-sm-10 list" >
                     <div class="row">
                          <h3 class="col-lg-6 col-md-6 col-sm-6">相册
管理</h3>
                          <div class="input-group" >
                               <form method="post">
                                    <input type="hidden" name="action"
value="new" />
                                    <input type="text" name="new_name"
placeholder="输入相册名称" required  />
                                  <input type="submit" value="创建相册"
class="sub_add"  name="sub_add" />
                               </form>
                               <form method="post" enctype="multipart/
form-data">
                                   <input type="hidden" name="action"
value="upload" />
                                   <input type="file" name="upload"
required />
                                    <input type="submit" value="上传
图片" />
                               </form>
                          </div>
                     </div>
                     <div class="container">
                          <div class="top-nav"><a href="index.php">
首页</a>
                            <?php foreach($nav as $val){ ?>
                                 <i></i> <a href="index.php?id=<?php
echo $val['id']; ?>"><?php echo htmlspecialchars($val['name']); ?></a>
```

```
                        <?php } ?>
                    </div>
                </div>
                <div class="album">
                        <?php if(empty($list['album']) &&
empty($list['picture'])): ?>
                            <div class="album-tip">相册暂无内容</
div>
                        <?php endif; ?>
                        <!-- 相册列表 -->
                        <?php foreach ($list['album'] as $v):
?>
                          <div class="album-list">
                            <div class="list-content">
                              <a href="?id=<?=$v['id']?>"><img
src="./covers/<?=$v['cover'] ?: 'nopic.jpg'?>"></a>
                                <div class="list-desc"><p><a hr
ef="?id=<?=$v['id']?>"><?=htmlspecialchars($v['name'])?></a>
(<?=$v['num']?>)</p></div>
                                <div class="list-opt">
                                    <form method="post">
                                        <input type="hidden"
name="action_id" value="<?=$v['id']?>">
                                        <button name="action"
value="del" class="del" >删除</button>
                                    </form>
                                </div>
                            </div>
                          </div>
                        <?php endforeach; ?>
                </div>
            </div>
        </div>
    </div>
</body>

</html>
```

若在相册内上传了照片，可以把喜欢的照片设置为封面。设置封面前端页面的代码如下。

```
<form method="post">
            <input type="hidden" name="action_id"
value="<?=$v['id']?>">
            <?php if($id){ ?><button name="action" value="pic_
cover">设为封面</button><?php } ?>
            <button  name="action" value="pic_del">删除</button>
</form>
```

后端通过init.php接收，代码如下。

```
//提交按钮
 $action = input('post', 'action', 's');
//获取提交对应id
$t_id=input('post', 'action_id', 'd');
switch ($action){
   case 'new'://创建相册
      $name = input('post', 'new_name', 's');
      if(newalbum($id,$name)>1){
          //创建成功重定向
        header("location:index.php?id=".$id);
      };
      break;
   case 'upload'://上传图片
      $file=input($_FILES, 'upload', 'a');
      //print_r($_FILES);
      upload($id,$file);
      break;
   case 'del'://删除相册
      delete($t_id);
      break;
   case 'pic_cover'://设置为封面
      picture_cover($t_id, $id);
      break;
   case 'pic_del'://删除图片
      picture_delete($t_id);
      break;
 }
```

注意 代码中case 'pic_cover'表示当前提交过来的数据为设置的封面。picture_cover()函数在album.php中定义并执行相应操作。

7. 图片展示

用户完成相册的创建后，就可以在相册中上传图片并将图片按照相册方式显示到页面中。因为数据是保存到数据库中的，所以通过查询数据库中的相册和图片数据，以数组的方式保存到变量$list内。页面将数据显示出来，图片列表显示index.php页面文件的代码如下（核心代码见加粗部分）。

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,initial-scale=1.0,maximum-scale=1.0,user-scalable=no" charset="utf-8" />
<title><?=$title?>--在线相册</title>
<link rel="stylesheet" type="text/css" href="../css/bootstrap.min.css" />
<link rel="stylesheet" type="text/css" href="../css/main.css" />
</head>

<body>
<div class="body">
<nav class="navbar navbar-expand-md bg-dark navbar-dark">
    <i class="fa fa-chrome fa-spin fa-lg fa-inverse"></i>  
    <a class="navbar-brand" href="index.php">在线相册</a>
</nav>

<div class="container">
     <div class="row">
          <div class="col-lg-2 col-md-2" >
               <p><a href="index.php" class="btn btn-info">相册管理</a></p>
               <p><a href="javascript:void(0)" class="btn btn-outline-info">图片管理</a></p>
          </div>
          <div class="col-lg-10 col-md-10 col-sm-10 list" >
               <div class="row">
                    <h3 class="col-lg-6 col-md-6 col-sm-6">相册管理</h3>
```

```
                    <div class="input-group" >
                        <form method="post">
                            <input type="hidden" name="action"
value="new" />
                            <input type="text" name="new_name"
placeholder="输入相册名称" required  />
                            <input type="submit" value="创建相册"
class="sub_add"  name="sub_add" />
                        </form>
                        <form method="post" enctype="multipart/
form-data">
                            <input type="hidden" name="action"
value="upload" />
                            <input type="file" name="upload"
required />
                            <input type="submit" value="上传图片"
/>
                        </form>
                    </div>
                </div>
                <div class="container">
                    <div class="top-nav"><a href="index.php">首页</a>
                      <?php foreach($nav as $val){ ?>
                          <i></i> <a href="index.php?id=<?php echo
$val['id']; ?>"><?php echo htmlspecialchars($val['name']); ?></a>
                      <?php } ?>
                    </div>
                <div class="album">
                        <?php if(empty($list['album']) &&
empty($list['picture'])): ?>
                          <div class="album-tip">相册暂无内容</div>
                       <?php endif; ?>
                       <!-- 相册列表 -->
                       <?php foreach ($list['album'] as $v): ?>
                         <div class="album-list">
                           <div class="list-content">
                           <a href="?id=<?=$v['id']?>"><img src="./
covers/<?=$v['cover'] ?: 'nopic.jpg'?>"></a>
```

```
                        <div class="list-desc"><p><a href="?id=<?=
$v['id']?>"><?=htmlspecialchars($v['name'])?></a>  (<?=$v['num']?>)</
p></div>
                        <div class="list-opt">
                                <form method="post">
                               <input type="hidden" name="action_
id" value="<?=$v['id']?>">
                                <button  name="action"  value="del"
class="del" >删除</button>
                                </form>
                          </div>
                        </div>
                      </div>
                    <?php endforeach; ?>
                    <!-- 子相册列表 -->
                    <?php foreach ($list['picture'] as $v): ?>
                      <div class="album-list">
                        <div class="list-content">
                         <a href="show.php?id=<?=$v['id']?>"><img
src="./thumbs/<?=$v['path'] ?: './covers/nopic.jpg'?>"></a>
                         <div class="list-desc"><p><a href="show.
php?id=<?=$v['id']?>"><?=htmlspecialchars($v['pic_name'])?></a></
p></div>
                        <div class="list-opt">
                            <form method="post">
                               <input type="hidden" name="action_
id" value="<?=$v['id']?>">
                            <?php if($id): ?><button name="action"
value="pic_cover">设为封面</button><?php endif; ?>
                               <button  name="action" value="pic_
del">删除</button>
                            </form>
                        </div>
                      </div>
                    </div>
                  <?php endforeach; ?>
            </div>
      </div>
```

```
    </div>
</div>
</div>
</body>

</html>
```

其中，后端数据$list在init.php中获得，代码如下。

```
$list = album_list($id, $sort);
```

代码中的album_list()函数在album.php中定义并执行相应操作。