# 第 3 章　流程控制语句

任何 PHP 脚本都是由一系列语句构成的。一条语句可以是一个赋值语句、一个函数调用、一个循环、一个条件语句或者甚至是一个什么也不做的语句(空语句)。语句通常以分号结束。此外,还可以用花括号将一组语句封装成一个语句组。语句组本身可以当作是一行语句。

## 3.1　条件语句与运算符

### 3.1.1　if 条件语句

条件语句像变量一样是程序设计的一个组成部分,大多数人都熟悉它们的某一种或另外一种形式。动态 Web 页面经常需要使用条件语句,依据设置的准则来改变脚本的行为。

PHP 用于创建条件语句的 3 个主要术语是:if、else 和 elseif(它也可以写作两个单词:else if)。

在 PHP 中,条件语句有三种形式,每个条件语句都包含有一个 if 子句。

**1. if 结构**

```
if(条件){
  语句
}
```

首先判断“条件”,如果条件为真(true),则执行“语句”;如果条件为假(False),将忽略“语句”。

**2. if-else 结构**

条件语句的第二种形式是 if-else,除了 if 语句之外,还加上了 else 语句,它可以在 if 语句中的表达式的值为 False 时执行。

```
if(条件){
  语句 1
}else{
  语句 2
}
```

首先判断“条件”，如果条件为真(true)，则执行“语句 1”；如果条件为假(False)，则执行“语句 2”。

**3. if-elseif 结构**

条件语句的第三种形式是 if-elseif，elseif 是 if 和 else 的组合。和 else 一样，它延伸了 if 语句，可以在原来的 if 表达式值为 False 时执行不同语句。但是和 else 不一样的是，它仅在 elseif 的条件表达式值为 True 时执行语句，语法如下：

```
if(条件 1){
  语句 1
}elseif(条件 2){
  语句 2
}elseif(条件 3){
  …
}else{
  语句 n
}
```

如果条件为真，则将执行其下面的花括号({})中的代码。如果条件不为真，PHP 将继续往下执行。首先判断“条件 1”，如果“条件 1”为真(true)，则执行“语句 1”；如果“条件 1”为假(False)，则判断“条件 2”。如果“条件 2”为真(true)，则执行“语句 2”；如果“条件 2”为假(False)，则判断“条件 3”，以此类推，如果所有的表达式的值都为 False，则执行“语句 n”。

这个过程将继承——可以根据需要使用多个 elseif 子句——直至 PHP 遇到 else，此时将自动执行它；或者如果没有 else，则 PHP 将执行到条件语句终止为止。

因此，总是把 else 放在最后，并将其视作默认的动作，除非满足特定的条件，这样做是重要的。

PHP 中条件为真的原因有许多种。下面是常见的条件为真的情况：

- $var，如果变量 $var 非空，也就是具有非 0 值、空字符串或 NULL，则条件为真。
- isset($var)，如果变量 $var 具有不同于 NULL 的任何值，包括 0 或空字符串，则条件为真。
- TRUE、true、True 等。

在上述为真条件中，引入了一个函数 isset()。这个函数用于检查一个变量是否被设置，这意味着它具有一个不同于 NULL 的值。还可以连同符号一起使用比较和逻辑运算符(参见表 3-1 和表 3-2)，来建立更复杂的表达式。

在编写条件语句时，经常会使用如表 3-1 所示的比较和逻辑运算符。

比较运算符，允许对两个值进行比较，PHP 常用的比较运算符如表 3-1 所示。

如果比较一个整数和字符串，则字符串会被转换为整数。如果比较两个数字字符串，则作为整数比较。

PHP 常用的逻辑运算符如表 3-2 所示。

表 3-1　算术运算符

| 符　号 | 含　义 | 示　例 |
|---|---|---|
| == | 等于 | $a==$b |
| === | 全等,等于且类型相同 | $a===$b |
| != | 不等 | $a!=$b |
| !== | 不全等,不等或类型不同 | $a!==$b |
| > | 大于 | $a>$b |
| < | 小于 | $a<$b |
| >= | 大于或等于 | $a>=$b |
| <= | 小于或等于 | $a<=$b |

表 3-2　逻辑运算符

| 符　号 | 含　义 | 示　例 |
|---|---|---|
| ! | 逻辑非 | !$a |
| && | 逻辑与 | $a&&$b |
| \|\| | 逻辑或 | $a\|\|$b |
| XOR | 逻辑异或 | $a XOR $b |

“与”和“或”有两种不同形式运算符,它们运算的优先级不同,&& 和||优先级高。

**4. 使用条件语句**

(1) 在文本编辑器中创建一个新的 PHP 文档(参见脚本 3-1)。

**脚本 3-1**　代码中的条件语句允许依据不同的条件执行不同的行为。

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4  <head>
5  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
6    <title>条件语句</title>
7   </head>
8   <body>
9   <?php #脚本 3-1-if.php

10  //初始化 $gender 变量
11  $gender=男;
12
13  //根据变量的值输出欢迎内容
14  if(isset($gender)){
```

```
15      if($gender=='男'){
16          echo '<p><b>您好,先生!</b></p>';
17      }elseif($gender=='女'){
18          echo '<p><b>您好,女士!</b></p>';
19      }else {                    //$gender变量的值无效
20          echo '<p><b>请您重新输入您的性别。</b></p>';
21      }
22  }else{
23      echo '<p><b>您还未输入您的性别。</b></p>';
24  }
25
26  ?>
27  </body>
28  </html>
```

在 HTML 表单验证输入时,经常会使用单选按钮、复选框或选项菜单等控件,此时使用条件语句是一种简单、有效的方式。如果用户选中了任何一个性别单选按钮,那么传递到后台的变量将具有一个值,这时可以使用 isset()函数来判断条件是否为真。如果条件不为真,那么指示它没有值或无效。

(2) 在第一个条件语句(isset())中,嵌套了另一个条件语句,基于 $gender 的值打印一条消息。

```
if($gender=='男'){
    echo '<p><b>您好,先生!</b></p>';
}elseif($gender=='女'){
    echo '<p><b>您好,女士!</b></p>';
}else {
    echo '<p><b>请您重新输入您的性别。</b></p>';
}
```

这个 if-elseif-else 条件语句会查看 $gender 变量的值,并针对每一种可能性打印一条不同的消息。双等于号(==)指相等,而单个等于号(=)则用于赋值,记住这一点非常重要。这个区别很重要,因为条件 $gender==='男'可能为真,也可能不为真;但是 $gender='男'总是为真。

(3) 保存文件,上传到 Web 服务器,并在 Web 浏览器中测试它(参见图 3-1、图 3-2 和图 3-3)。

在此例中使用了条件语句的嵌套,把一个条件语句放在另一个条件语句内部。遇到这种复杂语句结构时,可以使用制表位(或 4 个空格)实现代码的缩进,以这种方式格式化条件语句是一个标准的过程和良好的编程形式。

这个脚本中的第一个条件语句(isset())是一个检查默认值的理想示例。在后续学习了 HTML 表单后,可使用 PHP 的表单变量来获取用户实际选择的性别($gender)变量的值。

另外,如果只执行一条语句,则用于指示条件语句开始和结束的花括号不是必需的。


图 3-1　基于性别的条件语句针对表单中的每个选择打印一条不同的消息


图 3-2　当更改表 $ gender 初始值时，相同的脚本将产生不同的结果(对比图 3-1)


图 3-3　如果没有选择性别(未初始化 $ gender)，就会打印一条消息，指出用户的疏忽

不过，出于对清晰性的考虑，在这里建议初学者使用它们。

### 3.1.2　switch 条件语句

PHP 还有另一种条件语句，称为 switch，能够最佳地用于代替较长的 if-elseif-else 条件语句。上一节介绍的 if 语句只有两个分支可供选择，而实际问题中常需要用到多分支

的选择，比如，成绩的等级分为优秀、良好、及格或不及格等。当然可以通过使用多个 if 语句，或嵌套的 if 语句来处理，但是如果分支较多，就会使程序冗长而且可读性较差。PHP 提供了分支(switch)语句来直接处理多分支选择。switch 的语法是：

```
switch($variable){
  case '值 1':
    语句 1
    break;
  case '值 2':
    语句 2
    break;
  default:
    其他语句
    break;
}
```

switch 语句是一行接一行地执行的，开始时没有代码被执行。switch 条件语句将 $variable 变量的值与不同的 case 作比较。仅当一个 case 语句中的值和 $variable 变量的值匹配时 PHP 才开始执行语句，直到 switch 的程序段结束或者遇到第一个 break 语句为止。如果不在 case 的语句段最后写上 break，PHP 将继续执行下一个 case 中的语句段。一个 case 的特例是 default。它匹配了任何与其他 case 都不匹配的情况，并且应该是最后一条 case 语句。如果没有发现匹配，则会执行 default，它是可选的。

因此，脚本 3-1 中的条件语句可以重写为：

```
switch($gender){
  case '男':
    echo '<p><b>您好,先生!</b></p>';
    break;
  case '女':
    echo '<p><b>您好,女士!</b></p>';
    break;
  default:
    echo '<p><b>请您重新输入您的性别。</b></p>';
    break;
}
```

需要注意的是，switch 条件语句仅限于在可以检查变量值与某些情况的相等性的条件下使用，往往不能轻松地检查更复杂的条件语句。

## 3.2　循环结构

下面将讨论的语言构造是循环，有两种常用的循环类型：while 和 for。

### 3.2.1　while 循环

**1. while 循环**

while 循环是 PHP 中最简单的循环类型。while 语句的基本格式是：

```
while(条件)
    语句
```

只要判断条件为 TRUE，就重复执行嵌套中的循环语句。表达式的值在每次开始循环时检查，所以即使这个值在循环语句中改变了，语句也不会停止执行，直到本次循环结束。如果表达式的值一开始就是 FALSE，则循环语句一次都不会执行。后面在第 9 章中将介绍，当从数据库中检索结果时，经常使用 while 循环。

和 if 语句一样，可以在 while 循环中用花括号括起一个语句组。例如：

```
$i=1;
while($i <=10){
  echo $i++;                    //从 1~10 依次输出
}
```

**2. do-while 循环**

do-while 和 while 循环非常相似，区别在于表达式的值是在每次循环结束时检查而不是开始时。do-while 语句的基本格式是：

```
do{
  语句
}while(条件)
```

和 while 循环主要的区别是 do-while 的循环语句保证会执行一次，因为 do-while 循环的表达式是否为真在每次循环结束后检查，然而在 while 循环中就不一定了，因为 while 循环的条件是否为真在循环开始时检查，如果一开始就为 False，则整个循环立即终止。例如：

```
$i=0;
do {
  echo $i;
} while($i >0);
```

以上循环将正好运行一次，因为经过第一次循环后，当检查表达式是否为真时，其值为 False( $i 不大于 0)而导致循环终止。

### 3. 使用 while 和 do-while 循环

(1) 在文本编辑器中创建一个新的 PHP 文档(参见脚本 3-2)。

**脚本 3-2** 使用 while 和 do-while 循环。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
  <title>while 和 do-while 循环</title>
 </head>
 <body>

 <body>
 <?php
 $num=0;
 while($num<10){
     echo "第".$num."次循环。<br />";
     $num++;
 }
 ?>
 <br />
 <?php
 $i=0;
 do{
     echo "第".$i."次循环。<br />";
     $i++;
 }while($i<10);
 ?>
 </body>
 </html>
```

(2) 使用 while 循环实现 10 次循环。

```
$num=0;
while($num<10){
    echo "第".$num."次循环。<br />";
    $num++;
 }
```

(3) 使用 do-while 循环实现 10 循环。

```
$i=0;
do{
    echo "第".$i."次循环<br />";
    $i++;
}while($i<10);
```

两次循环设定循环变量分别是 $num 和 $i，两次循环的显示效果是一致的，都是从 0 开始循环至 9，共 10 次。

(4) 保存文件，上传到 Web 服务器，并在 Web 浏览器中测试它(参见图 3-4)。

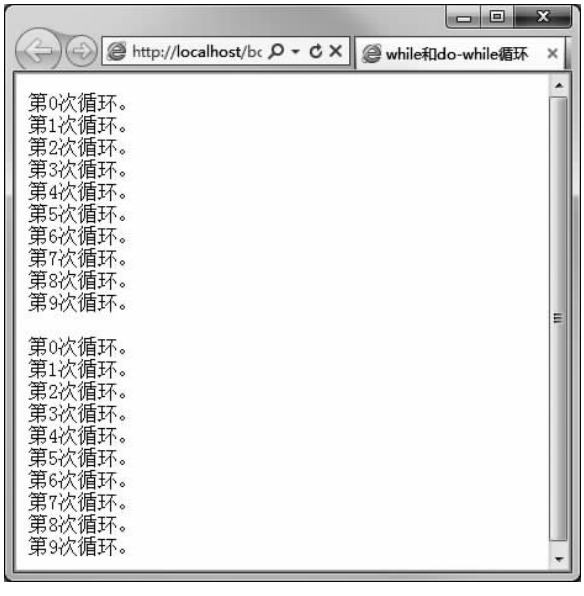



图 3-4　while 和 do-while 循环

### 3.2.2　for 循环

#### 1. for 循环

for 循环具有更复杂的语法：

```
for(expr1; expr2; expr3){
    statement
}
```

在第一次执行循环时，会运行初始表达式(expr1)，然后检查条件(expr2)，如果条件为真，就执行循环的内容(statement)。执行之后，将会运行结束表达式(expr3)，并再次检查条件(expr2)。这个过程会继续下去，直到条件为假。例如：

```
for($i=1;$i<=10;$i++){
    echo $i;
}
```

第一次运行这个循环时，将把 $i 变量的初始值设置为 1，然后检查条件(1 小于或等于 10 吗?)，由于这个条件为真，就会打印出 1(echo $i)；然后将 $i 递增为 2($i++)，再检查条件，以此类推。这个脚本的执行结果将打印出数字 1～10。

在这里需要注意的是，由于 for 和 while 循环的语法和功能足够相似，因此经常可以互换使用它们。尽管如此，经验表明：当要把某件事情做一定的次数时，for 循环是更好

的选择;只要条件为真就做某件事情时,最好使用 while 循环。

使用循环时,要注意观察参数和条件,避免可怕的无限循环,当循环条件永远不会为假时,就会发生这种情况。

**2. 使用 for 循环**

(1) 在文本编辑器中创建一个新的 PHP 文档(参见脚本 3-3)。

**脚本 3-3** 使用 for 循环。

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4  <head>
5  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
6    <title>for 循环</title>
7   </head>
8   <body>
9
10  <body>
11  <?php
12  for($i=0;$i<10;$i++){
13      echo "第".$i."次循环。</br>";
14  }
15  ?>
16  </body>
17  </html>
```

(2) 使用 for 循环实现 10 次循环。

```
for($i=0;$i<10;$i++){
    echo "第".$i."次循环。</br>";
}
```

循环变量为 $i,循环次数从 0～9,共 10 次。

(3) 保存文件,上传到 Web 服务器,并在 Web 浏览器中测试它(参见图 3-5)。


图 3-5 for 循环

## 3.3　项目训练——动态年月日下拉菜单

### 3.3.1　项目说明

网络上众多注册表单在涉及选择日期或生日等表单控件时，多数会使用下拉列表。通过 PHP 循环，实现下拉列表年月日自动生成为列表菜单项目。

### 3.3.2　设计思路

不管是年月日的哪一项，都是一定范围内的整数集合，年份可以是 2005—2015 年，月份规定为 1～12 月，日期为 1～31 日，这里暂且忽略月份之间天数的差异。

为此，同样还是先制作静态页面，制作一个包含 3 个下拉菜单格式的 Web 页，然后使用任何一种 PHP 的循环结构去实现有规律的整数范围，并且年份的周期也方便在日后随时修改。

### 3.3.3　设计过程

循环实现动态年月日下拉菜单，这里分别使用了 while 循环、do-while 循环及 for 循环。

(1) 在文本编辑器中创建一个新的 PHP 文档(参见脚本 3-4)。

**脚本 3-4**　使用循环实现下拉菜单。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
  <title>while 和 for 循环结构</title>
 </head>
 <body>

 <form action="" method="post">
 <?php #脚本 3-4-calendar.php
 //创建年份下拉菜单
 echo '<select name="year">';
 $year=2005;
 while($year <=2015){
     echo "<option value=\"$year\">$year</option>\n";
     $year++;
 }
 echo '</select>年 ';

```

```
21  //创建月份下拉菜单
22  echo '<select name="month">';
23  $month=1;
24  do{
25      echo "<option value=\"$month\">$month</option>\n";
26      $month++;
27  }while($month <=12);
28  echo '</select>月 ';
29
30  //创建日期下拉菜单
31  echo '<select name="day">';
32  for($day=1; $day <=31; $day++){
33      echo "<option value=\"$day\">$day</option>\n";
34  }
35  echo '</select>日 ';
36
37  ?>
38  </form>
39  </body>
40  </html>
```

这里需要注意的是,需要 HTML 表单标签创建下拉菜单。因为在何处以及何时嵌入 PHP 是无关紧要的,所以这里把表单标签放在 PHP 代码前面。

(2) 使用 while 循环生成年份下拉菜单。

```
echo '<select name="year">';
$year=2005;
while($year <=2015){
    echo "<option value=\"$year\">$year</option>\n";
    $year++;
}
echo '</select>年 ';
```

最初设置 $year 变量,只要它小于或等于 2015,就会执行循环。在循环内,将会运行 echo()语句,然后递增 $year 变量。

(3) 使用 do-while 循环生成月份下拉菜单。

```
echo '<select name="month">';
$month=1;
do{
    echo "<option value=\"$month\">$month</option>\n";
    $month++;
}while($month <=12);
echo '</select>月 ';
```

这里使用的 do-while 循环生成月份与 while 循环生成年份功能相似。

(4) 使用 for 循环生成日期下拉菜单。

```
echo '<select name="day">';
for($day=1; $day <=31; $day++){
    echo "<option value=\"$day\">$day</option>\n";
}
echo '</select>日 ';
```

这个标准的 for 循环首先将 $day 变量初始化为 1。它将继续运行循环,直到 $day 大于 31,并在每次迭代时,将 $day 递增 1。循环自身的内容(它将执行 31 次)是一条 echo()语句。

(5) 将文件另存为 calendar.php,上传到 Web 服务器,并在 Web 浏览器中测试它(参见图 3-6)。


图 3-6　这些下拉菜单是使用循环结构创建的

(6) 选择“查看”→“源文件”命令,或“查看”→“页面源文件”命令,在浏览器中查看源代码(参见图 3-7)。


图 3-7　大多数源代码都是由仅仅几行 PHP 代码生成的

## 本章小结

本章介绍了 PHP 语言的条件语句及其余的运算符，以及循环语言构造。这些语言基础将会为完成后续的复杂程序设计提高效率。

## 重点回顾

1. PHP 的比较和逻辑运算符。
2. 如何使用分支语句 if 结构与 switch 结构实现各种条件判断。
3. 合理使用循环结构实现高效率的编程。

## 本章实训

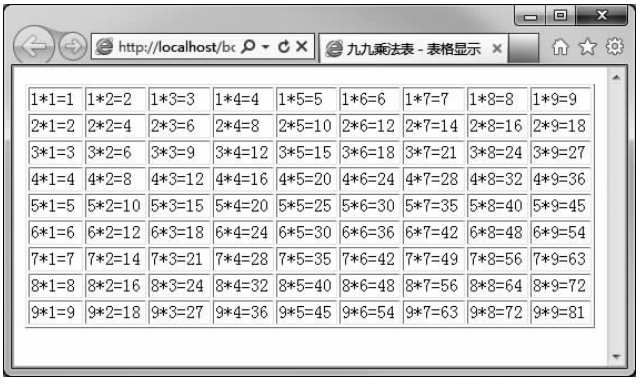
**【实训 1】**

新建 switch.php，参考 PHP 中文帮助，使用 date 函数获取今天星期几，然后使用 switch 分支语句分别给出不同说明。网页运行效果如图 3-8 所示。


图 3-8　switch.php 的运行效果

**【实训 2】**

新建 99.php，使用 for 循环实现九九乘法表，并将结果输出为表格格式。网页运行效果如图 3-9 所示。




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1*1=1 | 1*2=2 | 1*3=3 | 1*4=4 | 1*5=5 | 1*6=6 | 1*7=7 | 1*8=8 | 1*9=9 |
| 2*1=2 | 2*2=4 | 2*3=6 | 2*4=8 | 2*5=10 | 2*6=12 | 2*7=14 | 2*8=16 | 2*9=18 |
| 3*1=3 | 3*2=6 | 3*3=9 | 3*4=12 | 3*5=15 | 3*6=18 | 3*7=21 | 3*8=24 | 3*9=27 |
| 4*1=4 | 4*2=8 | 4*3=12 | 4*4=16 | 4*5=20 | 4*6=24 | 4*7=28 | 4*8=32 | 4*9=36 |
| 5*1=5 | 5*2=10 | 5*3=15 | 5*4=20 | 5*5=25 | 5*6=30 | 5*7=35 | 5*8=40 | 5*9=45 |
| 6*1=6 | 6*2=12 | 6*3=18 | 6*4=24 | 6*5=30 | 6*6=36 | 6*7=42 | 6*8=48 | 6*9=54 |
| 7*1=7 | 7*2=14 | 7*3=21 | 7*4=28 | 7*5=35 | 7*6=42 | 7*7=49 | 7*8=56 | 7*9=63 |
| 8*1=8 | 8*2=16 | 8*3=24 | 8*4=32 | 8*5=40 | 8*6=48 | 8*7=56 | 8*8=64 | 8*9=72 |
| 9*1=9 | 9*2=18 | 9*3=27 | 9*4=36 | 9*5=45 | 9*6=54 | 9*7=63 | 9*8=72 | 9*9=81 |

图 3-9　99.php 的运行效果

# 第 4 章　数　　组

数组是一种数据类型，它的使用频率相当高，学会处理数组会使我们在编程时得心应手。数组同数值或者字符串有着显著的不同，并且如果没有掌握和理解它，大多数 PHP 编程工作都将无法实现。

## 4.1　什么是数组

数组(array)组成一个复杂但是非常有用的概念。数值和字符串都是标量变量，它们只含有一个单独的值，数组实际上是一个数据集合，它可以保存多份单独的信息，就像是值的列表，其中每个值可以是字符串或数字，甚至是另一个数组。因此比起简单的字符串或者数字，数组能够让一个变量以指数级承载更多的信息。

### 4.1.1　索引数组与联合数组

数组就是一组数据的集合，把一系列数据组织起来，形成一个可操作的整体。数组的每个实体都包含两项：键和值。

数组可以构造成一系列键-值(key-value)对，其中每一对都是那个数组的一个项目或元素(element)。对于列表中的每个项目，都有一个与之关联的键(key)(或索引(index))。得到的结构很像是电子表格或数据库表(参见表 4-1 和表 4-2)。

**表 4-1　$ days 索引数组使用数字作为它的键**

| 键 | 值 | 键 | 值 |
|---|---|---|---|
| 0 | 星期天 | 3 | 星期三 |
| 1 | 星期一 | 4 | 星期四 |
| 2 | 星期二 | | |

**表 4-2　$ user 联合数组使用字符串作为它的键**

| 键 | 值 | 键 | 值 |
|---|---|---|---|
| id | 0922188 | gender | 男 |
| name | 张明 | age | 30 |
| pwd | 1234567890 | | |

PHP 支持两种数组：索引数组(indexed array)和联合数组(associative array)，前者使用数字作为键，后者使用字符串作为键。

同 C 语言和大部分语言一样，索引数组的第一个索引开始于 0，而不是从 1。键(key)可以是整型或者字符串，值(value)可以是任何类型。例如，$days 数组，如表 4-1 所示。

联合数组就是在索引值方面与索引数组不同，其索引值中包含了字符串，这样可以更方便地处理实际生活的问题，比如把名字作为索引，资料为索引的值，但是在操作数据时就需要比较复杂的方法。例如，$user 数组，如表 4-2 所示。

数组遵守与任何其他变量相同的命名规则，数组的名称使用美元符号($)开头。因此，单看变量名 $var 无法区分是数组，还是字符串或数字。

由于 PHP 对其变量结构的要求很宽松，数组甚至可以使用数字和字符串的组合作为它的键。唯一重要的规则是数组的每一个键都必须是唯一的。

### 4.1.2 创建数组

创建一个数组是指定义一个数组的名字和结构，可以初始化其内部数据元素的值，也可以不作初始化处理，即所有的元素值为空。在 PHP 中，创建或声明数据的方式主要有两种：一种是直接为数组元素赋值，一次添加一个元素来构建数组；另一种是应用 array()函数声明数组。

**1. 直接为数组元素赋值**

直接为数组元素赋值方法是使用方括号的语法，通过在方括号内指定键为数组赋值来实现，也可以省略键，在这种情况下给变量名加上一对空的方括号([])。语法如下：

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

其中，键 key 可以是整型或者字符串，值 value 可以是任何类型。例如：

```
$colName[0]="id";
$colName[1]="username";
$colName[2]="pwd";
```

如果给出方括号但没有指定键，则取当前最大整数索引值，新的键将是该值加 1。如果当前还没有整数索引，则键将为 0。例如：

```
$array[]='Math';
$array[]='9月 1日';
$array['teacher']='Chen';
```

需要注意的是，如果指定一个键，并且已经存在用那个相同的键进行索引的一个值，则新值将覆盖现有的值。例如：

```
$array['son']='Mark';
$array['son']='Michael';
$array[2]='apple';
```

```
$array[2]='orange';
```

如果在创建数组时不知所创建数组的大小，或在实际编写程序时数组的大小可能发生变化，那么采用这种数组创建的方法较好。

**2. 使用 array()函数创建数组**

创建数组正式的方法是使用 array()函数。它的语法如下：

```
$list=array('apple', 'banana', 'orange');
```

除非特别指出，否则数组的索引自动从 0 开始。本示例并未为元素指定特定的索引，因此 apple 作为第一项自动索引为 0，而第二项索引为 1，第三项索引为 2。

可以在使用 array()时为索引赋值，运算符“=>”用于定义数组元素的值。语法“key=>values”，用于定义数组键和对应的值。键可以是数字或字符串。例如：

```
$list=array(1=>'Monday', 2=>'Tuesday', 3=>'Wednesday');
$fruit=array("first"=>'apple',"second"=>'banana',"third"=>'orange');
```

最后，指定的索引值不必为数值，还可以使用字符串。这种索引的技术在制作更加有意义的列表时非常实用。

### 4.1.3 数组的打印

**1. 打印单个数组元素**

在 PHP 中对数组元素输出，可以通过 echo 和 print 语句来实现，但这只能对数组中的单个元素进行输出。要从数组中检索特定的值，需要先引用数组名称，然后在方括号中指出键。例如：

```
echo $days [2];                    //输出星期二
echo $user ['name'];               //输出张明
```

可以看到在 PHP 中，若数组的键值为数字(例如，2)则不会用引号括住，而对字符串(例如，name)则必须这样做。

由于数组使用的语法不同于其他变量，对它们执行打印更具技巧性，以下是容易犯的错误。

(1) 因为数组可以包含多个值，所以不能编写简单的代码：

```
echo "输出星期：$day";
```

(2) 联合数组中的键复杂化了打印它们的操作。下面的代码将引起一个解析错误：

```
echo "客户的姓名是$user['name']";
```

为了解决这个问题，当数组使用字符串作为它的键时，把数组名和键包装在花括号中：

```
echo "客户的姓名是{$user['name']}";
```

不过,数字式索引数组没有这种问题:

```
echo "今天是$days [4]";
```

**2. 输出整个数组结构**

要将数组结构输出则要使用 print_r()函数。语法如下:

```
print_r(mixed expression);
```

如果参数 expression 为普通的整型、字符型或实型变量,则输出该变量本身;如果该参数为数组,则按一定键值和元素的顺序显示出该数组中的所有元素。

作为示例,可以利用下面的脚本创建一个数组来记录一周中每天的天气,并且可以通过使用赋值运算符为每个元素赋值的方式直接向数组添加元素。该示例也将诠释如何能够以及不能打印数组。

(1) 在文本编辑器中创建一个新的文档(参见脚本 4-1):

**脚本 4-1** $weather 数组包含 3 个元素并且使用字符串作为它的键。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
  <title>打印数组</title>
 </head>
 <body>

 <?php //Script 4-1-weather1.php
 //创建数组:
 $weather=array(
   'Monday'=>'晴转多云',
   'Tuesday'=>'多云转阴',
   'Wednesday'=>'阴转阵雨'
   'Thursday'=>'多云'
 );

 //打印数组结构及内容:
 print_r($weather);

 //使用 print 打印数组:
 print "<p>$weather</p>";

 //为数组再添加 3 个元素:
 $weather['Friday']='多云转晴';
 $weather['Saturday']='多云转阴';
```

```
28 $weather['Sunday']='阴有阵雨';
29
30 var_dump($weather);
31 ?>
32 </body>
33 </html>
```

(2) 开始脚本中的 PHP 片段,使用 array()函数创建一个数组:

```
$weather=array(
    'Monday'=>'晴转多云',
    'Tuesday'=>'多云转阴',
    'Wednesday'=>'阴转阵雨',
    'Thursday'=>'多云'
);
```

这是在 PHP 中用字符串作为索引对数组进行初始化(创建和为其赋值)所使用的正确格式。由于其键和值都是字符串类型,因此需要用单引号将之引用。

不要在最后一个数组元素之后添加逗号(索引为 Thursday),这样做会导致解析错误发生。

(3) 尝试打印这个数组:

```
print_r($weather);
```

使用 print_r()函数可以来显示任何变量的内容和结构。

(4) 使用与前述不同的 print()函数打印数组:

```
print "<p>$weather</p>";
```

如图 4-1 所示,这里将不会显示数组内容,由于数组和其他变量类型是结构上不同的,因此直接使用“print”方法打印数组的请求会导致数值类型名称“Array”的出现(以及错误的出现,这取决于 PHP 的版本及其设置)。

(5) 为数组再添加 3 个元素:

```
$weather['Friday']='多云转晴';
$weather['Saturday']='多云转阴';
$weather['Sunday']='阴有阵雨';
```

这些代码又向已有的数组添加了 3 天的天气,分别用 Friday、Saturday 和 Sunday 进行索引。

(6) 再次打印数组。

这里用 var_dump()函数代替 print_r(),它不仅显示数组的内容,同时还会以更加详细的格式呈现数组的结构,如图 4-1 所示。在 PHP 第 6 版中,这个函数还会指出一个字符串是否使用了 Unicode 编码。

(7) 结束 PHP 和 HTML 代码部分,将文档保存为 weather1.php,放置在启用了 PHP 的服务器上适当的目录中,并且在 Web 浏览器中进行测试(参见图 4-1)。

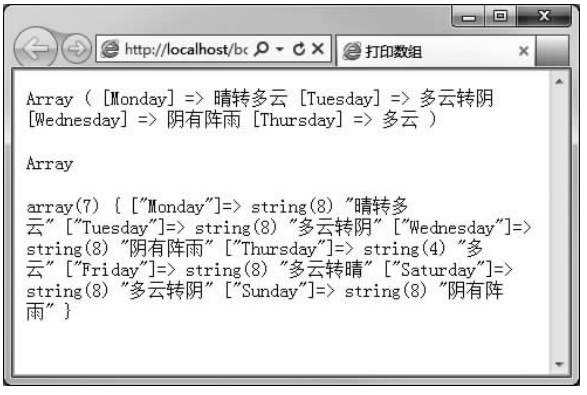



图 4-1　weather1.php 的执行结果

## 4.2　访问数组

### 4.2.1　foreach 循环

无论是如何创建数组的，从数组中检索某个特定元素（或者值）只有一个方法，那就是用它的索引。要访问数组中所有值的最快捷而且最简单的方法是使用 foreach 循环。这个循环结构将遍历数组中的每个元素。

要访问每个数组元素，可以使用 foreach 循环。foreach 循环有两种格式。如果只访问数组元素的值，可以使用：

```
foreach($array as $value){
  echo $value;                                    //$value 变量名字可以随意合法设置
}
```

foreach 循环将会迭代 $array 中的每个元素，并把每个元素的值赋予 $value 变量。

如果要访问键和值，可以使用：

```
foreach($array as $key=>$value){
  echo "元素的键为:$key ,对应的值为:$value.";  //$key 和$value 变量名字可以随意合法设置
}
```

现在可以利用这些知识编写一个新的 weather 脚本。不仅仅能够打印出数组中拥有的元素数量，还将能够访问到它真实的值。

**1. 打印数组的值**

（1）在文本编辑器中新建一个文档（参见脚本 4-2）。

**脚本 4-2**　foreach 循环是访问数组中每个元素的最简便的方法。

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```