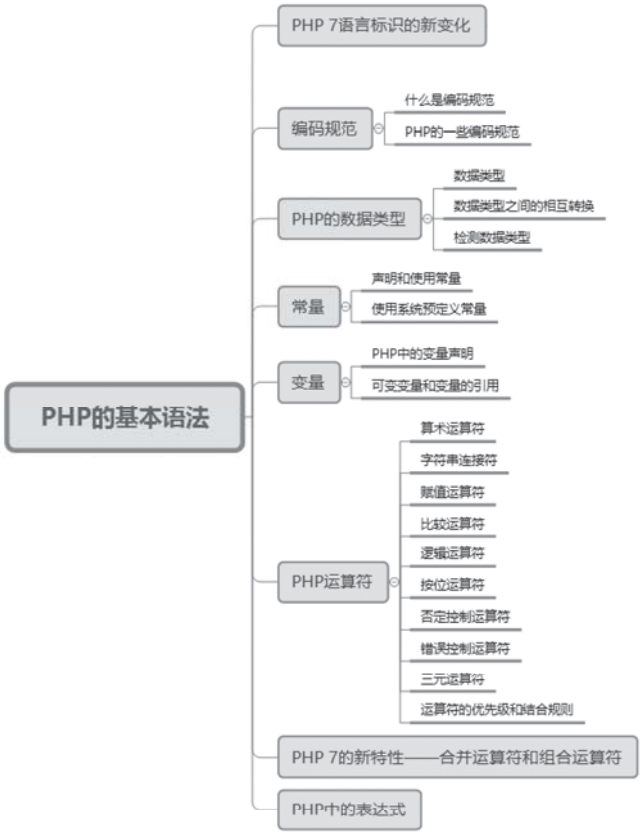
第2章　基本语法

本章导读

PHP 语言比较易学和易用，开发和运行的速度都比较快。不过要想精通 PHP 程序开发，基础语法知识还必须要深入掌握。本章将开始学习 PHP 的基本语法，主要包括 PHP 的标识符，编码规范、常量、变量、数据类型、运算符和表达式等。通过本章技能训练营的实战练习，可以让读者在掌握基础知识的前提下体验 PHP 程序开发的乐趣。

知识导图



2.1 PHP 7 语言标识的新变化

作为嵌入式脚本语言，PHP 是以“<?php”“ ?>”标识符作为开始标记和结束标记的。当服务器解析一个 PHP 文件时，通过寻找开始标记和结束标记，告诉 PHP 开始和停止解析其中的代码，凡是标记符号以外的内容都会被 PHP 解释器忽略。

PHP 7 版本支持标记风格如下：

```
<?php
    echo "这是PHP的标记风格。"
?>
```

为了让 PHP 程序的风格统一，目前 PHP 7.3 版本已经移除了另外两种标记风格，包括脚本风格和 ASP 风格。

1. 脚本风格

有的编辑器对 PHP 代码完全采用另外一种表示方式，比如 <script></script> 的表示方式。

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

2. ASP 风格

受 ASP 的影响，早期版本中 PHP 提供了 ASP 标志风格。例如：

```
<%
    echo "这是PHP的ASP的表示方式。";
%>
```

2.2 编码规范

由于现在的 Web 开发往往是多人一起合作完成的，所以使用相同的编码规范显得非常重要。特别是新的开发人员参与时，往往需要知道前面开发的代码中变量或函数的作用等，如果使用统一的编码规范，就容易多了。

2.2.1 什么是编码规范

编码规范规定了某种语言的一系列默认编程风格，用来增强这种语言的可读性、规范性和可维护性。编码规范主要包括语言下的文件组织、缩进、注释、声明、空格处理、命名规则等。

遵守 PHP 编码规范有下列好处。

（1）编码规范是团队开发中对每个成员的基本要求，对编码规范遵循得好坏是一个程序员成熟程度的表现。

（2）能够提高程序的可读性，利于开发人员互相交流。

（3）良好一致的编程风格在团队开发中可以起到事半功倍的效果。

（4）有助于程序的维护，可以降低软件成本。

2.2.2 PHP 的一些编码规范

PHP 作为高级语言的一种，十分强调编码规范。以下是规范在 3 个方面的体现。

1. 表述

比如在 PHP 的正常表述中，每一条 PHP 语句都是以“;”结尾，这个规范就告诉 PHP 要执行此语句。例如：

```
<?php
    echo "PHP以分号表示语句的结束和执行。";
?>
```

2. 注释

在 PHP 语言中，常见的注释包括以下几种风格。

（1）单行注释。

这是一种来源于 C++ 语法的注释模式，可以写在 PHP 语句的上方，也可以写在 PHP 语句的右侧。例如：

```
<?php
    //这是写在PHP语句上方的单行注释
    echo "这是单行注释的效果！";
    echo "这是单行注释的效果！";                    //这是写在PHP语句右侧的单行注释
?>
```

（2）多行注释。

这是一种来源于 C 语言语法的注释模式。例如：

```
<?php
    /*这是
    C语言风格
    的注释内容
    */
    echo "这是多行注释的效果！";
?>
```

注意：注释不能嵌套，因为 PHP 不进行块注释的嵌套检查，所以以下写法是错误的：

```
/*这是
    echo "这里开始嵌套注释";/*嵌套注释时PHP会报错*/
*/
```

（3）# 号风格的注释。

这种方法只能一句注释占用一行。使用时可单独一行，也可以使用在 PHP 语句右侧的同一行。例如：

```
<?php
    echo "这是#号风格注释的效果！";   #这是写在PHP语句右侧的单行注释
?>
```

需要特别注意的是，在单行注释中不要出现“?>”内容，否则解释器会认为是 PHP 脚本结束，而不会执行“?>”后面的代码。

3. 空白

PHP 对空格、回车造成的新行以及 Tab 等留下的空白的处理也遵循编码规范。PHP 对它们都进行忽略，这跟浏览器对 HTML 语言中的空白的处理是一样的。

合理地运用空白符，可以增强代码的清晰性和可读性。

（1）下列情况应该总是使用两个空白行。

① 两个类的声明之间。

② 一个源文件的两个代码片段之间。

（2）下列情况应该总是使用一个空白行。

① 两个函数声明之间。

② 函数内的局部变量和函数的第一个语句之间。

③ 块注释或单行注释之前。

④ 一个函数内的两个逻辑代码段之间。

（3）合理利用空格可以使代码缩进、提高可读性。

① 空格通常用于关键字与括号之间，但是，函数名称与左括号之间不使用空格分开。

② 函数参数列表中的逗号后面通常会插入空格。

③ for 语句的表达式应该用逗号分开，后面添加空格。

4. 指令分隔符

在 PHP 代码中，每个语句后需要用分号结束命令。一段 PHP 代码中的结束标记隐含表示了一个分号，所以在 PHP 代码段中的最后一行可以不用分号结束。例如：

```
<?php
    echo "这是第一个语句";                    // 每个语句都加入分号
    echo "这是第二个语句";
    // 结束标记隐含了分号，下面的语句可以省略分号
    echo "这是最后一个语句"?>
```

5. 与 HTML 语言混合搭配

凡是在一对 PHP 开始和结束标记之外的内容都会被 PHP 解析器忽略，这使得 PHP 文件可以具备混合内容。可以使 PHP 嵌入到 HTML 文档中去。例如：

```
<HTML>
<HEAD>
<TITLE>PHP与HTML混合搭配</TITLE>
</HEAD>
<BODY>
<?php
    echo "嵌入的PHP代码";
?>
</BODY>
<HTML>
```

2.3 PHP 的数据类型

从 PHP 4 开始，PHP 中的变量不需要事先声明，赋值即声明。在声明和使用这些数据类型前，读者需要了解它们的含义和特性。

2.3.1 数据类型

不同的数据类型其实就是所存储数据的不同种类。PHP 一共支持 8 种数据类型，包括整型、浮点型、字符串型、布尔型、数组、对象、资源和空类型。

（1）整型（integer，int）：用来存储整数。

（2）浮点型（float）：用来存储实数。和整数不同的是，它有小数位。

（3）字符串型（string）：用来存储字符串。

（4）布尔型（boolean）：用来存储真（true）或假（false）。

（5）数组（array）：用来存储一组数据。

（6）对象（object）：用来存储一个类的实例。

（7）资源（respirce）：资源是一种特殊的变量类型，保存了到外部资源的一个引用。

（8）空类型（NULL）：没有被赋值、已经被重置或者被赋值为特殊值 NULL 的变量。

作为弱类型语言，PHP 也被称为动态类型语言。在强类型语言中（例如 C 语言），一个变量只能存储一种类型的数据，并且这个变量在使用前必须声明变量类型。而在 PHP 中，给变量赋什么类型的值，这个变量就是什么类型的。例如以下几个变量：

```
<?php
    //由于'秦时明月汉时关'是字符串，则变量$a的数据类型就为字符串类型
    $a = '秦时明月汉时关';
    //由于9988为整型，所以$b也就为整型
    $b = 9988;
    //由于99.88为浮点型，所以$c就是浮点型
    $c = 99.88;
?>
```

由此可见，对于变量而言，变量的类型是由所赋值的类型来决定的。

实例 1：输出商品信息

本实例通过 echo 语句输出商品信息，包括名称、产地、价格和库存，代码如下：

```
<?php
    $name = "风韵牌洗衣机";
    $place = "上海";
    $price = 3998.88;
    $amount = 2000;
    echo "名称：".$name ."<br />";
    echo "产地：".$place."<br />";
    echo "价格：".$price ."元<br />";
    echo "库存：".$amount ."台";
?>
```

上述代码中，包含了整型、浮点型和字符串型数据。“.”是字符串连接符，“
”是换行标记。程序运行结果如图 2-1 所示。

图 2-1 输出商品信息

2.3.2 数据类型之间的相互转换

数据从一个类型转换到另外一个类型，就是数据类型转换。在 PHP 语言中，有两种常见的转换方式：自动数据类型转换和强制数据类型转换。

1. 自动数据类型转换

这种转换方法最为常用，直接输入数据的转换类型即可。

例如，float 型转换为整数 int 型，小数点后面的数将被舍弃。如果 float 数超过了整数的取值范围，则结果可能是 0 或者整数的最小负数。

实例 2：自动数据类型转换

```
<?php
    $fa=99.88;                          // 定义float类型
    echo  (int)$fa."<br/>";             // 转换为整数类型输出
    $fb=4E32;                           // 超过整数取值范围
    echo (int)$fb;
?>
```

程序运行结果如图 2-2 所示。

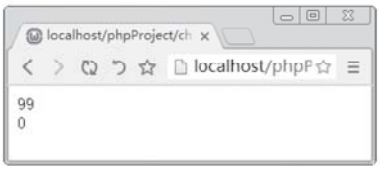


图 2-2 自动数据类型转换

2. 强制数据类型转换

在 PHP 中，可以使用 setType 函数强制转换数据类型。基本语法如下：

```
Bool setType(var, string type)
```

注意：type 的可能值不能包含资源类型数据。

实例 3：强制数据类型转换

```
<?php
    $fa = 99.88;
    echo setType($fa, "int").."<br/>";
    echo $fa;
?>
```

程序运行结果如图 2-3 所示。转型成功，则返回 1，否则返回 0。变量 fa 转成整型后值为 99。

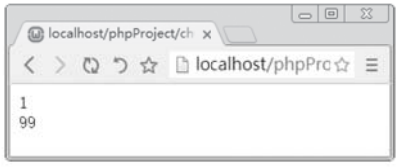


图 2-3 使用强制数据类型转换

2.3.3 检测数据类型

通过 PHP 内置的函数可以检测数据类型。针对不同类型的数据进行检测，可判断其是否属于某个类型，如果符合则返回 true，否则返回 false。

检测数据类型的函数和含义如下。

（1）is_bool()：检测变量是否为布尔类型。

（2）is_string()：检测变量是否为字符串类型。

（3）is_float() 和 is_double()：检测变量是否为浮点类型。

（4）is_int()：检测变量是否为整型。

（5）is_null()：检测变量是否为 null。

（6）is_array()：检测变量是否为数组类型。

（7）is_object()：检测变量是否是一个对象类型。

（8）is_numeric()：检测变量是否为数字或由数字组成的字符串。

实例 4：检测数据类型

```
<?php
    $fa = 99.88;            // 定义float类型
    if(is_float($fa)){   // 检测变量fa是否为浮点型变量
        echo "变量fa是浮点型变量";
    }else{
        echo "变量fa不是浮点型变量";
    }
    echo "<br />";
    if(is_null($fa)){   // 检测变量fa是否为空
        echo "变量fa是null";
    }else{
        echo "变量fa不是null";
    }
?>
```

程序运行结果如图 2-4 所示。

图 2-4 检测数据类型

2.4 常量

常量和变量是构成 PHP 程序的基础。本节来讲述常量的相关知识。

2.4.1 声明和使用常量

在 PHP 中，常量是一旦声明就无法改变的值。PHP 通过 define() 命令来声明常量。格式如下：

```
define("常量名", 常量值);
```

常量名是一个字符串，往往在 PHP 编码规范的指导下使用大写的英文字符来表示，例如 CLASS_NAME、MYAGE 等；常量值也可为表达式。

常量值可以是很多种 PHP 数据类型，可以是数组，可以是对象，当然也可以是字符和数字。常量就像变量一样存储数值，但是，与变量不同的是，常量的值只能设定一次，并且无论在代码的任何位置，它都不能被改动。

常量声明后具有全局性，函数内外都可以访问。

实例 5：声明并输出常量

```
<?php
    define("CL","烟外驿楼红隐隐，渚边云树暗苍苍。");          //定义常量CL
    echo CL;                                                  //输出常量CL
?>
```

程序的运行结果如图 2-5 所示。

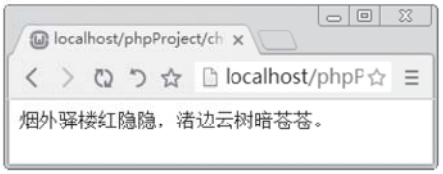


图 2-5　声明和使用常量

需要特别注意的是，从 PHP 7.3 版本开始，PHP 废弃使用与定义时的大小写不一致的名称，访问大小写不敏感的常量。这里 true、false 以及 null 除外。

2.4.2　使用系统预定义常量

PHP 的系统预定义常量是指 PHP 在语言内部预先定义好的一些量。PHP 中预定了很多系统内置常量，这些常量可以被随时调用。例如，下面是一些常见的内置常量。

（1）__FILE__：这个默认常量是 PHP 程序文件名。若引用文件（include 或 require），则在引用文件内的该常量为引用文件名，而不是引用它的文件名。

（2）__LINE__：这个默认常量是 PHP 程序行数。若引用文件（include 或 require），则在引用文件内的该常量为引用文件的行数，而不是引用它的文件的行数。

（3）PHP_VERSION：这个内置常量是 PHP 程序的版本，如 3.0.8-dev。

（4）PHP_OS：这个内置常量指执行 PHP 解析器的操作系统名称，如 Linux。

（5）TRUE：这个常量就是真值（true）。

（6）FALSE：这个常量就是伪值（false）。

（7）E_ERROR：这个常量指到最近的错误处。

（8）E_WARNING：这个常量指到最近的警告处。

（9）E_PARSE：该常量为解析语法有潜在问题处。

（10）E_NOTICE：这个量为发生异常（但不一定是错误）处。例如存取一个不存在的变量。

实例 6：输出系统常量

```
<?php
    echo "当前文件的路径是：".__FILE__;                    // 输出文件的路径和文件名
    echo "<br />";                                        // 输出换行
    echo "当前行数是：".__LINE__;                          // 输出语句所在的行数
    echo "<br />";                                        // 输出换行
    echo "当前PHP的版本是：".PHP_VERSION;                  // 输出PHP的版本
    echo "<br />";                                        // 输出换行
    echo "当前操作系统是：". (PHP_OS);                     // 输出操作系统名称
?>
```

程序的运行结果如图 2-6 所示。

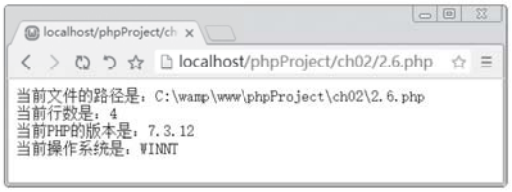


图 2-6　使用内置常量

2.5　变量

变量像是一个贴有名字标签的空盒子。不同的变量类型对应不同种类的数据，就像不同种类的东西要放入不同种类的盒子一样。

2.5.1　PHP 中的变量声明

与 C 或 Java 语言中不同的是，PHP 中的变量是弱类型的。在 C 或 Java 中，需要对每一个变量声明类型，而在 PHP 中不需要这样做。这是极其方便的。

PHP 中的变量一般以“$”作为前缀，然后以字母 a ～ z 的大小写或者下划线“_”开头。

合法的变量名可以是：

```
$hello
$Aform1
$_formhandler
```

非法的变量名如：

```
$168
$!like
```

PHP 中不需要显式地声明变量，但是定义变量前进行声明并带有注释，这是一个好的程序员应该养成的习惯。PHP 的赋值有两种，包括传值和引用，它们的区别如下。

（1）传值赋值：使用“=”直接将赋值表达式的值赋给另一个变量。

（2）引用赋值：将赋值表达式内存空间的引用赋给另一个变量。需要在“=”左右的变量前面加上一个“&”符号。在使用引用赋值的时候，两个变量将会指向内存中同一个存储空间，所以任意一个变量的变化都会引起另外一个变量的变化。

实例 7：使用两种方式赋值变量

```
<?php
    echo "使用传值方式赋值：<br/>";    // 输出 使用传值方式赋值：
    $a = "稻云不雨不多黄";
    $b = $a;                          // 将变量$a的值赋值给$b，两个变量指向不同内存空间
    echo "变量a的值为：".$a."<br/>"; // 输出 变量a的值
    echo "变量b的值为：".$b."<br/>"; // 输出 变量b的值
    $a = "荞麦空花早着霜";            // 改变变量a的值，变量b的值不受影响
    echo "变量a的值为：".$a."<br/>"; // 输出 变量a的值
    echo "变量b的值为：".$b."<p>";  //输出 变量b的值
    echo "使用引用方式赋值：<br/>"; //输出 使用引用方式赋值：
    $a = "已分忍饥度残岁";
    $b = &$a;                        // 将变量$a的引用赋给$b，两个变量指向同一块内存空间
    echo "变量a的值为：".$a."<br/>";              // 输出 变量a的值
    echo "变量b的值为：".$b."<br/>";              // 输出 变量b的值
    $a = "更堪岁里闰添长";
    //改变变量a在内存空间中存储的内容，因变量b也指向该空间，b的值也发生变化
    echo "变量a的值为：".$a."<br/>";              // 输出 变量a的值
    echo "变量b的值为：".$b." <br/>";             // 输出 变量b的值
?>
```

程序运行结果如图 2-7 所示。

图 2-7　使用两种方式赋值变量

2.5.2　可变变量和变量的引用

一般的变量很容易理解，但是有两种变量的概念比较难于理解，这就是可变变量和变量的引用。下面通过以下的例子对它们进行学习。

实例 8：可变变量和变量的引用

```
<?php
    $value0 = "guest";                  //定义变量$value0并赋值
    $$value0 = "customer";              // 再次给变量赋值
    echo $guest."<br/>";                // 输出变量
    $guest = "张飞";                    // 定义变量$guest并赋值
    echo $guest."\t".$$value0."<br/>";
    $value1 = "王小明";                 // 定义变量$value1
    $value2 = &$value1;                 // 引用变量并传递变量
    echo $value1."\t".$value2."<br/>";
    $value2 = "李丽";
```

```
    echo $value1."\t".$value2;
?>
```

上述代码的详细分析如下：

（1）在代码的第一部分中，$value0 被赋值为 guest。而 $value0 相当于 guest，则 $$value0 相当于 $guest。所以当 $$value0 被赋值为 customer 时，打印 $guest 就得到 customer。反之，当 $guest 变量被赋值为“张飞”时，打印 $$value0 同样得到“张飞”。这就是可变变量。

（2）在代码的第二部分中，$value1 被赋值为“王小明”，然后通过“&”引用变量 $value1 并赋值给 $value2。而这一步的实质是，给变量 $value1 添加了一个别名 $value2。所以打印时，都得出原始赋值“王小明”。由于 $value2 是别名，与 $value1 指的是同一个变量，所以 $value2 被赋值为“李丽”后，$value1 和 $value2 都得到新值“李丽”。

（3）可变变量其实是允许改变一个变量的变量名，允许使用一个变量的值作为另外一个变量的名。

（4）变量引用相当于给变量添加了一个别名。用“&”来引用变量。其实两个变量名指的都是同一个变量。就像是给同一个盒子贴了两个名字标签，两个名字标签指的都是同一个盒子。

程序运行结果如图 2-8 所示。

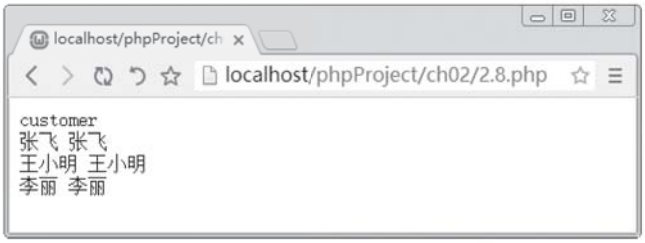


图 2-8　使用可变变量和变量的引用

2.6　PHP 运算符

PHP 包含三种类型的运算符：一元运算符、二元运算符和三元运算符。一元运算符用在一个操作数之前，二元运算符用在两个操作数之间，三元运算符是作用在三个操作数之间。

2.6.1　算术运算符

算术运算符是最简单，也是最常用的运算符。常见的算术运算符如表 2-1 所示。

表 2-1　常用运算符

运算符	名称	运算符	名称
+	加法运算	%	取余运算
-	减法运算	++	累加运算
*	乘法运算	--	累减运算
/	除法运算		

实例 9：计算部门的销售业绩差距和平均值

这里首先定义两个变量，用于存储各部门的销售额，然后使用减法计算销售业绩差距，最后应用加法和除法计算平均值。

```
<?php
    $branch1=760009;                                    //部门branch1的销售额
    $branch2=540000;                                    //部门branch2的销售额
    $sub= $branch1- $branch2;                           //销售业绩差距
    $avg= ($branch1+$branch2)/2;                        //计算平均值
    $savg=(int)$avg;                                    //销售额的平均值取整
    echo "部门1和部门2的销售业绩差距是：".$sub;
    echo "<br />";                                      //输出换行
    echo "两个部门销售业绩的平均值是：".$savg;
?>
```

程序运行结果如图 2-9 所示。

图 2-9　使用算术运算符

2.6.2　字符串连接符

字符运算符“.”把两个字符串连接起来，变成一个字符串。如果变量是整型或浮点型，PHP 也会自动地把它们转换为字符串输出。

例如下面的代码：

```
<?php
    $a = "馒头";
    $b = 2;
    echo "我今天吃了".$b."个".$a;
?>
```

输出的结果如下：

```
我今天吃了2个馒头
```

需要新手特别注意的是，对于字符串型数据，既可以使用单引号，还可以使用双引号，分别使用单引号和双引号输出同一个变量，结果却完全不同。单引号输出的是字符串，双引号输出的是变量的值。

实例 10：区分单引号和双引号在输出时的不同之处

```
<?php
    $a = "秦时明月汉时关";                              //定义一个字符串变量
    echo "$a";                                          //使用双引号输出
    echo "<br />";                                      //输出换行
```

```
    echo '$a';                                    //使用单引号输出
?>
```

程序运行结果如图 2-10 所示。

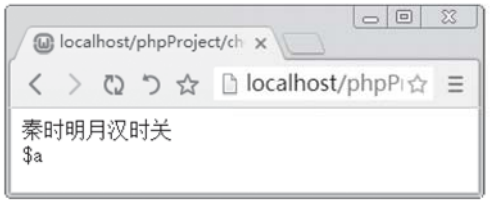


图 2-10　区分单引号和双引号的不同

2.6.3　赋值运算符

赋值运算符的作用是把一定的数值加载给特定的变量。

赋值运算符的具体含义如表 2-2 所示。

表 2-2　赋值运算符

运算符	名称
=	将右边的值赋值给左边的变量
+=	将左边的值加上右边的值，赋给左边的变量
-=	将左边的值减去右边的值，赋给左边的变量
*=	将左边的值乘以右边的值，赋给左边的变量
/=	将左边的值除以右边的值，赋给左边的变量
.=	将左边的字符串连接到右边
%=	将左边的值对右边的值取余数，赋给左边的变量

例如，$a-=$b 等价于 $a=$a-$b，其他赋值运算符与之类似。从表 2-2 可以看出，赋值运算符可以使程序更加简练，从而提高执行效率。

2.6.4　比较运算符

比较运算符用来比较其两端数值的大小。比较运算符的具体含义如表 2-3 所示。

表 2-3　比较运算符

运算符	名称	运算符	名称
==	相等	>=	大于等于
!=	不相等	<=	小于等于
>	大于	===	精确等于（类型）
<	小于	!==	不精确等于

其中，“===”和“!==”需要特别注意一下。“$b===$c”表示 $b 和 $c 不只是数值上相等，而且两者的类型也一样；“$b!==$c”表示 $b 和 $c 有可能是数值不等，也可能是类型不同。

实例 11：比较两个部门的销售业绩

```
<?php
    $branch1=760009;          //定义变量，存储部门1的销售额
    $branch2=540000;          //定义变量，存储部门2的销售额
    echo "部门1的销售业绩是：".$branch1."，部门2的销售业绩是：".$branch2;
    var_dump ( $branch1==$branch1 ) ;                    //等于操作
    var_dump ( $branch1>$branch2 ) ;                     //大于操作
    var_dump ( $branch1<$branch2 ) ;                     //小于操作
?>
```

上述代码中，var_dump() 函数用于输出变量的相关信息，包括表达式的类型与值。程序运行结果如图 2-11 所示。

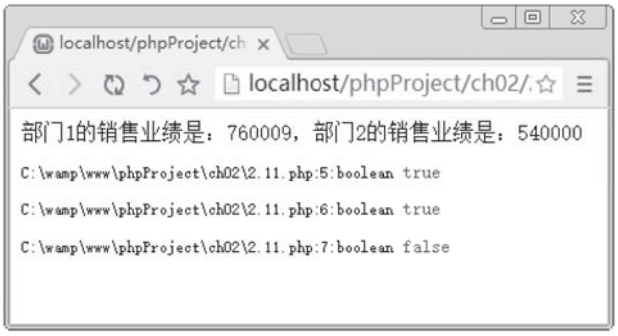


图 2-11　使用比较运算符

2.6.5　逻辑运算符

一个编程语言最重要的功能之一就是进行逻辑判断和运算，比如逻辑和、逻辑或、逻辑非。逻辑运算符的含义如表 2-4 所示。

表 2-4　逻辑运算符

运算符	名称	运算符	名称
&&、AND	逻辑和	!、NOT	逻辑非
||、OR	逻辑或	XOR	逻辑异或

2.6.6　按位运算符

按位运算符是把整数以“位”为单位进行处理。按位运算符的含义如表 2-5 所示。

表 2-5　按位运算符

运算符	名称	运算符	名称
&	按位和	^	按位异或
|	按位或		

2.6.7　否定控制运算符

否定控制运算符用在“操作数”之前，用于操作数真假的判断。否定控制运算符的含义

如表 2-6 所示。

表 2-6　否定控制运算符

运算符	名称	运算符	名称
!	逻辑非	~	按位非

2.6.8　错误控制运算符

错误控制运算符是用“@”来表示的，在一个操作数之前使用，该运算符用来屏蔽错误信息的生成。

2.6.9　三元运算符

三元运算符“?:”是作用在三个操作数之间的。语法格式如下：

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

如果表达式 expr1 为真，则返回 expr2 的值；如果表达式 expr1 为假，则返回 expr3 的值。从 PHP 5.3 开始，可以省略 expr2，表达式为（expr1）?:（expr3），如果表达式 expr1 为真，则返回 expr1 的值；如果表达式 expr1 为假，则返回 expr3 的值。

实例 12：使用三元运算符

```
<?php
    $a = 100>99;
    $b = $a ?: '100不大于99';
    $c = $a ? '100大于99': '100不大于99';
    echo $b;
    echo "<br />";                                    //输出换行
    echo $c;
?>
```

程序运行结果如图 2-12 所示。

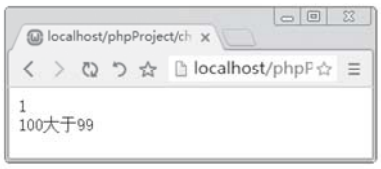


图 2-12　三元运算符

2.6.10　运算符的优先级和结合规则

运算符的优先级和结合规则其实与正常的数学运算符的规则十分相似：

- 加减乘除的先后顺序与数学运算中的完全一致。
- 对于括号，则先运行括号内，再运行括号外。
- 对于赋值，则由右向左运行，也就是依次从右边向左边的变量进行赋值。

2.7 PHP 7 的新特性——合并运算符和组合运算符

PHP 7 新增加的合并运算符“??”用于判断变量是否存在且值不为 NULL，如果是，它就会返回自身的值，否则返回它的第二个操作数。

语法格式如下：

```
(expr1) ? ? (expr2)
```

如果表达式 expr1 为真，则返回 expr1 的值，如果表达式 expr1 为假，则返回 expr2 的值。

实例 13：使用合并运算符

```
<?php
    $a = 100>99;
    $b = 10>99;
    $c = $a ?? $b;
    echo $c;
?>
```

程序运行结果如图 2-13 所示。

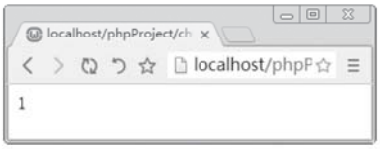


图 2-13 合并运算符

PHP 7 新增加的组合运算符“<=>”，用于比较两个表达式 $a 和 $b，如果 $a 小于、等于或大于 $b 时，它分别返回 -1、0 或 1。

实例 14：使用组合运算符

```
<?php
    // 整型比较
    echo( 100 <=> 100);echo "<br />";
    echo( 100 <=> 99);echo "<br />";
    echo( 100 <=> 120);echo "<br />";

    // 浮点型比较
      echo( 99.99 <=> 99.99);
echo "<br />";
      echo( 88.88 <=> 99.99);
echo "<br />";
      echo( 99.99 <=> 88.88);
echo "<br />";
    echo(PHP_EOL);

    // 字符串比较
    echo( "a" <=> "a");echo "<br />";
    echo( "a" <=> "b");echo "<br />";
    echo( "b" <=> "a");echo "<br />";
?>
```

程序运行结果如图 2-14 所示。

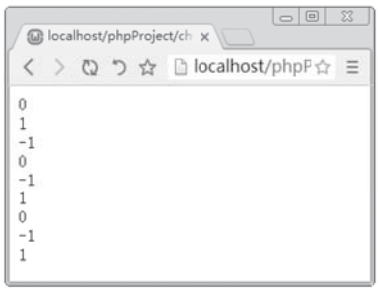


图 2-14 组合运算符

2.8 PHP 中的表达式

表达式是表达一个特定操作或动作的语句。

表达式由“操作数”和“操作符”组成。操作数可以是变量，也可以是常量。操作符则体现了要表达的各个行为，如逻辑判断、赋值、运算等。

例如“$a=5”就是表达式，而“$a=5;”则为语句。另外，表达式也有值，例如表达式“$a=1”的值为 1。

> **提示：**在 PHP 代码中，使用“;”号来区分表达式和语句，即一个表达式和一个分号组成一条 PHP 语句。在编写代码程序时，应该特别注意表达式后面的“;”，不要漏写或写错，否则会提示语法错误。

2.9 新手疑难问题解答

疑问 1：如何快速区分常量和变量?

常量和变量的明显区别如下：

（1）常量前面没有美元符号（$）。

（2）常量只能用 define() 函数定义，而不能通过赋值语句定义。

（3）常量可以不用理会变量范围的规则，可以在任何地方定义和访问。

（4）常量一旦定义就不能被重新定义或者取消定义。

（5）常量的值只能是标量。

疑问 2：PHP 中的常见的输出方式有几种?

在 PHP 中，常见的输出语句如下：

（1）echo 语句：可以一次输出多个值，多个值之间用逗号分隔。这是 PHP 中最常用的输出语句。

（2）print 语句：只允许输出一个字符串。例如以下代码：

```
print "Hello world!";
```

（3）print_r() 函数：可以把字符串和数字简单地打印出来，而数组则以括起来的键和值的列表形式显示，并以 Array 开头。但 print_r() 输出布尔值和 NULL 的结果没有意义，因为都是打印“\n”，因此用 var_dump() 函数更适合调试。

（4）var_dump() 函数：判断一个变量的类型与长度，并输出变量的数值。如果变量有值，输出的是变量的值并返回数据类型。此函数显示关于一个或多个表达式的结构信息，包括表达式的类型与值。

2.10 实战技能训练营

实战 1：输出学生信息

使用 echo 语句输出学生信息，包括学号、姓名、性别、年龄和总成绩，运行结果

如图 2-15 所示。

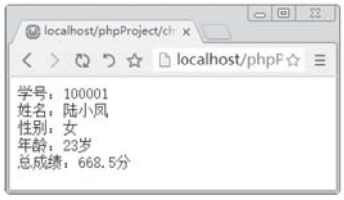


图 2-15　输出学生信息

实战 2：计算长途汽车行驶一段距离所需的时间

本实例将编写一个程序，计算长途汽车以 90 千米 / 小时的速度行驶 800 千米需要多长时间，答案以“* 小时 * 分”的格式。运行结果如图 2-16 所示。

图 2-16　输出所需的时间