PHP 快速参考手册完整目录与试读

目 录

前 言

关于本书

我们知道，PHP 作为一个流行的开源项目，其功能和扩展性都是非常强大的，而我们一直在思考的问题是，学习 PHP 的第一步是什么？我想，初学者当然没有必要使用 C 语句或 PEAR 去扩展 PHP 的功能，而是能够踏踏实实地学习 PHP 的基础知识；实际上，能够写出 PHP 代码，并能够与相关的开发技术关联起来，然后，能够开发出自己的 Web 项目才是初学者应有的目标，也许，本书的目的就是这样。

当然，一本书不可能包括 PHP 的所有内容，哪怕只是应用方面的；本书中，我们会以使用者的角度学习和讨论 PHP 应用开发中的一些常用、实用的内容，而更多内容则需要在不断地学习和实践中逐渐累积的。

本书的特点在于，我们会从功能开发的角度来介绍技术的实践过程，突出实际应用，并且会有大量封装代码可以直接在项目中使用，读者可以根据学习或工作中的需求随时关注相关主题。

学习一门技术，官方资料都是很有重要的学习资源，我们可以在 http://php.net 找到 PHP 的官方资料。官网文档突出技术本身的完整性，当我们需要了解某一技术点的完整细节时，可以在官网找到相应的内容。读者可以结合本书的内容和官方资料完整地理解 PHP 中的各项技术细节和应用方法，这会是一个不错的学习方式。

此外，本书内容适合于 PHP 的较新版本，对于 PHP 官方已经不建议使用或不再支持的功

能（如魔术引号等内容），本书将不会过多介绍。如果大家在旧版本的 PHP 项目中发现不明白的代码，可以在 http://php.net 网站找到相应的说明。

本书内容

第一部分，从第 1 章到第 6 章，是 PHP 编程语言方面的学习，为我们下一步实现 Web 项目的各项功能打下基础。

学习软件开发的最好方法就是实践，所以，在第 1 章，我们介绍了如何在 Windows 系统中搭建 PHP 网站，为进一步学习和测试做好准备。

第 2 章介绍了 PHP 编程的基础知识，通过这些内容的学习，我们将可以写出简单的 PHP 代码，并了解如何在 PHP 中处理各种类型的数据、如何控制程序的流程，以及创建自定义函数等。

第 3 章和第 4 章介绍了 PHP 5 中的面向对象开发,讨论了如何创建类及类的实例 对象），并学习通过类的继承达到代码复用的目的。

第 5 章介绍了接口，以及如何通过类来实现接口。通过接口的应用，可以为我们创建复杂的代码结构提供支持，如实现设计模式。

第 6 章讨论了在 PHP 中处理异常的相关内容。

第二部分，从第 7 章到第 14 章，我们结合 Web 项目的特点，介绍了常用的功能实现与技巧。

第 7 章介绍了 Web 项目的工作特点,讨论了 Web 开发中的常用技术,如 HTML 表单、HTML5 中的新表单元素、如何使用 PHP 处理 HTML 表单数据，以及 AJAX 应用基础等。

第 8 章介绍了如何在服务器端使用 SimpleXML 处理 XML，以及如何在客户端使用 JavaScript 处理 XML 对象。

第 9 章讨论了如何使用 PHP 内置的 SQLite3 数据库。

第 10 章讨论了一个数据操作应用框架，我们将讨论如何使用统一的接口进行数据库操作，对数据操作基础代码进行封装，从而能够在项目中更专注于业务处理。

第 11 章讨论了 MySQL 数据库应用基础，并封装到我们的数据操作接口组件中。

第 12 章讨论了如何使用 PHP 操作文件,并演示了如何处理 Web 项目中用户上传的文件。

第 13 章介绍了使用 GD2 库进行图形处理，并讨论了如何在 Web 项目中实现验证码。

第 14 章，我们讨论了两个综合应用，包括表单处理的更多内容，以及使用 AJAX 实现一个页面聊天室。

第三部分，从第 15 章到第 17 章。

第 15 章，主要介绍了与 Windows 平台相关的内容，包括在 IIS 中支持 PHP，以及如何在 PHP 中操作 SQL Server 数据库。

第 16 章，我们以 Fedora 操作系统为例，简单介绍了与 Linux 平台相关的内容。

第 17 章，作为继续深入学习 PHP 及 Web 开发的简单讨论。包括最新的 PHP 7 特性介绍，以及一些在 Web 项目开发中应该深入学习的内容。

读者类型

本书主要面向 PHP 初学者和正在使用 PHP 的开发者，可以作为学习的教材，也可以作为一本随手的参考手册。

我的 E-mail 是 chydev@163.com，欢迎大家随时交流。

1. 准备工作

关于古今中外 PHP 的发展就不多说了，大家摆渡一下就会到达光明的彼岸。

本章，我们将介绍 Apache 和 PHP 应用的基本知识，并为学习和测试做好准备；不过，为了简化准备工作和测试环境，在介绍 PHP 相关内容时只考虑 Windows 环境下的使用；在本书偏后的章节中，我们将会讨论在 Fedora 系统中如何配置和测试 Apache 和 PHP 网站。

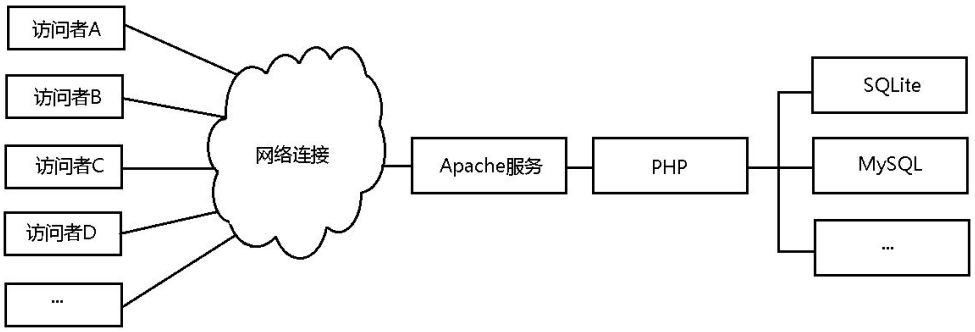
这一章的内容包括：

- 简介
- 配置 Apache
- 配置 PHP
- 安装 Apache 服务
- 创建 PHP 文件
- 注释
- print 和 echo 语句

1.1. 简介

Apache+PHP+MySQL 的组合，曾经是开源世界构建网站的黄金搭档，关于它们的历史，在这里就不多说了，相信大家都可以很方便的了解这些内容。

不过，为了更好的理解本书的内容，我们还是简单的了解一下 Apache、PHP 和 MySQL 在整个应用环境中的角色。



图中，我们可以看到，Apache、PHP 与数据库的角色有着明显的分工：

- Apache 是一个网站服务，负责对 Web 服务端口（如 HTTP 默认端口 80）的侦听，当一个访问者连接到 Apache 服务器时，Apache 会对访问者提交的请求做出响应，并返回相应的资源，如访问者需要访问哪个网页、哪个图片等等；如果用户访问的是 PHP 文件，那么 PHP 的代码就会执行，并返回执行结果。
- PHP 是一项动态页面技术，其功能是在 Web 服务器端通过编程对访问者提交的请求进行更多的处理，并返回相应的内容。如访问者登录时，PHP 就可以对其提交的用

户名、密码等信息进行验证。整个 PHP 环境包括了编程语言、内置和扩展开发资源，我们可以使用这些资源很方便地进行 Web 服务器端开发。

- SQLite 和 MySQL 分别是两种数据库。虽然说 SQLite 是 PHP 环境中集成的一种数据库，不过，对于项目的分层设计来讲，数据库的应用层面都是相同的，它们都是为项目提供外部的数据管理服务。

1.2. 配置 Apache

开始配置 Apache 和 PHP 网站之前，请大家注意以下几个路径，本书的大部分测试内容都位于这几个目录之中：

- d:\Apache24，Apache2.4 的安装目录。
- d:\php56，PHP 5.6 的安装目录。
- d:\mysql56，MySQL 5.6 安装目录。
- d:\phpweb\root，PHP 网站目录，也是本书源代码所在目录。

我们可以从 http://www.apache.org 网站获取 Apache 的最新版本，请注意，本书使用的是 Apache 2.4 中的 64 位版本，下载的是 zip 文件，直接解压到 D:盘根目录即可，此时，在 d:盘根目录下就是有一个 Apache24 目录了。需要注意的是，在 Windows 环境下，Apache 2.4 需要 VC++2010 运行库的支持。

Apache 的配置文件位于安装目录下的/conf/httpd.conf 文件。其中，对于初学者，或者是说本书的测试环境，我们需要注意和修改的参数项包括：

- Define SRVROOT "d:/apache24"，定义系统变量 SRVROOT 的值，即 Apache 服务根目录的路径。注意路径中使用的“/”符号，即使是在 Windows 系统下，也是没有问题的。
- ServerRoot "d:/apache24"，同样是指定 Apache 服务根目录的路径。默认的设置使用了 SRVROOT 常量，不过，为了更清晰地显示参数内容，我们使用了实际的路径，大家可以根据实际需要设置。
- Listen 8080，Apache 侦听的端口，在本书的示例中，我们使用了 8080 端口，而不是 HTTP 协议中默认的 80 端口。
- ServerName localhost:8080，设置 Apache 服务名称，localhost 指定为本地主机，并指定端口为 8080，这和 Listen 设置的端口号是一致的。在实际的网站中，需要将 localhost 设置为相应的服务器 IP 地址或域名。
- DocumentRoot "d:/phpweb/root"，设置网站的根目录。
- <Directory "d:/phpweb/root">，与网站根目录相关的参数。
- DirectoryIndex index.php，网站的默认页面，如果指定更多的默认主页，可以使用空格将它们隔开，如 DirectoryIndex index.php index.html。
- ScriptAlias /cgi-bin/ "d:/Apache24/cgi-bin/"，设置 CGI 相关资源的路径。
- <Directory "d:/Apache24/cgi-bin/">，设置与 CGI 相关的指令。

此外，默认情况下，通过 Apache 服务，用户是可以在浏览器中查看网站的文件和目录结构的，很多情况下，这显然是不需要的，我们可以通过修改 httpd.conf 配置文件禁止此功能，在配置文件中找到类似如下的代码。

```
<Directory "d:/phpweb/root">
    Options Indexes FollowSymLinks
</Directory>
```

将代码中的 Indexes 删除，即可禁止索引网站文件与目录结构的功能。

为了让 Apache 服务支持 PHP，我们还需要在 Apache 配置文件中设置以下参数，你可以手工添加到配置文件末尾。

```
LoadModule php5_module d:/php56/php5apache2_4.dll
AddType application/x-httpd-php .php
Action application/x-httpd-php "d:/php56/php.exe"
```

请注意 LoadModule 指令中，使用了与 Apache 2.4 版本相对应的 PHP 模块（php5apache2_4.dll 文件），如果你的测试环境中使用了不同版本的 Apache，请注意此处应该引用相对应的文件。

1.3. 配置 PHP

我们可以从 http://php.net/网站获取 PHP 的最新版本，本书使用的是 PHP 5.6.x 中的 64 位版本，下载的同样是 zip 文件，压缩时指定路径为“d:\php56”。在 Windows 环境下，PHP 5.6.x 需要 VC++ 2012 运行库的支持。

PHP 的配置文件是 php.ini，初始情况下这个文件是没有的，不过，我们可以从几个配置文件中复制一个并重新命名，默认情况下，在 PHP 的安装目录中会有以下两个配置文件：

- php.ini-development
- php.ini-production

现在，我们是学习和测试，所以，可以选择复制 php.ini-development 文件，重新命名为 php.ini，并将此文件复制到 Windows 的安装目录，一般是在“c:\windows”目录中，如果你拿不准 Windows 系统的安装目录，可以执行 cmd 命令打开命令行窗口，然后使用如下命令查看：

```
echo %windir%
```

在 php.ini 文件中，我们需要注意的参数包括：

- doc_root = "d:/phpweb/root"，设置网站的根目录，这与 Apache 中 DocumentRoot 参数的值是一样的。
- extension_dir = "d:/php56/ext"，指定 PHP 扩展库的路径，它一般都位于 PHP 安装路径中的 ext 目录。

- extension = php_mysqli.dll，我们可以看到 php.ini 文件中包含很多 extension 指令，它们都是 PHP 中的功能扩展模块，大多都使用分号（;）定义为了注释，在实际应用中，我们可以根据实际情况将所需要的扩展库前的分号删除，然后重启服务即可。如 php_gd2.dll、php_mysqli.dll、php_sqlite3.dll、php_mbstring.dll 等。
- date.timezone = "Asia/Shanghai"，设置时区，大陆地区可以设置为上海或重庆，完整的时区列表可参考：http://php.net/manual/zh/timezones.php。

1.4. 安装 Apache 服务

本部分将介绍如何在 Windows 环境下启动 Apache 服务，我们使用的是 64 位的 Windows 7 操作系统。

前面，Apache 和 PHP 的基本配置已经完成，下面所需要做的工作就是启动 Apache 服务，我们所使用是 Apache 安装路径下/bin 目录中的 httpd.exe 命令。

运行 cmd 打开 Windows 的命令行窗口，然后切换到 d:\Apache24\bin 目录，如果你忘记了 DOS 命令的使用，下面是你需要执行的两行命令：

```
C:\Users\当前 Windows 用户名> d: <回车>
D:\> cd apache24\bin <回车>
```

安装/删除 Apache 服务

以下命令可以将 Apache 安装为 Windows 系统服务：

```
httpd -k install -n"Apache24"
```

其中，-k 参数指定为操作类型，install 为安装服务，而 uninstall 为删除服务。-n 参数指定所操作的服务名称。完整的参数说明可以使用“httpd -?”命令查看。

启动/停止 Apache 服务

成功安装 Apache 服务后，我们可以通过两种方式启动服务，一是通过 Windows 中的 net 命令，如：

```
net start Apache24
```

第二种方法是使用 httpd.exe 命令，如：

```
httpd -k start -n"Apache24"
```

而停止 Apache 服务同样可以使用这两种方法，如：

```
net stop Apache24
```

或

```
httpd -k stop -n"Apache24"
```

1.5. 创建 PHP 文件

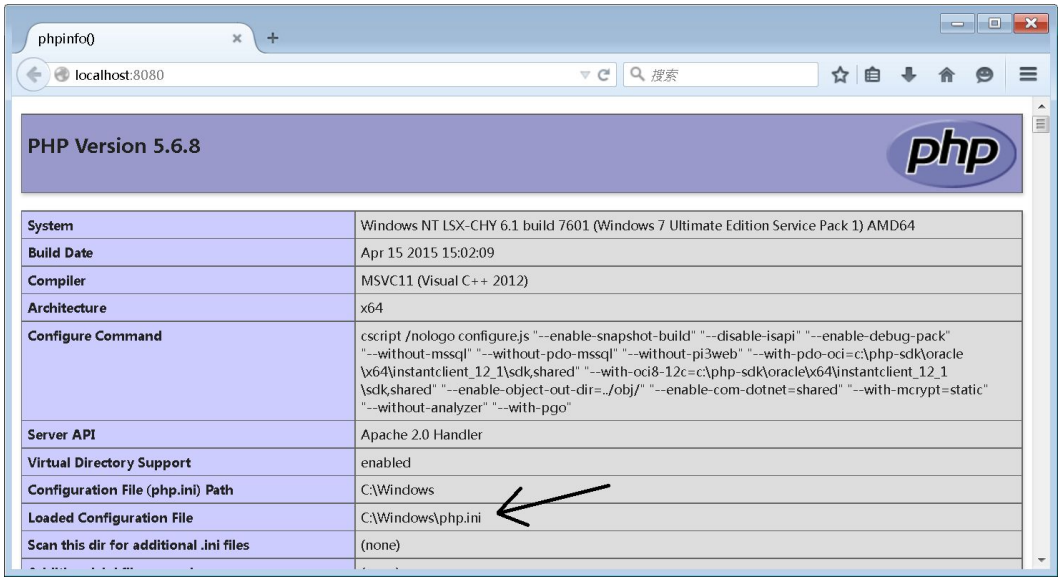
本书中，我们在 Windows 系统下使用的 PHP 代码编辑工具是 Notepad++，这是一个开源的文本编辑器，而且是免费使用的，可以方便地编写各种类型的文本文件，特别是各种类型的源代码文件。当然，你也可以使用自己熟悉的文本编辑器或集成开发环境（如 Eclipse、Zend Studio 等），只要能够方便、快捷地编辑源代码就可以了。

如果是第一次打开 Notepad++，会自动创建一个新文件，我们可以随时通过菜单“语言”选择文档中需要关注的源代码类型，相应的关键字可以进行高亮显示（如 PHP、HTML、CSS、JavaScript 等）。另一个小技巧是可以随时调节显示文本的尺寸，我们可以使用 Ctrl+"+" 放大字体尺寸，或者 Ctrl+"-"键缩小字体尺寸。

下面，我们编辑一个名为 index.php 的文件，并保存到 d:\phpweb\root 目录（或者是你指定的 PHP 网站根目录），文件内容如下：

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

然后，我们在浏览器中打开 localhost:8080（就是我们在 Apache 配置文件中设置的服务名称）。如果看到如下页面，则说明 Apache 和 PHP 已经正常工作了。



这个页面包含了大量的 PHP 配置信息，实际应用中，我们可以根据观察这些 PHP 的参数设置，并根据实际需要修改 php.ini 配置文件中相应的参数。请注意箭头所指的参数，这里是实际载入配置文件（php.ini）的路径，如果参数应用方面有问题，首先应该确认是否应用了正确的配置文件，然后，我们可以在这个配置文件中进行相应的修改。

我们知道，PHP 是服务器端的动态页面技术，在页面文件中，可以是完全由 PHP 代码组成，就像前面的代码一样，同时，也可以与 HTML 页面内容混合使用，如下面的代码就是在一个 HTML5 页面中嵌入了 PHP 代码块（/template.php）：

```
<?php
```

```
    // PHP 代码
?>

<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>PHP 页面模板</title>
</head>
<body>
<h1>页面内容</h1>
</body>
</html>
```

现在，我们在浏览器中通过“localhost:8080/template.php”查看这个页面，如下图：

无论在页面中如何应用 PHP 代码，它都应该包含在<?php 和?>之间，如果只有简单的一行代码，我们也可以将它们都写在一行，如：

```
<?php echo 'Hello PHP' ?>
```

此外，我们可以在 php.ini 中将 short_open_tag 参数设置为 On（默认为 Off），此时，可以使用简单的打开标签，如：

```
<? print 'Hello PHP' ?>
```

不过，使用<?php 作为开始标签可以让代码更清晰——这里是 PHP 代码！所以，本书中我们都将使用<?php 和?>来包含 PHP 代码。

1.6. 注释

在代码中添加注释是常见而又非常重要的工作，PHP 中，我们可以使用三种注释方法：

- //，C++风格注释，由//到行尾作为注释。
- #，脚本风格注释，从#符号到行尾作为注释。
- /* */，C 风格注释，在/*和*/之间的内容都作为注释，可以包含多行注释内容。

1.7. print 与 echo 语句

print 语句和 echo 语句都用于在页面中显示一些内容，但它们也有一些区别，首先看一下 print 语句的使用，它一次只能显示一个字符串，如下面的代码：

```
print "Hello PHP";  // 显示 Hello PHP
```

在开发过程，我们可能需要一次显示多个内容，如下面我们定义了三个字符串变量（更多内容将在下一章讨论）：

```
$str1 = "abc";
$str2 = "def";
$str3 = "ghi";
```

现在，我们需要做的就是在一条语句中将这三个字符串显示出来，使用 print 语句有两种方法：

```
print "$str1$str2$str3";
```

或

```
print $str1.$str2.$str3;
```

第一种方法是在双引号字符串中直接使用变量，它可以直接显示变量的内容；第二种方法是将三个字符串连接成一个字符串后显示。

使用 echo 语句显示这两个变量的内容，可以使用下面的代码。

```
echo $str1,$str2,$str3;
```

echo 语句可以一次显示多个字符串，每个字符串使用逗号分隔；与 print 语句相比，这种方法不需要进行字符串的连接操作，也不需要在字符串中查找变量。

实际应用中，我们可以根据显示内容的具体要求来确定使用 print 或 echo 语句。

1.8. 小结

本章，我们首先介绍了如何在 Windows 系统下搭建 Apache+PHP 网站环境，并了解了如何在页面中添加 PHP 代码，从下一章开始，我们就可以进行更多的学习和测试工作了。

2. PHP 编程基础

上一章，我们配置了 Apache 和 PHP 运行环境，并在页面中编写了简单的 PHP 代码，本章开始，我们将讨论 PHP 编程语言方面的一些基本知识，这些内容包括：

- 数据处理
- 常用运算符
- 字符串
- 条件控制语句
- 选择控制语句
- 循环控制语句
- 自定义函数
- 数组
- 日期与时间
- 引用外部文件

请注意，本章内容是进行 PHP 项目开发的基础，而且每一个示例的代码都不会太长，所以，我们强烈建议大家，特别是初学者都能够自己动手敲敲代码，以加深映像；此外，页面中测试的代码都应该包含在 PHP 页面（.php 文件）中的<?php 和?>之间，就像下面的结构。

```
<?php
// 测试代码
?>
```

对于简单的代码，我们可以直接在 d:/phpweb/root/index.php 文件或创建一个 test.php 文件进行测试。然后，可以在浏览器通过“http://localhost:8080”或“http://localhost:8080/test.php”查看运行结果。

2.1. 数据处理

每一种编程语言都有基本数据类型的处理方法，PHP 也是这样，下面我们就先来看一看 PHP 有哪些常用的基本数据类型。

2.1.1. 基本数据类型

整数

PHP 中的整数定义为有符号整数，但取值范围与平台有关，如在 32 位环境中，它就是一个 32 位的有符号整数；而在 64 环境中，就是 64 位有符号整数。

本书大部分示例中都使用了 64 位的操作系统和 PHP 环境，所以，我们可以认为 PHP 环

境下的整数为 64 位有符号整数，其取值范围为-2^{64}到$2^{64}-1$。

除了十进制整数，PHP 中同样可以使用十六进制（0x 前缀）和八进制（0 前缀），如：

- 0x1F，表示十六进制数 1F。
- 0123，表示八进制数 123。

浮点数

PHP 中的浮点数取值范围同样与平台有关，一般为 64 位浮点数。

NULL 值

NULL 值表示没有数据，这与数据库中的 NULL 值的含义是相同的。

布尔类型

布尔类型数据的值只能是 true 或 false。当其它类型的值转换为布尔类型数据时，会遵循以下基本规则：

- 整数转换为布尔型，只有 0 转换为 false 值，其它数值都会转换为 true 值。
- 浮点数转换为布尔型，只有 0.0 转换为 false 值，其它数值都转换为 true 值。
- NULL 值转换为布尔型，始终为 false 值。

由于布尔类型只支持 true 和 false 值，所以，它们的运算也相对比较简单，如：

- &&或 and 运算符，进行左右两个操作数的“与”运算，只有两个操作数都是 true 值时，运算结果才为 true 值，否则运算结果为 false。如 true && false 结果为 false，而 true && true 结果为 true。
- ||或 or 运算符，进行左右两个操作数的“或”运算，当两个操作数中的一个为 true 时，运算结果就是 true；只有两个操作数都是 false 时，运算结果才是 false 值。
- !运算符，取反运算。!true 为 false 值，!false 为 true 值。
- xor，进行左右两个操作数的异或运算。当两个操作数不同时，运算结果为 true，相同时运算结果为 false。如 true xor true 结果为 false，而 true xor false 结果为 true。

需要注意的是“与”运算和“或”运算，它们都具有“短路”功能。在“与”运算中，如果左操作数为 false，则不会计算右操作数，而是直接返回 false 值；而在“或”运算中，如果左操作数为 true，则不会计算右操作数，而是直接返回 true。实际应用中，我们可以将主要或重要的判断条件放在左操作数，这样有助于提高代码的运行效率。

此外，请注意 NULL 值与 false 值，当你使用“false == NULL;”语句，结果会是 true；但是，我们知道 false 值和 NULL 值的含义是不同的，所以，当你使用“false === NULL”语句时会返回 false，即它们不是全等的。

2.1.2. 变量

PHP 与 C、C++、C#、Java 等语言最大的不同在于，使用 PHP 变量时不需要事先声明，比如，在 C 语言中声明变量 i 并赋值为 1 时，会使用如下代码：

```
int i = 1;
```

而在 PHP 中，只需要使用如下代码：

```
$i = 1;
```

在 PHP 中，变量都使用$符号开始，后面的命名规则就与其它编程语言差不多了，即使用字母或下划线开头，然后由字母、下划线或数字组成。

我们可看到，在使用变量时，并没有指定变量的类型，那么，在运行当中，它是怎么工作的呢？答案是，PHP 变量会根据赋予的值来判断类型，如 1 被认为是整数、1.0 被认为是浮点数等。

此外，在调试过程中有一个非常实用的 var_dump()函数，它的作用就是显示变量的类型与值信息，此函数可以有一个或多个参数，如：

```
$intVal = 1;
$floatVal = 1.1;
$boolVal = true;
var_dump($intVal, $floatVal, $boolVal);
```

其显示信息为“int(1) float(1.1) bool(true)”。

在使用变量前，我们可使用 isset()语句结构判断变量是否存在或者是否包含有效数据（非 NULL 值），如：

```
$i = 1;
$j = NULL;
var_dump(isset($i)); // bool(true)
var_dump(isset($j)); // bool(false)
var_dump(isset($k)); // bool(false)
```

与 isset()语句结构相关的语句还有：

- unset()语句结构，取消变量的定义并自动释放其所占内存（如果可以的话）。
- empty()语句结构，当变量没有定义或为 null 值时，返回 true 值；当参数中包含有效数据时返回 false，此函数与 isset()函数的功能正好相反。

2.1.3. 常量

在 PHP 中定义常量时，可以使用 define()函数或 const 语句，其中，define()函数是

在代码运行时创建全局常量，这些常量可以在其所定义文件或引用此文件的页面中使用，其格式如下：

```
define(<常量名>,<常量值>,<是否忽略大小写>);
```

其中：

- <常量名>，指定常量名称，习惯用法是使用 C 常量风格，即所有字母大写，单词间使用下划线分隔。本书中将使用这一风格。
- <常量值>，指定常量的值。
- <是否忽略大小写>，这是一个可选参数，如果不指定此参数，则默认为 true，即在常量使用时忽略字母的大小写。

下面就是一个常量应用的代码：

```
define("MAX_NUMBER", 100);
echo MAX_NUMBER; // 显示 100
```

如果我们需要判断一个常量是否已定义，可以使用 defined()函数，如：

```
define("MAX_NUMBER", 100);
var_dump(defined("MAX_NUMBER")); // 显示 bool(true);
```

const 语句一般在类（class）中定义常量，用于模拟枚举类型，我们会在介绍面向对象编程时介绍。

2.1.4. 类型判断

PHP 中定义了一系列的数据类型判断函数，它们的返回结果都是布尔类型（true 或 false），常用的包括：

- is_integer()、is_int()和 is_long()函数，判断参数是不是一个整数。
- is_double()、is_float()和 is_real()函数，判断参数是不是一个浮点数。
- is_null()函数，判断参数是否为 NULL 值。
- is_bool()函数，判断参数是否为布尔类型。
- is_string()函数，判断参数是否为字符串类型。
- is_array()函数，判断参数是否为数组。稍后介绍数组相关内容。
- is_scalar()函数，当参数为整数、浮点数、字符串或布尔类型值时返回 true，否则返回 false。
- is_numeric()函数，参数可以转换为数值时返回 true，否则返回 false。

此外，我们还可以使用 gettype()函数直接获取参数的数据类型名称，如下面的代码。

```
$val = true;
echo gettype($val);  // 显示 boolean
```

2.1.5. 类型转换

数据类型转换前，我们可以使用前面介绍的类型判断函数做一些测试工作，当我们需要将一个值转换为另外一个类型时，有几种基本的方法可以使用：

第一种方法是使用 settype()函数，其中，参数一为原值，参数二为目标类型，使用字符串形式的类型名称。如下面的代码：

```
$val = '123';
echo gettype($val);
echo '<br>';
settype($val,'integer');
echo gettype($val);
```

代码显示结果如下图。

我们可以看到，使用 settype()函数实际上是修改变量自身的类型，如果我们不修改变量自身类型，而是需要一个类型转换后的数据，可以使用与 C 风格相似的转换方法，即在需要转换的数据前使用小括号包含目标类型名称，如下面的代码。

```
$val = '123';
echo gettype($val);
echo '<br>';
$num = (int)$val;
echo gettype($num);
```

本代码显示结果与上图是一样的。

在 PHP 中，还有第三种方法可以用于数据类型转换，即使用相应类型的转换函数，如：

- intval()函数，将参数一转换为整数，参数二为可选，用于指定原数据的进制，默认为十进制，函数将返回转换后的整数数据，如果转换失败则返回 0。
- floatval()函数，将参数转换为浮点数，函数将返回转换后的浮点型数据，如果转换失败则返回 0。
- strval()函数，将参数转换为字符串形式。
- boolval()函数，将参数转换为布尔类型。

下面的代码演示了这几个函数的使用。

```
$str = 'F';
$num = intval($str,16);
var_dump($num);  // int(15)，十六进制数 F 转换为十进制数 15
```

```
var_dump(floatval($str)); // float(0)
var_dump(strval($num));   // string(2)"15"
var_dump(boolval($str)); // bool(true)
```

此外，关于浮点数，我们可以使用 round()函数确定它的小数位，对于多出的小数位，会进行四舍五入操作，如果指定的小位数多于原数小数位，则会显示原数值。如下面的代码：

```
echo round(2.125, 1); // 显示 2.1
echo '<br>';
echo round(2.125, 2); // 显示 2.13
echo '<br>';
echo round(2.125, 4); // 显示 2.125
```

另外一个常用的格式化数值的函数是 number_format()，它会返回一个格式化后的字符串，其定义如下：

```
string number_format(<原值>, <小数位>, <整数与小数分隔符>, <千位分隔符>)
```

下面的代码演示了常用的几种方法：

```
$val = 123456.92789;
echo number_format($val);   // 显示 123,457
echo '<br>';
echo number_format($val, 2);   // 显示 123,456.93
echo '<br>';
echo number_format($val, 4, '.', ',');   // 显示 123,456.9279
echo '<br>';
echo number_format($val, 3, '.', '');   // 显示 123456.928
```

代码中分别使用了四个 number_format()函数：

- number_format($val)，使用一个参数，指定需要转换的数值，默认情况下会对小数部分进行四舍五入，使用逗号作为千位分隔符。
- number_format($val, 2)，使用两个参数，其中参数二为需要保留的小数位，其余部分进行四舍五入，使用逗号作为千位分隔符。
- number_format($val, 4, '.', ',')，使用四个参数，参数三设置整数部分与小数的分隔符，参数四设置逗号作为千位分隔符。
- number_format($val, 3, '.', '')，使用四个参数，参数三设置整数部分与小数的分隔符，参数四设置千位分隔符为空，即不使用千位分隔符。

关于数据类型的转换，我们不得不提的是 PHP 中强大的自动转换功能，如使用 print 或 echo 语句显示内容时，无论你指定显示的是什么类型的内容，这两个语句都能自动将这些内容转换成文本内容并显示。不过，这并不包括没有定义__toString()方法的对象，下一章，我们将会了解详细情况。

在实际应用中，我们应该多思考，并进行相应的测试工作，以掌握 PHP 在数据类型转换

中的灵活性，同时，也要注意转换过程中可能出现的问题。

2.2. 常用运算符

本节我们给出 PHP 中一些基本的运算符，对于一些特殊的运算符，如布尔运算、字符串操作等，会在相关章节中介绍。

2.2.1. 算术运算符

与大多数编程语言一样，PHP 中的算术运算符包括以下五种：

- +，加法运算。
- -，减法运算。
- *，乘法运算。
- /，除法运算。
- %，取模运算，更直观的名称是取余数运算，此运算符只使用于整数，必要时会自动转换数据类型。

2.2.2. 比较运算符

比较运算符的结果为布尔类型，一般用于条件判断，PHP 中的比较运算符包括：

- ==，等于，这只是形式上的等于，如"99" == 99 会返回 true。
- !=，不等于。
- <，小于。
- >，大于。
- <=，小于等于。
- >=，大于等于。
- ===，全等，数据类型和值都相等才会返回 true，如"99"===99 返回 false。如果是两个对象进行全等比较，则必须是同一实例才返回 true，否则返回 false（即使两个对象是相同类型）。
- !==，不全等。

2.2.3. 位运算符

位运算应用于整数，如果运算数不是整数类型，则会自动转换。PHP 中的位运算符包括：

- &，按位与运算。
- |，按位或运算。
- ~，按位非运算。

- ^，按位异或运算。
- <<，左移运算。
- >>，右移运算。

位运算是对整数的二进制编码进行运算，在一般的应用开发中并不经常使用，如果有需要，你可以在 php.net 网站参考完整的说明文档。

2.2.4. 赋值运算符

除了标准的赋值运算符（=），PHP 也支持一些常用的组合赋值运算符，如：+=、-=、*=、/=、&=、|=、^=、<<=、>>=。如：

```
$i = 1;
$i += 2; // 相当于 $i = $i + 2
echo $i; // 显示 3
```

2.2.5. 递增与递减运算符

我们首先讨论一下递增运算，包括前递增与后递增，先看下面的代码。

```
$i = 1;
$i++;
$j = 1;
++$j;
echo $i; // 显示 2
echo $j; // 显示 2
```

我们可以看到，代码中，无论是前递增运算还是后递增运算，运算后的变量都会加 1，那么，它们的区别是什么呢？

答案是，前递增运算和后递增运算时，其表达式的值不同，区别在于：

- 前递增运算，表达式是变量加 1 的值，即和计算后的变量值相同。我们可以理解为先运算后使用变量值。
- 后递增运算，表达式是变量的原值。相应的，可以理解为先使用变量值，再进行加 1 运算。

如下面的代码。

```
$i = 1;
$j = 1;
echo ++$i; // 显示 2
echo $j++; // 显示 1
```

在使用递增运算时，如果是只使用计算后的变量值，那么，前递增运算和后递增运算的结果是一样的；但是，需要使用递增运算表达式的值时，就应该非常小心地区分它们。

递减运算同样分别前递减运算和后递减运算，只是执行的是变量减 1 的操作，而其它特点则与递增运算相似。

2.2.6. ?:运算符

?:是一个三元运算符，它包括三个运算数，如：

```
<表达式 1>?<表达式 2>:<表达式 3>
```

其中：当<表达式 1>结果为 true 时，返回<表达式 2>的内容，否则返回<表达式 3>的内容。如下面的代码：

```
$num = 2;
echo $num%2==0 ? '偶数' : '奇数';  // 显示 偶数
```

2.3. 字符串

PHP 中的字符串是一种类 C 风格的字符串，只不过已经得到了很好的封装。在 PHP 中，可以将字符串包含在一对双引号或一对单引号之中，我们先来看一看双引号形式的字符串都有哪些特点。

如果使用过 C、C++、C#或 Java，你一定知道在字符串中会使用一些转义字符，在 PHP 中的字符串中同样有这样的转义字符，以下就是在双引号定义的字符串中可以使用的转义字符。

转义字符	说明
\0	ASCII 值为 0 的字符（NULL 值）。
\n	换行符。
\r	回车符，即回到行首。
\t	制表符。
\$	将$符号当作普通符号，而不是变量前缀。
\″	双引号。
\\	反斜线（\）符号。
\{n}	n 将以八进制显示。
\x{n}	n 将以十六进制显示。

在这里需要注意的是\$转义，在 PHP 双引号字符串中，我们可以直接使用变量名，而显示时会自动替换为变量的值，如：

```
$num = 15;
print "$num"; // 显示 15
```

如果我需要显示$符号，就需要使用\$转义，如下面的代码：

```
$num = 15;
print "\$num = $num"; // 显示$num = 15
```

另一种常用的字符串是通过一对单引号来定义，这种字符串中只支持两种转义字符（\'和\\），并且不支持显示变量值的功能，所以，在处理单引号字符串时，会减少很多工作，如查找字符串中的变量名和更多的转义字符等。

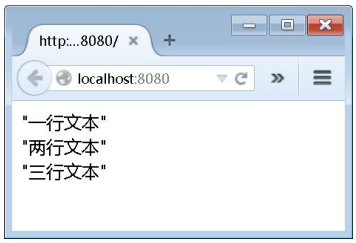
在 PHP 中，还有一种定义文本的方法是使用定界符<<<，常用格式如下：

```
<<<标签名称
<文本内容>
标签名称;
```

在使用定界符定义字符串时，结束处的标签名称应该顶格书写。此外，我们同样可以使用变量名，但与双引号字符串不同的是，大多转义字符并不需要使用，如双引号、单引号等，而对于变量前的$符号，则还是需要使用\$进行转义。下面的代码就是<<<定义文本的一个示例。

```
echo
<<<DEFINE_TAG
"一行文本"<br>
"两行文本"<br>
"三行文本"
DEFINE_TAG;
```

代码显示结果如下图。



请注意，代码中标签使用大写字母只是为了引起注意，实际上你也可以使用小写的标签。

2.3.1. 字符串连接

很多编程语言中的字符串连接都会使用+运算符，但在 PHP 中不是，在 PHP 中，如果我们需要将两个字符串连接到一起，应使用串联运算符（.）。我们看几个例子：

```
$num = 1;
$str = '2';
```

```
echo $num+$str; //显示 3
echo '<br>';
echo $num.$str; //显示 12
```

我们可以看到，PHP 会很智能地判断运算目的，使用+运算符就是数值相加，PHP 会将非数值转换为数值后计算；而使用.运算符就是连接字符串，PHP 会将数值转换为字符串进行连接操作。

2.3.2. 访问字符串成员

在 PHP 中，我们可以使用从 0 开始的索引来获取或重写字符串中的字符，如：

```
$str = 'abcdefg';
echo $str{2}; //  显示 c
```

我们可使用{}或[]来指定索引值，不过，建议使用更新的{}，这样可以区分字符串索引和数组索引（稍后讨论），提高代码的可读性。

此外，当我们获取字符串中的字符时，索引必须在允许的范围内，即从 0 到字符串字符数-1；在重写指定位置的字符时，如果索引超出了已有的范围，字符会写到指定索引的位置，而中间空的字符位置会使用空格填充。如下面的代码：

```
<?php
$str = 'abc';
$str{6} = 'g';
echo $str; //  显示 abc      g
?>
```

请注意，在 PHP 中处理字符串时，字母 g 前会有三个空格，但是，显示到页面中就未必了，我们知道，页面中会忽略不必要的空白字符，如空格、回车、制表符等；不过，当我们查看页面源代码时，g 前面还是有三个空格的。

接下来，我们了解一些常用的字符串函数。

2.3.3. 获取字符数

strlen()函数用于获取字符串中的字符数量，如：

```
$str = 'abcdefg';
echo strlen($str); //  显示  7
```

请注意，如果字符串中包括中文，strlen()函数返回的结果并不是正确的字符数量，此时，我们可以使用 mbstring 系列函数中的 mb_strlen()函数。在代码中使用 mbstring 系列函数，你必须已经在 php.ini 配置文件中打开此功能，如：

```
extension = php_mbstring.dll
```

此外，如果修改了 php.ini 配置文件，请记得需要重启 Web 服务（如 Apache 服务）才

能生效。

下面的代码演示了 mb_strlen()函数的使用。

```
echo mb_strlen("中国");  // 显示 2
echo mb_strlen("中国的 PHP 社区");  // 显示 8
```

mb 是 multibyte（多字节）的缩写，mbstring 系统函数则可以用于处理中文、日文等多字节字符，更多的 mbstring 系列函数列表，你可以在 php.net 中搜索“mb_”进行查询。

2.3.4. 字母大小写转换

- PHP 中字母大小写转换相关的函数有：
- strtoupper()，将字符串中的字母都转换为大写字母，并返回转换结果。
- strtolower()，将字符串中的字母都转换为小写字母，并返回转换结果。
- ucfirst()，如果字符串第一个字符是字母，则将其转换为大写，并返回转换结果。
- ucwords()，如果每一个单词的第一个字符是字母，则将它们转换为大写，并返回转换结果。

2.3.5. 去空白字符

去空白字符函数可以将字符串前部或结尾处的空格、制表符等不可见字符去除；比如，我们获取用户输入的表单数据，就可以使用这些函数进行处理后保存到数据库，从而避免保存多余的空白字符。常用的函数有：

- trim()，去除字符串前后的空白字符。
- rtrim()，去除字符串结尾的空白字符。
- ltrim()，去除字符串前部的空白字符。

2.3.6. 字符串比较

strcmp()函数，按 ASCII 码比较两个字符串，相同返回 0，否则返回非零值，此函数区分字母的大小写。如 strcmp('abc','abc')结果为 0。

strcasecmp()函数，按 ASCII 码比较两个字符串，相同返回 0，否则返回非零值，此函数不区分字母的大小写。如 strcasecmp('abc','Abc')结果为 0。

strnatcmp()函数，按自然排序比较两个字符串，此函数区分字母大小写，如 strcmp('9','10')返回 1,即'9'的 ASCII 码值大于字符 2 中的第一个字符'1'的 ASCII 码值；而 strnatcmp('9','10')返回-1，即 9 小于 10。

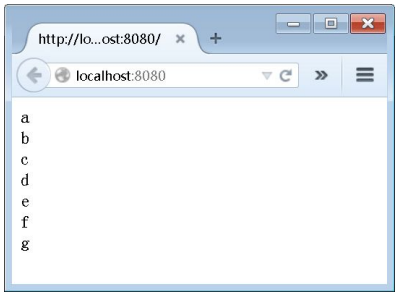
strnatcasecmp()函数，按自然排序比较两个字符串，此函数不区分字母大小写。

2.3.7. 连接与分割

explode()函数用于分割字符串，并返回分割后的数组，如：

```
$str = 'a,b,c,d,e,f,g';
$arr = explode(',',$str);
foreach($arr as $value)
{
    echo $value,'<br>';
}
```

代码中的 explode()函数，第一个参数为分隔字符串的字符或字符串，第二个参数为需要分隔的字符串；代码执行结果如下图：



关于数组的概念稍后讨论。

此外，explode()函数还可以有第三个参数，用于指定返回的数组成员数量，如前面的代码修改为 explode(',',$str,3)，则只会显示 a、b、c。

implode()和 join()函数功能一样，是将数组连接成一个字符串，并指定连接字符或字符串，如：

```
$arr = array('a','b','c','d','e','f','g');
$str = implode(',',$arr);
print $str;
```

代码会显示 a,b,c,d,e,f,g。

2.3.8. 截取字符串

substr()函数用于载取字符串中的一部分，它一般有三个参数：

- 参数一为原字符串。
- 参数二为开始截取的索引位置（首字符的索引值为 0）。
- 参数三为截取的字符数，这是一个可选参数，如果不指定，将截取从参数二指定的位置到字符串结束的全部内容。

如下面的代码：

```
$str = 'abcdefg';
print substr($str,2,3); //  显示 cde
print substr($str,2); //  显示  cdefg
```

2.3.9. 查找与替换

strstr()与 strchr()函数用于在一个字符串（参数一）中查找匹配的子字符串（参数二）。如果有匹配的内容，函数将返回从第一次出现一直到末尾的所有内容，如果在参数一中不包含参数二的内容，则返回空字符串。如下面的代码。

```
$str = 'abcdefg';
print strstr($str, 'cde');  // 显示 cdefg
```

stristr()函数，功能与 strstr()函数基本一样，唯一的区别在于 strstr()函数区分字母大小写，而 stristr()函数不区分字母大小写。

strrchr()函数，从参数一的末尾向前查找，此函数区分字母大小写。如下面的代码。

```
$str = 'abcdefgcde';
print strstr($str, 'cde');  // 显示 cdefgcde
print strrchr($str, 'cde');  // 显示 cde，这是末尾的 cde
```

strpos()函数，同样用于在一个字符串中查找一个子字符串，只不过此函数将返回子字符串首次出现的索引位置，此函数区分字母的大小写。如下面的代码。

```
$str = 'abcdefg';
print strpos($str,'cde');  // 显示 2，即 c 首次出现的索引位置
```

strrpos()函数，功能与 strpos()函数相似，只是 strrpos()函数会返回子字符串最后一次出现的索引位置。如下面的代码：

```
$str = 'abcdefgcde';
print strrpos($str,'cde'); // 显示 7
```

使用 strpos()和 strrpos()函数时请注意，如果没有找到匹配的子字符串，则函数会返回 false 值。但是，我们知道索引值 0 转换为布尔型时也是 false，所以，我们进行判断时，应用使用===运算符，如下面的代码：

```
$str = 'abcdefg';
if(strpos($str,'hij') === false)
    print '没有匹配的字符串';
```

代码运行会显示“没有匹配的字符串”，再看下面的代码。

```
$str = 'abcdefg';
if(strpos($str,'abc') == false)
    print '没有匹配的字符串';
else
    print '有匹配的字符串';
```

此代码同样会显示“没有匹配的字符串”，这显然是不对的，如果我们将==运算符修改为===运算符，则会显示“有匹配的字符串”。

str_replace()函数，将参数三指定的字符串中，原内容（参数一）替换为指定的内容（参数二），并返回修改后的结果。如下面的代码：

```
$str = 'abcdefg';
print str_replace('cde','***',$str); // 显示 ab***fg
```

代码中的功能就是将$str 变量中的 cde 替换成***，并返回替换后的结果。如果没有找到参数一中指定的内容，则返回参数三的全部内容。

str_replace()函数区分字母的大小写，如果不需要区分字母大小写，可以使用 str_ireplace()函数。

substr_replace()函数，用于将一个字符串（参数一）中指定位置的内容替换成指定的内容（参数二），而指定替换的开始位置由参数三指定，参数四为可选参数，用于指定替换的字符数量。如下面的代码。

```
$str = 'abcdefg';
print substr_replace($str,'*',3); // 显示为 abc*
```

代码的作用是将从索引位置 3 开始的所有内容都替换为*。

下面的代码使用了第四个参数。

```
$str = 'abcdefg';
print substr_replace($str,'*',3,3); // 显示为 abc*g
```

代码的作用是将从索引位置 3 开始的三个字符（本例为 def）替换为*。

2.4. 条件控制语句

在软件开发过程中，代码执行的流程控制是非常重要的一项工作，传统的流程控制语句包括条件语句、选择语句和循环语句，本节我们就先来看看 PHP 中的条件控制语句。

2.4.1. if 语句

和大多数编程语言一样，在 PHP 中的条件控制语句同样使用 if 关键字，并且使用方法也是与 C 风格相似的，如：

```
if(<条件>)
{
    <语句块>
}
```

如果<条件>成立（为 true 时）则执行<语句块>，如果条件不成立（为 false）时，则继续执行“}”后面的语句，如下面的语句，其功能是判断一个整数是不是偶数。

```
$num = 8;
if($num%2==0)
{
```

```
    echo "$num","是一个偶数";
}
```

此代码将显示“8 是一个偶数”。

如果语句块只有一条语句，就像上面的代码时，我们可以省略{和}，如：

```
$num = 8;
if($num%2==0)
    echo "$num","是一个偶数";
```

或者，你还可以将它们放在一行，如：

```
if($num%2==0) echo "$num","是一个偶数";
```

2.4.2. if-else 语句

前面的 if 语句的示例中，当条件成立时会显示一条信息，但条件不成立时呢？很多时候，我们必须对条件不成立时的情况作出相应的处理，此时，可以使用 if-else 语句结构，其基本使用方法如下：

```
if(<条件>)
{
    <语句块 1>
}
else
{
    <语句块 2>
}
```

当条件成立时执行<语句块 1>，条件不成立时执行<语句块 2>。如下面的代码。

```
$num = 9;
if($num%2==0)
{
    echo $num,'是一个偶数';
}
else
{
    echo $num,'不是一个偶数';
}
```

代码会显示“9 不是一个偶数”。

2.4.3. if-elseif 语句

当我们的代码逻辑中包含两个或更多的条件时，可以使用 if-elseif 语句，它的基本格式如下：

```
if(<条件 1>)
```

```
{
    <语句块 1>
}
elseif(<条件 2>)
{
    <语句块 2>
}
elseif(<条件 n>)
{
    <语句块 n>
}
```

如下面的代码，我们将根据分数进行评级。

```
$score = 88;
if($score>=90){
    echo '优秀';
}elseif($score>=80 && $score<90){
    echo '好';
}elseif($score>=70 && $score<80){
    echo '良';
}elseif($score>=60 && $score<70){
    echo '及格';
}else{
    echo '不及格';
}
```

代码会显示“好”。请注意代码的最后，我们添加了 else 语句，在实际应用中，可以根据需要选择是否在 if-elseif 语句结构中使用 else 语句，但应注意，else 语句只能在条件语句结构的最后，并且只能出现一次，这和 if-else 语句中的使用是一致的。

2.4.4. 复杂条件与嵌套

在 if-elseif 的示例中，我们在条件中使用了与运算符（&&），也就是两个条件都成立时，整个条件表达式的结果才为 true，在实际应用中，我们还可能需要写更复杂的判断条件，此时，应注意与运算（&&）、或运算（||）和非运算（!），以及各比较运算符的综合应用，使用()将条件进行有效的组合会让代码可读性更强，同时也会更安全，这样，你就不会因为忘了运算符优先级而导致运算顺序错误，这也是我们在本书中没有提及运算符优先级的原因。

如下面的代码，功能是判断一个年份是否为闰年。

```
$year = 2008;
if(($year%100!=0 && $year%4==0) ||
    ($year%100==0 && $year%400==0))
{
    echo $year,'年是闰年';
```

```
}
else
{
    echo $year,'年不是闰年';
}
```

代码会显示“2008 年是闰年”。条件中的执行逻辑有两种情况，如果有其中一种成立，指定的年份就是闰年，第一种情况是年份不能被 100 整除时，如果能被 4 整除，则年份是闰年；第二种情况是年份能被 100 整除，则必须也能被 400 整除才是闰年。

如果你觉得前面的条件比较复杂，我们还可以将其分解，使用嵌套条件语句来完成，如下面的代码。

```
$year = 2008;
$isLeapYear = false;
if($year%100!=0)
{
    if($year%4==0)
    {
        $isLeapYear = true;
    }
}else
{
    if($year%400==0)
    {
        $isLeapYear = true;
    }
}
if($isLeapYear)
{
    echo $year,'年是闰年';
}
else
{
    echo $year,'年不是闰年';
}
```

虽然代码长了很多，但可以帮助我们理解条件语句的嵌套使用。

此外，请注意$isLeapYear 变量的使用，如果不使用此变量，相信代码的结构会更复杂，你可以自己动手试一试。布尔型变量的这种使用方法，在复杂的条件判断或数据检查操作中是很常见的。

2.5. 选择控制语句

选择控制语句相对简单，使用 switch 语句结构即可，其基本格式如下：

```
switch(<条件>)
{
case <值 1>:
    <语句块 1>
case <值 2>:
    <语句块 2>
case <值 n>:
    <语句块 n>
default:
    <语句块 n+1>
}
```

switch 语句结构中只有一个条件，而这个条件可能会有多个值，值的类型可以是数值，也可以是字符串；我们使用 case 语句处理不同值的执行代码，而 default 语句则用于处理没有对应值的情况，它的功能与 if 语句中的 else 功能相似，可以根据实际情况选择使用。

如下面的代码，其功能是根据颜色英文名显示其对应的中文名称。

```
$color_en = 'red';
switch($color_en)
{
    case 'red':
    {
        echo '红色';
        break;
    }
    case 'green':
    {
        echo '绿色';
        break;
    }
    case 'blue':
    {
        echo '蓝色';
        break;
    }
    default:
    {
        echo '未知颜色';
        break;
    }
}
```

代码会显示“红色”。请注意代码中的 break;语句，你可以尝试删除它看看运行的结果如何。没错，代码会执行对应值以后所有的语句，直到有 break 或其它终止运行的语句（如 return、exit 等）；而在这里，break 语句的作用就是在适当的时候终止 switch 语句结构的工作。不过，有些时候，我们也可以利用没有 break 语句时的工作特点来完成一些工作，比如计算每个月的天数，如下面的代码。

```
$year = 2008;
$month = 2;
$daysOfMonth = 0;
switch($month)
{
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        $daysOfMonth = 31;
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        $daysOfMonth = 30;
        break;
    case 2:
    {
        if(($year%100!=0 && $year%4==0) ||
            ($year%100==0 && $year%400==0))
            $daysOfMonth = 29;
        else
            $daysOfMonth = 28;
    }break;
}
echo $year,'年',$month,'月有',$daysOfMonth,'天';
```

代码会显示“2008 年 2 月有 29 天”。

2.6. 循环控制语句

循环语句的作用就是在条件满足的情况下，可以执行多次相同或相似的任务，PHP 中的循环语句包括 for、while、do-while 和 foreach 语句结构，下面分别介绍。

2.6.1. for 语句

for 语句结构一般用于执行确定循环次数的操作，其条件包括三个部分，如：

```
for(<初始化循环控制变量>; <循环执行条件>; <控制变量的变化>)
{
    <语句块>
}
```

如下面的代码，其功能是计算 1 到 100 的和。

```
$sum = 0;
for($i=1; $i<=100; $i++)
{
    $sum += $i;
}
echo $sum; // 显示 5050
```

下面的代码会计算 1 到 100 中偶数的和。

```
$sum = 0;
for($i=2; $i<=100; $i+=2)
{
    $sum += $i;
}
echo $sum; // 显示 2550
```

2.6.2. while 语句

while 语句结构会根据指定的条件来执行循环操作，其一般使用格式如下。

```
while(<条件>)
{
    <语句块>
}
```

当<条件>满足时，会执行<语句块>，当<条件>不满足时会退出循环结构。如下面的代码，其功能是同样是计算 1 到 100 的和。

```
$sum = 0;
$i = 1;
while($i<=100)
{
    $sum += $i;
    $i++;
}
```

```
echo $sum; // 显示 5050
```

2.6.3. do-while 语句

do-while 语句结构与 while 语句结构的功能相似，只不过判断的条件放在了循环操作的后面，也就是说 do-while 循环至少会执行一次。其基本应用格式如下。

```
do
{
    <语句块>
}while(<条件>)
```

我们同样使用这个语句来完成 1 到 100 累加的计算，如下面的代码。

```
$sum = 0;
$i = 1;
do
{
    $sum += $i;
    $i++;
}while($i<=100);
echo $sum; // 显示 5050
```

使用 do-while 语句结构时应注意，由于循环最少会执行一次，所以，必须保证在第一次执行循环时不会出现异常，否则，还是使用 while 语句结构比较安全。

2.6.4. 循环中的 break 和 continue 语句

在 switch 语句结构中，我们已经了解了一些 break 语句的作用，在那里，它的作用就是终止 switch 语句结构，而在循环语句中，无论是 for、while，还是 do-while 语句结构，我们都可以使用 break 语句，它的作用就是终止当前的循环结构。

在循环语句结构中，另一个需要注意的是 continue 语句，它的作用是终止本次循环的执行，而继续执行下一次循环（如果条件为 true 的话）。如下面的代码，我们将使用 continue 语句来实现 1 到 100 中偶数的累加计算。

```
$sum = 0;
for($i=1; $i<=100; $i++)
{
    if($i%2 != 0) continue;
    $sum += $i;
}
echo '1 到 100 中偶数的和是',$sum;
```

前面，我们看到的是 break 和 continue 语句最基本的应用，在多重循环中，我们还可以使用 break 和 continue 语句执行更精确地控制，其中有两种基本用法：

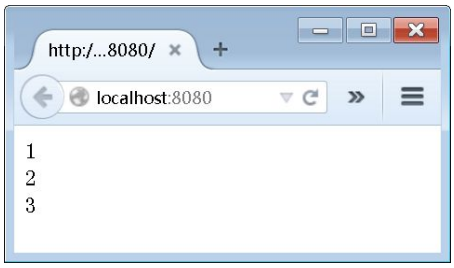
- break n 或 continue n。其中，n 是一个整数，用于指示终止第几层循环，而 n 为 1 时，其效果与单独使用 break 和 continue 语句的效果是一样的。
- break <变量>或 continue <变量>。其中，<变量>是指循环控制变量，也就是指定 break 和 continue 语句终止的是哪一个控制变量所在的循环结构。

2.6.5. foreach

foreach 语句又可以称为迭代语句结构，它并不是按条件来执行，而是逐一访问对象中的所有成员，我们可以在结构中对这些成员进行相同或相似的操作，如数组成员的访问，下面的代码将显示数组中每一个成员。

```
$arr = array(1,2,3);
foreach($arr as $value)
{
    echo $value,'<br>';
}
```

代码显示结果如下图。



像数组这样可以使用 foreach 语句访问的对象，是一种叫作“迭代器”的结构实现的，在我们创建类型中，如果需要使用 foreach 语句访问，可以让其实现 Iterator 接口，这个接口的成员包括：

- rewind()方法，指向第一个成员。
- current()方法，返回当前成员。
- key()方法，当前成员的键（或索引）。
- next()方法，指向下一个成员，此操作前应使用 valid()方法进行判断。
- valid()方法，返回 bool 类型，判断是否还有下一个成员。

接口与其实现等相关操作请参考“接口”一章。

2.7. 自定义函数

前面，我们已经使用过不少 PHP 中定义的内置函数，使用这些函数，我们可以完成很多基础工作，但在软件开发过程中，我们经常会封装一些自己的代码，以完成特定的工作。在 PHP 中，封装代码主要有两种基本方法，使用自定义函数和创建类（class）；本节，我们将

介绍如何创建自己的函数，在下一章，将专门介绍与类相关的内容。

首先，在我们需要定义一个函数之前，可以使用 function_exists()或 is_callable()函数判断函数名是否已存在，参数是一个有效的函数名称，函数存在时返回 true，否则返回 false。如下面的代码。

```
var_dump(function_exists('substr')); // 显示 bool(true)，即函数名存在
```

在使用 PHP 内置函数时，我们可观察到这些函数的命名规则是都使用小写字母，每个单词使用下划线进行连接；为了区分内置函数和自定义函数，我们约定：自定义函数名称不使用下划线，而是采用首字母小写，然后每个单词首字母大写的形式。

2.7.1. 创建函数

在 PHP 中创建函数使用 function 关键字，一个最简单的函数定义和调用如下面的代码：

```
function myFirstFunction()
{
    echo '我的第一个函数';
}
myFirstFunction(); // 调用函数
```

代码会显示“我的第一个函数”，这就是一个最简单的自定义函数。

2.7.2. 参数与返回值

函数中，参数的概念并不难理解，我们使用参数将需要的数据带入到函数中，根据这些数据进行相应的处理后，就可以返回函数的处理结果；如下面的函数，它的功能就是计算圆的周长。

```
function getPerimeterOfCircle($radius)
{
    return $radius * 2 * M_PI;
}
echo '圆的半径为', $radius, ',其周长为',getPerimeterOfCircle(10);
```

函数中，我们使用 return 语句返回计算结果，这也是函数的返回值。本代码运行会显示“圆的半径为 10 , 其周长为 62.831853071796”。如果你觉得小位数太多，可以使用 round()函数进行四舍五入，如下面的代码。

```
echo round(getPerimeterOfCircle(10), 2); // 显示 62.83
```

按引用传递参数

有些时候，我们可能需要在函数内对带入的变量进行实际的修改，而此时，参数的定义就应该添加&符号，其含义为参数按引用传递。如下面的代码。

```
function addOne(&$num)
{
```

```
    $num++;
}
$num1 = 10;
addOne($num1);
echo $num1; // 显示 11
```

如果我们将“&$num”修改为“$num”，则显示结果就是 10；这是因为，在默认情况下，参数带入的是变量的一个副本，对这个副本的操作不会影响变量原有的值。

请注意，如果参数是一个对象，默认情况下就是传递对象的引用，在函数内对象的操作都会实际反映到原对象上，下面的代码演示了这一点，大家先看一下，从下一章开始，我们详细介绍面向对象编程后，这些代码就非常容易理解了。

```
class CTest
{
    public $num;
}

function addOne(CTest $obj)
{
    $obj->num++;
}

$obj = new CTest();
$obj->num = 10;
addOne($obj);
echo $obj->num; // 显示 11
```

CTest 是一个类，而$obj 同是一个 CTest 类的实例（即对象），addOne()函数会对 CTest 对象中的$num 成员进行加 1 操作。

利用参数的按引用传递特性，我们还可以实现一个函数返回多个结果的功能，如下面的代码。

```
function func($str, &$sum)
{
    $count = strlen($str);
    $num_count = 0;
    $sum = 0;
    for($i=0; $i<$count; $i++)
    {
        if(is_numeric($str{$i}))
        {
            $num_count++;
            $sum+=intval($str{$i});
        }
    }
```

```
    return $num_count;
}

$str = 'a5e3k2nki1';
$sum = 0;
$num_count = func($str,$sum);
echo '字符串包含', $num_count, '个数字,它们的和是',$sum;
```

代码运行结果显示“字符串包含 4 个数字,它们的和是 11”。

我们定义的 func()函数，其返回值是字符串中的数字的个数，我们同时通过第二个参数$sum 返回了这些数字的和。

可选参数

在定义函数时，如果我们给参数设置了默认值，即它就成为一个可选参数，在调用函数时就可以不指定这个参数；可选参数可以有多个，但它们必须放在参数列表的最后，如下面的代码。

```
function printInteger($min, $max, $step=1)
{
    for($i=$min; $i<=$max; $i+=$step)
    {
        echo $i,',';
    }
}

printInteger(1,10);   // 显示 1,2,3,4,5,6,7,8,9,10,
echo '<br>';
printInteger(1,10,2);   // 显示 1,3,5,7,9,
```

关于 mixed 类型

查看 PHP 文档时，我们可以看到很多函数或方法返回的类型是混合类型，即 mixed；这说明，这些函数和方法返回的类型可以不止一种，比如一个函数需要返回远程资源（如 string 类型），但是，但远程连接错误时，就可以返回 false 值来通知调用者。

在我们定义函数时，并不需要指定函数的返回值，但是，我们在项目中还是应该对函数的返回值类型做出约定，以便正确调用。

2.8. 数组

PHP 中的数组使用哈希表（hashtable）的形式构成，其功能是非常强大的，在哈希表中，每一个数据项都由两个部分组成，即键（key）和值（value）；键就是数据的名称，而值就是真正的数据了。

2.8.1. 创建与访问数组

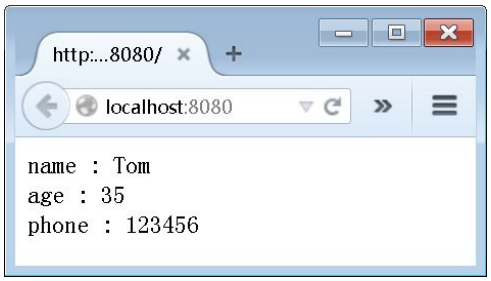
在 PHP 中，定义数组使用 array()语句结构。如下面的代码，我们将创建一个名片类的数组。

```
$arr = array('name'=>'Tom','age'=>35,'phone'=>'123456');
```

代码中，我们在数组中定义了三个数据项，分别是 name、age 和 phone，然后，我们可以通过 foreach()访问全部成员，如下面的代码。

```
foreach ($arr as $key=>$value)
{
    echo $key, ' : ', $value, '<br>';
}
```

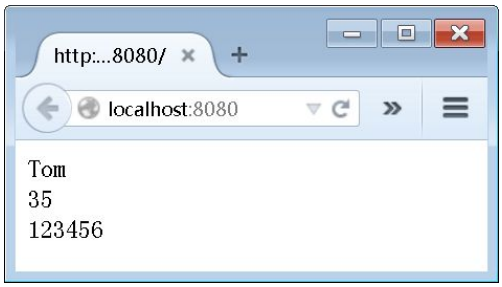
代码在页面中的显示结果如下图。



此外，我们还可以通过数组的键来访问其成员，此时使用[]指定键，如下面的代码。

```
echo $arr['name'], '<br>';
echo $arr['age'], '<br>';
echo $arr['phone'], '<br>';
```

代码执行结果显示如下图。



很多情况下，我们可能需要使用向 C 或 Java 风格的数组，也就是从 0 开始的索引值来表示数组成员，在 PHP 中，如果我们使用 array()结构创建数组时不指定键，则创建的就是这样的数组，如下面的代码。

```
$arr = array('a','b','c','d','e','f','g');
$count = count($arr);
for($i=0; $i<$count; $i++)
{
    echo $arr[$i];
}
```

当然，这样的数组同样可以使用 foreach 语句访问，如下面的代码：

```
$arr = array('a','b','c','d','e','f','g');
foreach($arr as $value)
{
    echo $value;
}
```

这两段代码在页面中会显示 abcdefg。也许你还记得代码中的功能可以使用 implode()或 join()函数来实现，如下面的代码。

```
$arr = array('a','b','c','d','e','f','g');
$str = implode("",$arr);
echo $str;
```

如果需要数组的索引值不是从 0 开始，可以在创建数组时指定，如：

```
$arr = array(1=>'a','b','c','d','e','f','g');
echo $arr[1]; // 显示 a
```

这样，"a"成员的索值就是是 1，其后面的成员索引值也会依次递增。

在开发过程中，如果我们想一次查看数组的全部内容，还可以使用 print_r()函数，它用于按规则显示对象的内容，而这在代码调试过程中是很有用的。如下面的代码。

```
$arr = array('name'=>'Tom','age'=>35,'phone'=>'123456');
print_r($arr);
```

其显示结果如下图。

如果数组成员是具有相同间隔（步长）的数据内容，我们还可以使用 range()函数快速创建这个数组，如下面的代码。

```
$arr1 = range(1,5); // 默认间隔为 1，数组成员为 1,2,3,4,5
$arr2 = range(1,5,2); // 指定间隔为 2，数组成员为 1,3,5
```

除了数字，range()函数还可以创建字符类的数组，但这似乎并不经常用到，如：

```
$arr = range('a','d'); // 数组成员为 a,b,c,d
```

2.8.2. 修改数组成员

对数组的单个成员修改是很方便的，我们可以通过键或索引值来进行相应的操作，如：

```
$arr = array('name'=>'Tom','age'=>35,'phone'=>'123456');
$arr['name'] = 'Jerry';
echo $arr['name']; // 显示 Jerry
```

或者是：

```
$arr = array('a','b','c','d','e','f','g');
$arr[2] = '*'; // 将 c 修改为*
echo $arr[2]; // 显示*
```

现在的问题是：如果指定的键或索引值不存在会怎么样呢？如下面的代码。

```
$arr = array('name'=>'Tom','age'=>35,'phone'=>'123456');
$arr['organization'] = 'XYZ';
echo $arr['organization']; // 显示 XYZ
```

代码中，当我们使用键指定数组中不存在的成员时，会在数组中自动添加由新键和值组成的成员。通过索引值设置数组成员时，也是这样，如下面的代码。

```
$arr = array('a','b','c');
$arr[5] = '*';
echo $arr[5]; // 显示 *
```

接下来，我们做个小测试，将成员从 1 到 10 的数组成员都添加前缀“No.”，可以使用几种方法。

方法一，如下面的代码：

```
$arr = range(1,10);
for($i=0; $i<10; $i++)
{
    $arr[$i] = 'No.'.$arr[$i];
}
print_r($arr);
```

也许你还会使用 foreach 语句的方法来实现，如下面的代码。

```
$arr = range(1,10);
foreach($arr as $value)
{
    $value = 'No.'.$value;
}
print_r($arr);
```

我们可以看到，在访问数组成员时使用 foreach 语句的确比 for 语句清晰很多，但你也会发现，这段代码并没有修改$arr 数组成员，原因就在于我们使用$value 变量修改数组成员的值时，实际上是在修改这些成员的副本，如果我们需要在循环结构中修改数组成员的值，就必须在 foreach 语句中的$value 前使用&符号，如下面的代码。

```
foreach($arr as &$value)
```

这样一来，在 foreach 结构内的对&value 的修改，实际上就是在修改$arr 数组成员的值。

此外，在 PHP 中并没有删除指定数组成员的函数，但是，我们可以使用 unset()语句结构来完成这项工作，如：

```
$arr = range(1, 10);
unset($arr[5]);
```

2.8.3. 数值索引的真相

在 PHP 中使用数组的一个事实，也是非常重要的一点就是：数组的数值索引实际上是以数值作为键索引的一种形式。

有时候，我们可能想判断数组的第一个成员是否存在，也许你会使用下面的代码。

```
$arr = array("a"=>"aa", "b"=>"bb");
var_dump(isset($arr[0])); // bool(false)
```

什么情况？这是因为，在$arr 数组中并没有以“0”为键的成员，所以，代码会显示 bool(false)。

2.8.4. 多维数组

在 PHP 中，多维数组更像是关于数组的数组，如下面的代码，它模拟了一个二维的数组结构。

```
$group = array(
    array('name'=>'Tom','age'=>29),
    array('name'=>'Jerry','age'=>33),
    array('name'=>'John','age'=>35)
);
echo $group[1]['name']; //  显示 Jerry
```

我们定义的数组变量$group，它的每一个成员也都是一个数组结构，包括 name 和 age 两个数据项，这样，我们就可以通过[][]结构来访问所需要的数据值。

2.8.5. 数组函数

以下介绍一些常用的数组处理函数。

count()函数

count()函数用于获取数组的成员数量，在前面我们已经使用过，与其功能相同的还有 sizeof()函数。

array_key_exists()函数

array_key_exists()函数用于检查数组的成员中是否存在指定的键。如下面的代码。

```
$arr = array('name'=>'Tom','age'=>35);
```

```
var_dump(array_key_exists('name', $arr));  // 显示 bool(true)
var_dump(array_key_exists('phone', $arr));  // 显示 bool(false)
```

此外，array_key_exists()函数对于数值索引同样适用。

array_keys()函数

array_keys()函数用于返回数组中的键组成的新数组，新数组以数值为索引，以原数组键名称为成员数据。array_keys()函数包括三个参数：

- 参数一，指定原有的数组。
- 参数二，可选，指定搜索的键名，如果指定此参数，则在原数组中存在它们时返回指定键名的数组；如果不指定这个参数，则返回原数组中所有键名。
- 参数三，可选，指定在搜索是否按全等于运算（===），默认为 false。

如：

```
$arr1 = array("a"=>"aa", "b"=>"bb", "c"=>"cc");
$arr2 = array_keys($arr1);
// $arr2 等于 array("0"=>"a", "1"=>"b", "2"=>"c");
```

array_splice()函数

用于删除、替换数组的一部分，它定义了四个参数，其中：

- 参数一，需要操作的数组，定义为按引用传递。
- 参数二，开始操作的数组成员位置，使用数值索引，但请注意，这个索引并不是成员的键，而是真正的成员位置索引（由 0 开始）。
- 参数三，可选。指定操作从参数二位置开始的数组成员数量，默认为 0。
- 参数四，可选。指定替换或插入指定位置的元素。

此外，array_splice()函数的返回值将是从原数组中移除的元素组成的数组，如果没有移除内容，而返回空数组。

如下面的代码。

```
$arr1 = array("a","b","c","d","e");
$arr2 = array_splice($arr1,2,1);
print_r($arr1); // a,b,d,e
$arr3 = array_splice($arr1,2,2,"*"); //  将 d,e 替换为*
print_r($arr1); // a,b,*
```

array_values()函数

array_values()函数用于返回指定数组中的所有的值组成的新数组，并且使用数值索引。如：

```
$arr1 = array("a"=>"aa", "b"=>"bb", "c"=>"cc");
$arr2 = array_values($arr1);
// $arr2 等于 array("0"=>"aa", "1"=>"bb", "2"=>"cc");
```

in_array()函数

判断一个值在数据成员中是否存在。如：

```
$arr = array("a", "b", "c");
var_dump(in_array("a", $arr)); // bool(true)
```

sort()与 rsort()函数

sort()函数用于对数组按成员的值排序，如下面的代码：

```
$arr1 = array(9,3,8,5,1,6);
sort($arr1);
print_r($arr1);
```

代码显示如下：

```
Array ( [0] => 1 [1] => 3 [2] => 5 [3] => 6 [4] => 8 [5] => 9 )
```

我们可以看到，调用 sort()函数后，数组的成员已经按升序进行了重新排列；与 sort()相对应的是 rsort()函数，它的功能是将数组成员按照降序进行排列。

在 sort()函数中，我们还可以使用第二个参数，其功能是指定数组成员在排列时的比较方法，其值由一些常量指定，包括：

- SORT_REGULAR 值，这也是默认值，成员将按原始类型进行比较。
- SORT_NUMERIC 值，数组成员作为数字进行比较。
- SORT_STRING 值，数组成员作为字符串进行比较。
- SORT_LOCALE_STRING 值，根据当前的区域（locale）设置，将数组成员作为字符串进行比较。区域设置可以使用 setlocale()函数。
- SORT_NATURAL 值，数组成员按自然顺序，并作为字符串进行排序。
- SORT_FLAG_CASE 值，数组成员按自然顺序，并作为字符串进行排序，而且不区分字母大小写。

asort()与 arsort()函数

asort()函数与 sort()功能相同，也是按照数组成员的值进行升序排列，而 arsort() 函数，则是按照数组成员值进行降序排列。

ksort()与 krsort()函数

ksort()函数按照数组成员的键进行升序排列，而 krsort()函数则是按照数组成员的键进行降序排列。

shuffle()函数

shuffle()函数用于对数组成员进行随机排序，比如在扑克类游戏中，你可以通过如下的代码非常方便地完成洗牌操作。

```
$cards = range(1,54);
shuffle($cards);
```

array_reverse()函数

array_reverse()函数用于对数组成员进行反向排序，并返回排序后的新数组，如：

```
$arr = range(1,5); // 1,2,3,4,5
$arr_rev = array_reverse($arr);
print_r($arr_rev); // 5,4,3,2,1
```

array_push()与 array_pop()函数

array_push()函数用于将一个成员添加到数组的末尾，而 array_pop()函数同是从数组的删除数组末尾的成员，并返回这个成员的数据。如：

```
$arr = range(1,5); // 1,2,3,4,5
array_push($arr, 6); // 1,2,3,4,5,6
$num = array_pop($arr);
echo $num; //  显示 6
```

array_count_values()函数

array_count_values()函数是一个快捷的数组成员统计工具，它可以方便地计算出一个数组中每一种数据出现了多少次，并返回统计结果；返回结果同样也是一个数组，其中，原数组中的成员数据作为新数组成员的键，其出现次数作为新数组成员的值。如下面的代码。

```
$arr = array(1,2,1,3,1,2,1,2,3,4);
$result = array_count_values($arr);
print_r($result);
```

其结果如下：

```
Array ( [1] => 4 [2] => 3 [3] => 2 [4] => 1 )
```

即在$arr 数组中，1 出现了 4 次，2 出现了 3 次，3 出现了 2 次，而 4 出现了 1 次。

extract()函数

extract()函数用于将数组分解成相对应的一系列变量，其中，成员的键作为变量名，成员的值作为变量的值。如下面的代码。

```
$arr = array('name'=>'Tom','age'=>35,'phone'=>'123456');
extract($arr);
echo $name;   //显示 Tom
echo $age;   //显示 35
echo $phone;   //  显示 123456
```

在我们处理表单数据或页面参数时，extract() 函数会非常有用，我们可以将$_GET、$_POST 或$_REQUEST 数组中用户提交的数据一次性转换为相应的变量，这样就可以更方便地引用这些数据了。

如我们在 index.php 中编写以下代码。

```
<?php
extract($_GET);
```

```
print "name=$name".'<br>';
print "language=$language";
?>
```

然后，我们通过“http://localhost:8080/index.php?name=Tom&language=zh-cn”来访问这个页面，我们得到如下图的结果。

如果数组中有相同键的成员，则会使用最后的成员信息创建变量，这是 extract()函数的默认处理方式，不过，extract()函数中还可以指定第二和第三个参数来改变这一规则；一般情况下，我们还是建议使用 extract()函数的默认规则来处理同名键的问题，这就需要对用户提交的数据进行更加严格的控制，避免同名键数据的出现。不过，在一些特殊情况下，可能还是需要使用 extract()函数的更多参数，在日期和时间处理的讨论中，我们可以看到相关的应用。

array_walk()函数

array_walk()函数用于将数组成员进行相同的操作，而这些工作是由另外一个函数来完成的，如下面的例子，我们将实现给每个数组成员添加前缀的工作。

```
function addPrefix(&$value, $key, $prefix)
{
    $value = $prefix.$value;
}
$arr = range(1,5);
array_walk($arr, 'addPrefix', 'No.');
print_r($arr);
```

代码运行结果如下：

```
Array (
[0] => No.1
[1] => No.2
[2] => No.3
[3] => No.4
[4] => No.5
)
```

首先，我们定义了 addPrefix()函数，它包括三个参数：

- 参数一，用于表示数组成员值的参数，请注意，需要使用&符号定义为引用参数，只有这样才能在函数中正确的修改数组成员的值。
- 参数二，表示数组成员键名称的参数。
- 参数三，用于指定操作相关数据的参数。

然后，通过调用 array_walk()函数，我们将一个 1 到 5 的整数数组的每个成员都添加“No.”前缀。在 array_walk()过程中同样使用了三个参数：

- 参数一，需要处理的数组变量。
- 参数二，字符串形式的函数名称。
- 参数三，可选，指定用于操作的相关附加数据；这个参数的指定应与被调用函数中第三个参数的定义相匹配，如例子中的 addPrefix()函数。下面的代码，我们将完成与前面代码相同的工作，而这一次，array_walk()函数和 addPrefix()参数都没有使用第三个参数。

```
function addPrefix(&$value, $key)
{
    $value = "No.".$value;
}
$arr = range(1,5);
array_walk($arr, 'addPrefix');
print_r($arr);
```

2.9. 日期与时间

日期与时间是一种比较复杂的组合型数据，各种环境下的处理方式并不完全一致；本节，我们就了解一些在 PHP 中处理日期与时间的函数，并对一些常用代码进行封装。

2.9.1. time()、mktime()与 checkdate()函数

在 PHP 中处理日期和时间，有一个很重要的概念就是 UNIX 时间戳（timestamp），这是一个整数，这个数据的基点就是 1970 年 1 月 1 日零时零分零秒的时刻，其值为 0，而时间戳数据实际上就是距离这一基准时间的秒数。

我们可以使用 time()函数获取系统中的当前时间戳，如下面的代码。

```
echo time();
```

除了使用 time()函数获取当前时间，我们还可以使用 mktime()函数创建一个时刻的时间戳，此函数的定义如下：

```
int mktime(<时>, <分>, <秒>, <月>, <日>, <年>, <是否夏令时>)
```

这几个参数都整数类型，并且都是可选的，如果这些参数都不指定，那么，mktime()函数返回的值与 time()函数的作用就是相同的。

使用 mktime()函数时应注意，如果环境中设置的时区不确定，会在页面中显示一个警告信息；在实际应用中，我们可以使用 date_default_timezone_set()函数设置默认的时区；在大陆地区，可以设置的时区参数有"Asia/Shanghai"和"Asia/Chongqing"，即上海和重庆。下面的代码，将分别显示当前时间戳和一个指定时刻的时间戳。

```
date_default_timezone_set('Asia/Shanghai');
echo mktime();
echo '<br>';
echo mktime(8,15,56,6,26,2011);
```

此外，如果只需要年、月、日，可以将 mktime()函数的前三个参数设置为 0，而年、月、日数据是否能够正确组成时间戳数据，还可以使用 checkdate()函数进行验证，它的三个参数都是整数，定义格式如下：

```
bool checkdate(<月>, <日>, <年>)
```

如下面的代码。

```
date_default_timezone_set('Asia/Shanghai');
var_dump(checkdate(2,30,2015));  // 显示 bool(false)，2 月没有 30 号
```

2.9.2. date()函数

如果我们只能使用整数来处理日期和时间的值，当然不太直观，此时，我们可以使用 date()函数将日期和时间信息转换为特定内容和格式的字符串，如下面的代码，将显示系统当前时间的标准日期时间字符串。

```
date_default_timezone_set('Asia/Shanghai');
print date('c');
```

这种格式包含了完整的日期与时间信息，并包含了“T”字母和时区信息，也许你还是觉得有些太复杂了，因为在实际应用中，我们可能只需要如下格式的日期和时间信息：

```
年-月-日 时:分:秒
```

此时，我们可以使用一些格式化字符来完成这项工具，如下面的代码。

```
date_default_timezone_set('Asia/Shanghai');
print date('Y-m-d H:i:s');
```

代码中的 date()函数将显示当前系统中的日期和时间，如果我们需要给出指定时刻的信息，可以在 date()函数的第二个参数中设置相应的时间戳。请注意代码中的格式化字符，它们是严格区分大小写的，本例中，将始终返回 19 个字符的字符串，包括 4 位年份，月、日、时、分、秒都为 2 位，另外还有 2 个连接符、2 个冒号和一个空格。

在 date()函数的第一个参数中，我们还可以使用很多的格式化字符，其中还有一些比较实用的，如大写字母 L，它会返回是否为闰年的信息，我们可以使用如下的代码。

```
date_default_timezone_set('Asia/Shanghai');
$isLeapYear = (bool)date('L');
if ($isLeapYear)
    echo '今年是闰年';
else
    echo '今年不是闰年';
```

使用大写字母 Z 作为格式化字符时，会返回当前时区与格林尼治（Greenwich）标准时间相差的秒数，如下面的代码。

```
date_default_timezone_set('Asia/Shanghai');
echo date('Z'); // 显示 28800，正八区，8*60*60
```

在 date()函数中，还有一个比较实用的格式化字符是大写字母 W，它可以给出当前日期是当年的第几周，如：

```
date_default_timezone_set('Asia/Shanghai');
echo date('W'); //
```

此外，小写的 w 返回的是星期几，0 是星期日，1 到 6 分别是星期一到星期六。

完成的格式化字符列表可以参考 php.net 网站，搜索 date 即可。

2.9.3. getdate()函数

getdate()函数可以一次性给出时间戳中的所有主要日期与时间信息，并以数组的形式返回，数组成员（日期或时间数据项）的索引（键名称）如下：

- 索引值为 0 的成员，保存时间戳数据。
- year，年份数据。
- mon，月份的数值数据。
- mday，当月中的第几天。
- hours，小时数据。
- minutes，分钟数据。
- seconds，秒钟数据。
- wday，一周中的第几天，0 为周日，1 到 6 分别是周一到周六。
- yday，一年中的第几天。
- month，月份的名称。
- weekday，星期的名称。

我们根据相应的索引或键可以很方便的获取相应的日期和时间数据，如果你想根据这些数据快速地创建变量，可以使用 extract()函数，如下面的代码。

```
date_default_timezone_set('Asia/Shanghai');
extract(getdate(), EXTR_PREFIX_ALL,'dt');
echo $dt_0;
```

代码会显示当前时间的时间戳。请注意 extract()函数的使用，我们对每一个数据成员名称都添加了 dt 前缀，如果不这样做，索引 0 的数据项不能正确创建变量，因为变量名不能以数字开始；通过添加前缀，我们可以将所有的数组成员转化为对应的变量，如$dt_year 变量保存年份数据、$dt_0 变量保存时间戳等。

2.9.4. microtime()函数

我们知道，使用 time()函数返回的时间戳只精确到秒，如果我们需要更小的时间单位数据，如微秒，则可以使用 microtime()函数。此函数有一个可选参数，其使用规则如下：

microtime()或 microtime(false)

这两种使用方法产生的结果是相同的。函数会返回一个字符串，包括两个使用空格分隔的部分，前一部分为微秒信息（小数形式），后一部分为秒数（整数形式），我们可以使用如下代码将它们分离后分别使用。

```
date_default_timezone_set('Asia/Shanghai');
$mtstr = microtime();
$arr = explode(' ',$mtstr);
$microsecond = (double)$arr[0];
$second = (int)$arr[1];
echo $second,$microsecond;
```

这种调用方式返回的小数精度很高，我们可以根据需要对其进行加工使用。

microtime(true)

这种调用形式将直接返回一个浮点数，包括秒钟数据（整数部分）和微秒数据（小数部分），如下面的代码。

```
date_default_timezone_set('Asia/Shanghai');
echo microtime(true);
```

这种调用方式只保留四位小数，一般情况下，这也够用了。

2.9.5. 封装日期时间处理代码

前面，我们了解了一些常用的日期和时间处理的函数，我们可以看到，在实际开发中的使用并不是十分方便，特别是在只对中国的日期和时间处理，所以，我们会考虑对这些功能进行一定的封装，在处理日期时间数据的时候使用更加便利。

在 PHP 中的代码封装，常用的方法包括自定义函数和类，而对于简单的功能，定义一些函数就已经足够了，下面就是一些简单的功能封装，这些代码们于/lib/cn.php 文件中，大家可以根据需求添加新的函数。稍后讨论如何在 PHP 文件中引用这些函数。

cnSetTimezone()函数

cnSetTimezone()函数用于设置中国的时区。其定义如下：

```
// 设置时区
function cnSetTimezone()
{
    date_default_timezone_set('Asia/Shanghai');
}
```

cnGetLongDateString()函数

cnGetLongDateString()给出指定时间戳的日期长格式字符串。其定义如下：

```
// 给出中国日期长格式字符串
function cnGetLongDateString($ts=null)
{
    cnSetTimezone();
    if (!is_int($ts)) $ts = time();
    return date('Y 年 m 月 d 日',$ts);
}
```

其中，参数为可选，如果不使用参数，则返回系统当前日期的长格式字符串。

cnGetShortDateString()函数

cnGetShortDateString()函数给出指定时间戳的日期短格式字符串。其定义如下。

```
// 给出中国日期短格式字符串
function cnGetShortDateString($ts=null)
{
    cnSetTimezone();
    if (!is_int($ts)) $ts = time();
    return date('Y-m-d',$ts);
}
```

参数与 cnGetLongDateString()函数一样为可选参数，如果不使用参数，则返回系统当前日期的短格式字符串。

isLeapYear()函数

isLeapYear()函数用于判断指定的时间戳是否为闰年。其定义如下：

```
// 判断是否为闰年
function isLeapYear($ts=null)
{
    cnSetTimezone();
    if (!is_int($ts)) $ts = time();
    return (bool)date('L',$ts);
}
```

getDateTimeString()函数

getDateTimeString()函数将会给出指定时间戳的“年-月-日 时:分:秒”格式字符串。其定义如下：

```
// 给出标准的日期时间串
function getDateTimeString($ts=NULL)
{
    cnSetTimezone();
    if (!is_int($ts)) $ts = time();
    return date('Y-m-d H:i:s', $ts);
```

```
}
```

cnGetWeekName()函数

cnGetWeekName()函数给出指定时间戳中的中文星期名称，如星期日、星期一等。其定义如下：

```
// 给出中文星期名称
function cnGetWeekName($ts=null)
{
    cnSetTimezone();
    if (!is_int($ts)) $ts = time();
    $week = intval(date('w', $ts));
    return array("星期日","星期一","星期二",
        "星期三","星期四","星期五","星期六")[$week];
}
```

如果查看源文件，还可以看到一个 CCn.php 文件，这是以上封装函数的面向对象版本，在后面的讨论中，我们可以了解如何使用它们。

2.10. 引用外部文件

前面，我们封装了一些关于日期和时间相关操作的函数，那么，在其它的 PHP 文件中如何使用它们呢？

在 PHP 中，引用外文件的功能主要有四个语句，即 include、require、include_once 和 require_once。它们都可以通过绝对路径、相对路径和 URL 来导入外部的 PHP 文件，而它们的不同点在于：

- include 语句每次调用都会引用一次文件，这就可能造成重复引用的问题，此时，可以考虑使用 include_once 语句，此语句可以保证在一个页面中只会引用一次外部文件，而不会造成重复引用。
- require 和 require_once 语句的功能与 include 和 include_once 功能相似，只是在 include 或 include_once 语句导入文件时，如果发生错误，PHP 会继续执行错误后的代码，而 require 和 require_once 语句遇到错误时，会终止代码的执行。

综合考虑，我们会在本书的代码中使用 require_once 语句，即保证代码文件只引用一次的同时，一旦出现错误就停止代码的运行；这就要求我们需要努力创建高质量的代码，以及高效的代码组织策略，我想，这应该是一个好习惯。

前面，我们说过，引用外部 PHP 文件，可以使用绝对路径、相对路径或 URL；本书中，我们将使用绝对路径，即从 PHP 网站根目录开始的绝对路径。如前面的/lib/cn.php 文件，我们就可以在 index.php 文件中使用如下代码来引用。

```
<?php
```

```
require_once $_SERVER['DOCUMENT_ROOT'].'/lib/cn.php';
?>
```

这是从当前网站根目录开始的绝对路径，在网站中，我们使用这种方法来引用文件，可以很直观，也会很安全。

经过引用/lib/cn.php 文件，我们就可以在 index.php 文件中使用 cn.php 文件中封装的函数了。如下面的代码。

```
<?php
require_once $_SERVER['DOCUMENT_ROOT'].'/lib/cn.php';

cnSetTimezone();
$ts = mktime(0,0,0,7,10,2015);
echo cnGetLongDateString($ts);
echo '<br>';
echo cnGetWeekName($ts);

?>
```

代码运行结果如下图。