第3章 PHP语言基础

PHP自1994年诞生以来,已席卷整个互联网,不仅有着广泛的用户群,更有着庞大的开发团队。PHP社区则是全球最活跃的开发社区之一,数以千万计的人们可以在此共享代码、技术交流。

PHP是运行在服务器端的,而HTML、CSS、JavaScript都是运行在浏览器上的。有时也把针对浏览器的网页设计称为Web前端开发,而把服务器端程序开发称为Web后台编程。

3.1 PHP语法入门

学习PHP语言的基本语法是进行PHP编程开发的第一步,PHP语言的语法混合了C、Java和Perl语言的特点,语法非常灵活,与其他编程语言有很多不同之处,读者如果学习过其他语言,可通过体会PHP与其他语言的区别来学习PHP。

3.1.1 PHP代码的基本格式

PHP是一种可嵌入HTML中的脚本语言。PHP的代码一般由两部分组成:

(1) HTML代码,其中还包括嵌入其中的CSS和JavaScript代码;

(2) 服务器端脚本,位于PHP定界符"<?"与"?>"之间的代码。

其中(1)是静态网页也具备的,它通过浏览器解释执行,又称为客户端代码。因此,PHP可以通俗地认为是把服务器端脚本放在"<?"和"?>"之间,再嵌入静态网页中。

提示:"<?"和"?>"称为PHP脚本的定界符,表示脚本的开始和结束。这是因为在PHP文件中,HTML代码和PHP代码混杂在一起(即页面和程序没有分离),必须使用专门的定界符对PHP代码进行区分。这样服务器就可以只对"<?"和"?>"之间的代码进行执行了。

1. PHP代码的4种风格

根据定界符的不同,PHP代码有4种风格,即XML风格、简短风格、脚本风格和ASP风格。使用任意一种都可将PHP代码嵌入HTML中去。

1）XML 风格

这种风格的 PHP 定界符是“ <? php ”和“? >”(<? 和 php 之间不能有空格)。例如：

```
<h1 > <? php echo '现在是'.date("Y 年 m 月 d 日 H:i:s");? > </h1 >
```

2）简短风格

将定界符“ <? php ”中的“php”省略,就成了简短风格,它的定界符是“ <? ”和“? >”。要使用简短风格,必须保证 php. ini 文件中的 short_open_tag = On(默认是开启的),本书中的 PHP 代码都采用这种风格。

3）脚本风格

这种风格将 PHP 代码写在 <script > 标记对中,例如：

```
<h1 > <script language = 'php' >echo '现在是'.date("m 月 d 日"); </script > </h1 >
```

4）ASP 风格

这种标记风格将 PHP 代码写在“ <% ”和“%> ”中,不推荐使用,并且默认是不能使用这种风格的,因为 php. ini 文件中的 asp_tags = Off。

2. PHP 代码的注释

注释即代码的解释和说明,程序执行时,注释会被 PHP 解析器忽略,因此浏览器端看不到 PHP 代码的注释。PHP 支持三种风格的注释。

(1）单行注释(//或#)。

```
<? echo 'PHP 动态网页';                    //输出字符串
                                           #单行注释用#号也可以
? >
```

需要注意的是,单行注释的内容中不能含有“? >”,否则解释器会认为 PHP 的脚本到此结束了,而去执行“? >”后面的代码,例如：

```
<h1 > <? echo '这样会出错的';              //不会看到? >会看到
? > </h1 >
```

(2）多行注释(/ * … * /)。

如果要添加大段的注释,则使用多行注释更方便,但多行注释符不允许嵌套使用,例如：

```
<h1 > <? echo '这样不会出错';              / * 多行注释的内容
不会被输出? > * / ? > </h1 >
```

3.1.2　简单 PHP 程序示例

利用 PHP 程序可以输出字符串、HTML 代码或 JavaScript 代码,下面是几个例子。

1. 输出当前日期时间

下面的程序以 h1 标题的形式输出当前日期和时间(3-1. php),代码如下。

```
<h1>
    <?echo '现在是'.date("Y年m月d日 H:i:s");?>
</h1>
```

在该程序中，<h1>和</h1>是 HTML 代码，<? … ?>是 PHP 代码。其中，echo 是 PHP 的输出函数，单引号括起来的表示这是一个字符串常量，“.”是字符串连接符，date()是时间日期函数，可以按指定的格式获取当前日期和时间。运行程序会在网页上以一级标题的形式显示：

现在是 2014 年 03 月 18 日 16:20:55

2. 输出不同大小的字体

下面的程序使用 PHP 的循环语句重复输出 <font> 标记(3-2. php)，其运行效果如图 3-1 所示。

```
<html><body>
<?echo '<p>PHP 代码和 HTML 代码可相互嵌套</p>';
for($i=3; $i<7; $i++){ ?>
    <font size="<?echo $i;?>">第<?echo $i-2;?>次 Hello World!
</font><br />
<?}?>
</body>
</html>
```

HTML代码嵌入PHP代码中

PHP代码嵌入HTML代码中

图 3-1　3-2. php 或 3-3. php 的运行结果

在该程序中，使用 for 循环语句循环输出 HTML 代码“…
”。从结构上，这条 HTML 代码被 PHP 代码包含。$i 是程序中定义的一个变量，PHP 规定所有变量名必须以“$”开头。可以看出，PHP 代码可以位于 HTML 代码的任意位置。如标记外：<? for($i=3;$i<7;$i++){ ?>、<?}?>，标记内：<? echo $i-2;?>，甚至是标记的属性内：<? echo $i;?>。从结构上看，可以是 HTML 代码中包含 PHP 代码，也可以是 PHP 代码中包含 HTML 代码。实际上，PHP 代码还可与 CSS 或 JavaScript 等浏览器端代码互相嵌入，因为 PHP 解析器只对“<?”和“?>”之间的代码进行处理。

注意：PHP 代码的定界符“<?”和“?>”不能够嵌套。如果遇到 HTML 代码(如<font…)，就必须立即用“?>”把前面的 PHP 代码结束，即使这段代码并不完整(但其中每行语句必须是完整的)。

3. 用 PHP 程序输出 HTML 代码，实现 3-2. php 的功能

在 3-2. php 中，由于 PHP 代码和 HTML 代码频繁地交替出现，以致经常需要使用定界符关闭和开始一段 PHP 代码，而如果把 HTML 代码当成字符串通过 PHP 程序来输出

(3-3. php),则可避免该问题。代码如下,运行效果如图 3-1 所示。

```
<html><body>
    <p>PHP 代码和 HTML 代码可相互嵌套</p>
    <?for($i=3;$i<7;$i++){
        echo '<font size='.$i.'>第'.($i-2).'次 Hello World!</font>
        <br/>';
    }?>
</body></html>
```

提示:使用 PHP 程序输出 HTML 代码是一项常用技巧。总的原则是:如果 PHP 代码之间的 HTML 代码很短,则使用 PHP 程序输出这些 HTML 代码更合适,而如果 PHP 代码之间的 HTML 代码很长,则还是作为外部 HTML 代码合适些。这样会使程序的可读性改善,并减少编写时出错的概率。

4. 输出 JavaScript 代码并传递变量值给 JavaScript 或表单

下面的程序定义了两个变量 $str1 和 $str2(3-4. php),并将字符串赋值给这两个变量(PHP 中没有变量声明语句,变量不需要声明就可赋值使用)。因为 JavaScript 代码也是客户端代码,可以使用 PHP 将 JavaScript 代码作为字符串输出。如果在输出的 JavaScript 代码或表单代码中嵌入了 PHP 变量,就可以把这些服务器端变量值传递到客户端。

```
<?    $str1="Hello";                          //在弹出框中显示
    $str2="start PHP";                        //在文本框中显示
    echo "<script>";
    echo "alert('".$str1."');";               //在 JavaScript 中使用 $str1 变量
    echo "</script>";    ?>
<input type="text" name="tx" size=20 value="<?echo $str1; ?>">
<input type="button" value="单击" onclick="tx.value='<?echo $str2; ?>'">
```

运行该程序,会在弹出警告框和文本框中显示 Hello,当单击按钮后,文本框中的内容会变为 start PHP。

5. 编写 PHP 程序的注意事项

(1) PHP 是一种区分大小写的语言,表现在:①PHP 中的变量和常量名是区分大小写的,② 但 PHP 中的类名和方法名,以及一些关键字(如 echo、for)都是不区分大小写的。在书写时,建议除了常量名以外的其他符号都小写。

(2) PHP 代码中的字符均为半角(英文状态下)字符,中文或全角字符只能出现在字符串常量中。

(3) 在“<?”和“?>”内必须是一行或多行完整的语句,如<? for($i=3;$i<7;$i++)?>不能写成<? for($i=3;?> <? $i<7;$i++)?>。

(4) 在 PHP 中,每条语句以“;”号结束,PHP 解析器只要看到“;”号就认为一条语句结束了。因此,可以将多条 PHP 语句写在一行内,也可以将一条语句写成多行。

3.2 常量、变量和运算符

3.2.1 常量和变量

1. 常量

在程序运行中,其值不能改变的量,称为常量,常量通常直接书写,如 10、-3.6、"hello" 都是常量,除此之外,还可以用一个标识符代表一个常量,称为符号常量。在 PHP 中使用 define( ) 函数来定义符号常量,符号常量一旦定义就不能再修改其值。另外,使用 defined( ) 函数可以判断一个符号常量是否已被定义。例如:

```
<?  define("PI","3.1416");                  //定义符号常量 PI,并且区分大小写
define("SITE","网页设计学习网",true);        //定义符号常量 SITE,不区分大小写
echo(defined("PI"));                        //如果已被定义则返回"1"
? >
```

在 PHP 中,还预定义了一些符号常量,如表 3-1 所示(注意__FILE__等常量左右两边是双下划线),这些符号常量可直接使用,如“echo __FILE__;”。

表 3-1 PHP 预定义的符号常量

常　　量	功　　能
__FILE__	存储当前脚本的物理路径及文件名称
__LINE__	存储该常量所在的行号
__FUNCTION__	存储该常量所在的函数名称
PHP_VERSION	存储当前 PHP 的版本号
PHP_OS	存储当前服务器的操作系统名

2. 变量

变量是指程序运行过程中其值可以变化的量,变量包括变量名、变量值和变量的数据类型三要素。PHP 的变量是一种弱类型变量,即 PHP 变量无特定数据类型,不需要事先声明,并可以通过赋值将其初始化为任何数据类型,也可以通过赋值随意改变变量的数据类型。下面是一些变量声明(定义)和赋值的例子:

```
<? $str1 = "PHP 变量 1";          //该变量为字符串变量
$num = 10 + 2 * 9;                //该变量为数值型变量
$_date = "2013 - 9 - 8";          //该变量为字符串变量,PHP 无日期型数据类型
$bol = true;                      //该变量为布尔型变量
$num = '赋值字符串';               //通过赋值改变变量的数据类型
$str1 = $num + $_date;            //执行相加运算会使 $str1 转换为数值型,值为 2013
var_dump( $num, $_date, $bol);    // var_dump 函数可输出变量的类型
? >
```

说明：

① PHP 变量必须以“ $ ”开头，区分大小写。

② 变量使用前不需要声明，PHP 中也没有声明变量的语句。

③ 变量名不能以数字或其他字符开头，其他字符包括@、#等。例如：$ xm，$ _id，$ sfzh 都是合法的变量名，而 $ -id，$ 57zhao，$ zh fen 都是非法的变量名。

④ 变量名长度应小于 255 个字符，不能使用系统关键字作为变量名。

3.2.2　变量的作用域和生存期

1. 变量的作用域

变量的作用域是指该变量在程序中可以被使用的范围。对于 PHP 变量来说，如果变量是定义在函数内部的，则只有这个函数内的代码才可以使用该变量，这样的变量称为“局部变量”。如果变量是定义在所有函数外的变量，则其作用域是整个 PHP 文件，减去用户自定义的函数内部（注意这和 ASP VBScript 语言是不同的），称为“全局变量”。例如：

```
<?    $a = "全局变量 <br > ";              //该变量为全局变量
function fun(){
    echo $a;                               //调用函数也不会输出"全局变量"
     $a = "局部变量";                      //该变量为局部变量
    echo $a;
  }
  fun();                                   //输出"局部变量"
echo $a;                                   //输出"全局变量"     ? >
```

输出结果为“局部变量、全局变量”。可见函数内不能访问函数外定义的变量。

如果一定要在函数内部引用外部定义的全局变量，或者在函数外部引用函数内部定义的局部变量，可以使用 global 关键字，示例代码（global. php）如下：

----------global. php----------

```
<?    $a = "全局变量";
function fun(){
       global $a;                          //为了引用函数外定义的变量 $a
       echo $a;
        $a = "局部变量";                   //将修改 $a 的值，添加 static 试试
       echo $a;                            //输出"局部变量"
       }
   fun();                                  //调用函数将输出"全局变量、局部变量"
echo $a;                                   //输出"局部变量"
? >
```

输出结果为“全局变量、局部变量、局部变量”。

提示：

① global 的作用并不是将变量的作用域设置为全局，而是起传递参数的作用。在函

数外部声明的变量,如果想在函数内部使用,就在函数内用 global 来声明该变量。

② 不能在用 global 声明变量的同时给变量赋值。例如 global $a = "全局" 是错误的。

③ global 只能写在自定义函数内部,写在函数外部没有任何用途。

④ 对于 global 变量,应该用完之后就用 unset( )销毁,因为它很占用资源。

另外,使用 $GLOBALS[ ]全局数组也能实现在函数内部引用外部变量,例如:

```
<?    $a = "全局变量";
function fun(){
    echo $GLOBALS['a'];
    $a = "局部变量";
    echo $a;                                    //输出"局部变量"
    }
fun();                                          //调用函数将输出"全局变量、局部变量"
echo $a;                                        //输出"全局变量"
? >
```

则输出结果为“全局变量、局部变量、全局变量”。可见 $GLOBALS[ ]和 global 是有区别的,它只能在函数内部引用外部变量,但不能在函数外部引用函数内部定义的局部变量。

2. 变量的生存期

变量的生存期表示该变量在什么时间范围内存在。全局变量的生存期从它被定义那一刻起到整个脚本代码执行结束为止;局部变量的生存期从它被定义开始到该函数运行结束为止。

可见,一般的局部变量在函数调用结束后,其存储的值会被自动清除,所占存储空间也会被释放。为了能在函数调用结束后仍保留局部变量的值,可使用静态变量,这样当再次调用函数时,又可以继续使用上次调用结束后的值。静态变量使用 static 关键字定义,例如:

```
<?  function Test(){
          static $w = 0;                        //声明静态变量 $w
          echo $w;
           $w ++;          }
Test();Test();Test();Test();Test();          ? >
```

程序的输出结果为 01234。而如果去掉程序中的 static,则运行结果为 00000。

提示:

① 静态变量仅在局部函数域中存在,函数外部不能引用函数内部的静态变量。例如将 global. php 中的“ $a = "局部变量";”改为“static $a = "局部变量";”,则最后一条语句将输出“全局变量”。

② 对静态变量赋值时不能将表达式赋给静态变量。如 static $int = 1 + 2; static $int = sqrt(9);都是错误的。

表 3-2 对三种类型的变量进行了总结。

表 3-2　变量根据作用域和生存期分类

类　　型	说　　明
全局变量	定义在所有函数外的变量,其作用域是整个 PHP 文件,减去用户自定义的函数内部
局部变量	定义在函数内部的变量,只有这个函数内的代码才可以使用该变量
静态变量	是局部变量的一种,能够在函数调用结束后仍保留变量的值

3.2.3　可变变量和引用赋值

1. 可变变量

可变变量是一种特殊的变量,这种变量的名称不是预先定义的,而是动态地设置和使用的。可变变量一般是使用一个变量的值作为另一个变量的名称,所以可变变量又称为变量的变量。可变变量直观上看就是在变量名前加一个“ $ ”,例如:

```
<?   $a = 'b';                          //定义变量 $a
     $b = '一个变量 <br > ';             //定义变量 $b
     echo  $ $a;                        // $ $a 就是一个可变变量,相当于 $b
     $b = '变化后';
     echo  $ $a;                        //通过可变变量输出变量 $b 的值
     $a = 'c';
     echo  $ $a;                        //相当于输出变量 $c 的值
  ? >
```

输出结果是“一个变量
 变化后”。由于没有给 $ *c* 赋值,第三条 echo 语句不会输出任何内容。

2. 引用赋值

从 PHP 4.0 开始,提供了“引用赋值”功能,即新变量引用原始变量的地址,修改新变量的值将影响原始变量,反之亦然。引用赋值使得不同的变量名可以访问同一个变量内容。使用引用赋值的方法是:在将要赋值的原始变量前加一个“&”符号,例如:

```
<?   $b =10;
$a = "hello ";                          //$a 赋值为 hello
$b = & $a;                              //变量$b 引用 $a 的地址
echo  $a;                               //输出结果为 hello
$b = "world ";                          //修改 $b 的值, $a 的值将一起变化
echo  $a;                               //输出结果为 world
$a = "cup";                             //修改 $a 的值, $b 的值将一起变化
echo  $b;                               //输出结果为 cup
? >
```

引用赋值的原理如图 3-2 所示。引用赋值后,两个变量指向同一个地址单元,改变任

意一个变量的值(即地址中的内容),另一个变量值也会随之改变。

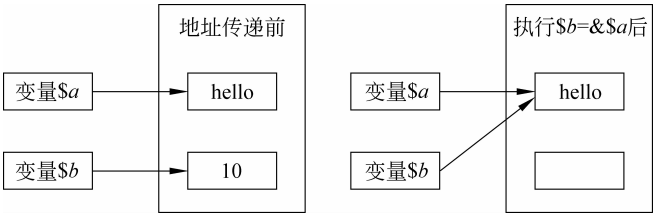

图 3-2 引用赋值——变量地址传递示意图

注意:只有已经命名过的变量才可以引用赋值,例如下面的用法是错误的:$bar = &(25 * 5)。

3.2.4 运算符和表达式

PHP 的运算符包括算术运算符、比较运算符、逻辑运算符、赋值运算符、连接运算符等。而表达式就是由常量、变量和运算符组成的符合语法要求的式子。PHP 主要有 5 种表达式,即数学表达式(如 3 +5 * 7)、字符串表达式(如" abc" ." gh")、赋值表达式(如 $a + = $b)、关系表达式(如 $i==5)和逻辑表达式(如 $a || $b&&$c)。

1. 算术运算符

算术运算符有:加(+)、减(-)、乘(*)、除(/)、取余(%)等。算术运算符的运算结果是一个算术值。例如:$a =7/2 +4 * 5 +1,结果是 24.5。$b = -7%3,结果是 -1(对于取余运算符来说,如果被除数是负数,那么取得的结果也是负数)。如果对 10 求余可得到一个数个位上的数字。

如果算术运算符的左右两边任一操作数或两个操作数都不是数值型,那么会将操作数先转换成数值型,再执行算术运算。

例如:$a =10 +'20',结果为 30。$a ='10' +'20',结果为 30。$a ='10' +'2.2ab8',结果为 12.2。$a ='10' +'ab2.2',结果为 10。$a ='10' + true,结果为 11。

字符串转换为数值型的原则是:从字符串开头取出整数或浮点数,如果开头不是数字的话,就是 0。布尔型的 true 会转换成数值 1,false 转成数值 0。

2. 连接运算符

PHP 中连接运算符只有一个,即“.”,它用于将两个字符串连接起来,组成一个新字符串。如果连接运算符左右两边任一操作数或两个操作数都不是字符串类型,那么会将操作数先转换成字符串,再执行连接操作,例如:

```
$a = 'PHP'. 5;                    // $a 的值为 PHP5,注意数字和"."之间要用空格隔开
$b = 'PHP'.'5';                   // $b 的值为 PHP5
$c = "PHP".true;                  // $c 的值为 PHP1
$d =5. 'PHP';                     // $d 的值为 5PHP
```

提示:如果“.”的左右有数字,注意将“.”和数字用空格隔开。

可见“.”是强制连接运算符,不管左右两边是什么数据类型,都会执行连接运算。

3. 赋值运算符

最基本的赋值运算符是“=”,它用于对变量赋值,因此它的左边只能是变量,而不能是表达式。例如:$a=3+5, $b=$c=9 都是合法的。此外,PHP 还支持像 C 语言那样的赋值运算符与其他运算符的缩写形式,如“+=”、“.=”、“&=”、“|=”等。

如 $a+=3 等价于 $a=$a+3, $a.=3 等价于 $a=$a.3。

4. 比较运算符

比较运算符会比较其左右两边的操作数,如果比较结果为真,则返回 true,否则返回 false。PHP 中的比较运算符有:是否相等(==)、大于(>)、小于(<)、大于等于(>=)小于等于(<=)、不等于(!=或<>)、恒等于(===)、非恒等于(!==)。

其中恒等于(===)表示数值相等并且数据类型也相同,非恒等于(!==)表示数值不相等或者数据类型不相等。

例如,若 $a=6, $b=3,则 $a<$b 返回 false, $a>$b 返回 true, $a<>$b 返回 true。$c="PHP"<"php"返回 true。$c="5"==5 返回 true, $c="5"===5 返回 false, $c=1==true 返回 true, $c=1!==true 返回 true。

5. 逻辑运算符

逻辑运算符用来组合逻辑运算的结果,例如对两个布尔值或两个比较表达式进行逻辑运算,再返回一个布尔值(true 或 false)。PHP 中的逻辑运算符有逻辑非(!)、逻辑与(&&或 and)、逻辑或(||或 or)、逻辑异或(xor)。

例如:!5<3 && 'b'=="b"返回 true,!(5>3 && 'b'==="b")返回 false。

逻辑与(&&和 and),逻辑或(||和 or),虽然含义相同,但它们的优先级却不同。“&&”的优先级比“and”高,“||”的优先级比“or”高。例如:$c=(1 or 2 and 0),会返回 true。因为 or 和 and 优先级相同,则按自右至左的执行顺序,先执行 2 and 0。

而 $c=(1||2 and 0),会先执行 1||2,再执行 true and 0,最终返回 false。

又如 $c=false or 1,返回 false, $c=false||1,返回 true。因为“=”的优先级比“or”高,但是比“||”低。

6. 加 1/减 1 运算符

加 1/减 1 运算符与 C 语言中的加 1/减 1 运算符相同,包括前加(++$a)、后加($a++)、前减(--$a)、后减($a--)4 种形式。

前加操作是先加 1,再赋值,后加操作是先赋值,再加 1。例如:$a=6; $b=++$a,则执行完后, $a=7, $b=7。$a=6; $b=$a++,执行完后, $a=7, $b=6。

前减操作和后减操作的规则与此相同。

7. 条件运算符

条件运算符是一个三元运算符,其语法如下:

```
条件表达式 ? 表达式 1 : 表达式 2
```

如果条件表达式的结果为 true,则返回表达式 1 的值,否则返回表达式 2 的值。

例如下面的表达式会得到 Yes。

```
$c = 10 > 2 ? "Yes" : "No"
```

在分页程序中,常通过条件运算符判断要显示的分页页面,如果获取的分页变量 page 的值存在,则显示该分页,如果获取不到 page 变量值,则显示第 1 页,代码如下:

```
$page = (isset( $_GET['page']))? $_GET['page']:"1";
```

8. 执行运算符

执行运算符,即反引号(`)(键盘上的反引号键在数字 1 键的左边)。可用来执行 Shell 命令。在 PHP 脚本中,将外部程序的命令行放入反引号中,并使用 echo( )或 print( )函数将其显示,PHP 将会在到达该行代码时启动这个外部程序,并将其输出信息返回,其作用效果与 shell_exec( )函数相同,例如:

```
<?   $output = `dir`;
echo $output;                          //输出当前目录下的内容
echo shell_exec(`dir `);               //输出当前目录下的内容,结果同上     ? >
```

提示:IIS 出于安全性考虑,禁止使用执行运算符,执行运算符只能在 Apache 中使用。

3.3 数据类型及类型转换

数据类型:是一个值的集合以及定义在这个值集上的一组操作。数据类型的使用往往和变量的定义联系在一起。虽然 PHP 定义变量时不需要指定数据类型,但它会根据对变量所赋的值自动确定变量的数据类型。确定了变量的数据类型就确定了变量的存储方式(占多少字节)和操作方法。PHP 具有的数据类型如表 3-3 所示。

表 3-3 PHP 中的数据类型

数 据 类 型	具 体 描 述
整型(integer)	即整数,占 4 个字节(32 位),取值范围从 -2 147 483 648 到 2 147 483 647,可以采用十进制、八进制(0 作前缀)、十六进制(0x 作前缀)表示
浮点型(float)	即实数(包含小数的数),如 1.0、3.14
布尔型(boolean)	只有 true(逻辑真)和 false(逻辑假)两种取值
字符串(string)	是一个字符的序列。组织字符串的字符可以是字母、数字或者符号
数组(array)	由一组相同数据类型的元素组成的数据结构,每个元素都有唯一的编号
对象(object)	是面向对象语言中的一种复合数据类型,对象就是类的一个实例

续表

数 据 类 型	具 体 描 述
NULL	空类型,只有一个值 NULL。如果变量未被赋值,或是被 unset( ) 函数处理后的变量,其值就是 NULL
资源(resource)	资源是 PHP 特有的一种特殊数据类型,用于表示一个 PHP 的外部资源,如一个数据库的访问操作,或者打开保存文件操作。PHP 提供了一些特定的函数,用于建立和使用资源
伪类型	只用于函数定义中,表示一个参数可接受多种类型的数据,还可以接受别的函数作为回调函数使用

3.3.1　字符串数据类型

任何由字母、数字、文字、符号组成的 0 到多个字符的序列都叫做字符串。在 Web 程序中,经常需要对字符串进行操作。如截取标题、连接字符串常量和变量等。PHP 规定字符串的前后必须加上单引号(')或双引号(")，例如："这是一个字符串"、'另一个字符串'、'5'、'ab'、"(空字符串)都是合法的字符串。但单双引号不能混用,如'day" 是非法的。

如果字符串中出现单引号(')或双引号(")，则需要使用转义字符(\'或\")来输出,例如:

```
echo 'I\'m a boy';                    //输出结果为 I'm a boy
```

1. 单引号字符串和双引号字符串

单引号表示包含的是纯粹的字符串;而双引号中可以包含字符串和变量名。双引号中如果包含变量名则会被当成变量,会自动被替换成变量值,单引号中的变量名则不会被当成变量,而是把变量名当成普通字符输出。示例代码如下,运行效果如图 3-3 所示。

图 3-3　单引号字符串与双引号字符串

```
<? $a = 'tang';
$b = 10;
echo '你好 $a';                       //使用单引号输出 $a
echo '<br>';
echo "你好 $a";                       //使用双引号输出变量
echo "你是第 $b 次光临";              //使用双引号输出变量    ? >
```

可见,在双引号字符串中,$a 和 $b 被解析成了变量 $a 和 $b 的值。因此建议:如果要书写纯字符串,建议用单引号字符串;如果要对字符串和变量进行连接操作,可以使用双引号字符串,以简化写法,例如:

```
echo "你是第 $b 次光临";              //注意 $b 后面要有个空格
```

等价于:

```
echo '你是第 '. $b .'次光临';
```

注意：在双引号字符串中，如果变量名后有其他字符的话，要在变量名后加空格，否则 PHP 解析器会认为后面的字符也是变量名的一部分，例如：

```
$sport = 'basket';
$hobby = "I like play $sportball.";          //包含变量的错误方法
echo $hobby;
```

则 PHP 解析器认为双引号中的变量是 $ sportball，而 $ sportball 未定义，视其值为空。因此会输出“I like play”。为解决这个问题，可以加空格，或用大括号将变量名包含起来。

```
$hobby = "I like play { $sport}ball.";       //包含变量的正确方法
$hobby = "I like play $sport ball.";         //包含变量的正确方法
```

双引号比单引号支持更多的转义字符，双引号支持的转义字符如表 3-4 所示。

表 3-4　双引号支持的转义字符及含义

转义字符	含义	转义字符	含义	转义字符	含义
\n	换行	\t	跳格 Tab	\\	反斜杠\
\r	回车	\"	双引号	\ $	显示 $ 符号

例如要在双引号字符串中输出 $ 符号、反斜杠和换行符，代码如下：

```
echo "变量\$a = '\\t'\n";                    //输出结果为:变量 $a = '\t'(换行)
```

但是换行符会被浏览器当成空格忽略，只有在网页源代码中才能看到换行符的效果。

2. 界定符表示字符串

除了使用单引号或双引号表示字符串外，还可使用界定符表示字符串或变量，例如：

```
<?   $i = '显示该行内容';
    echo < < < STD
    双引号""可直接输出，\$i 同样可以被输出 <br >
    \$i 的内容为: $i
STD;
? >
```

输出结果为：

```
双引号""可直接输出，$i 同样可以被输出
$i 的内容为:显示该行内容
```

说明：

① 程序中的 STD 是自定义的界定符，也可以使用任何其他标识符，只要首尾界定符相同即可。

② 开始界定符前面必须有三个左尖括号（< < <），后面不能有任何空格。结束界定符必须单独另起一行，前后不能有空格或任何其他字符（包括注释符），否则都会引起语

法错误。

③ 界定符和双引号唯一的区别是界定符中的双引号不需要转义就能显示,因此,如果需要处理大量的内容,同时又不希望频繁使用各种转义字符,则使用界定符更合适。

3. 获取字符串中的字符

在 PHP 中,可以通过给字符串变量加下标的方式获取字符串中的字符,语法为:

```
字符串变量[index]
```

其中,index 指定字符的位置,0 表示第 1 个字符,1 表示第 2 个字符,以此类推。但该方法不能用来获取中文字符,否则会出现乱码,因为一个中文占两个字符,例如:

```
<?   $i = 'Tom & Mary';
   echo $i[1] . $i[4];                    //输出结果为 o&,因为空格也算一个字符
? >
```

4. 获取字符串的长度

使用 strlen( ) 函数可获取字符串的长度,该函数的参数是一个字符串,例如:

```
<?  echo strlen('喜欢 PHP!');    ? >
```

输出的结果是 8,这是因为每个中文字符占 2 个字节,加上后面 4 个英文字符总共占 8 个字节。如果要计算中文字符串的长度,可以使用 mb_strlen( ) 函数,例如:

```
<?  echo mb_strlen('喜欢 PHP!',"gb2312");        ? >
```

则返回值为 6,将网页字符编码设置为 GBK 或 gb2312 即可获得正确的中文字符串长度。

如果要判断字符串的长度是否要小于某个值,有以下两种方法:

```
if(strlen( $pwd) <6)echo "密码太短";
if(!isset( $pwd{6}))echo "密码太短";        //如果第 7 个字符不存在
```

而第二种方法不需要使用函数,效率更高。

3.3.2　数据类型的转换

PHP 中数据类型的转换有以下两种情况。

1. 自动类型转换

(1) 如果对变量重新赋了不同数据类型的值,则变量的数据类型会自动转换,例如:

```
$a = "Hello";          $a = 12;
```

则变量 $ a 的数据类型就会由字符串型转换成整型。

(2) 如果不同数据类型的变量进行运算操作,则将选用占字节最多的一个运算数的数据类型作为运算结果的数据类型,例如:

```
$a = 1 + 3.14;
$b = 2 + "2.0";
$c = 3 + "php";
var_dump( $a, $b, $c);                    //输出 float(4.14)float(4)int(3)
```

则 $a 的数据类型为浮点型。在第二个赋值表达式中,首先将字符串数据"2.0"转换成浮点型数据 2.0,然后进行加法运算,赋值后 $b 的数据类型为浮点型。在第三个表示式中,首先将字符串数据转换成整型数据 0,然后进行加法运算,赋值后 $c 的数据类型为整型。

2. 强制数据类型转换

利用强制类型转换可以将数据类型转换为指定的数据类型,其语法如下:

(类型名)变量或表达式

其中类型名包括 int,bool,float,double,real,string,array,object,类型名两边的括号一定不能省略。例如:

```
$a = "2.0";              $b = (int) $a;
$c = (array) $a;         print_r( $c);
```

则 $b 将转换成整型,$c 将转换为数组类型(Array([0] =>2.0))。

虽然强制数据类型转换使用起来很方便,但也存在一些问题,例如字符串型转换成整型该如何转换,整型转换成布尔型该如何转换,这些都需要一些明确的规定,PHP 为此提供了相关的转换规定,如表 3-5 所示。

表 3-5 PHP 类型转换的规定

源 类 型	目 的 类 型	转 换 规 则
float	integer	保留整数部分,小数部分无条件舍去
Boolean	integer 或 float	false 转换成 0,true 转换成 1
Boolean	string	false 转换成空字符串"",true 转换成字符串"1"
string	integer	从字符串开头取出整数,开头没有,就是 0。例如字符串"3M"、"8.6uc"、"x5"会转换成整数 3、8、0
string	float	从字符串开头取出浮点数,开头没有,就是 0.0。例如字符串"3M"、"8.6uc"、"x5"会转换成整数 3.0、8.6、0.0
string	Boolean	空字符串""或字符串"0"转换成 false,其他都转换成 true,因此字符串"false"也会转换成 true
integer float	Boolean	0 转换成 false,非零的数都转换成 true
integer float	string	将所有数字转换成字符串,如 12 转换成"12",3.14 转换成"3.14"
integer float Boolean string	array	创建一个新的数组,第一个元素就是该整数、浮点数、布尔值或字符串
array	string	字符串"Array"
object	Boolean	没有成员的对象转换成 false,否则会转换成 true

提示：如果使用 echo 函数输出布尔值：echo true，会输出字符串"1"，echo false，会输出空字符串。因为任何数据类型输出时都将被转换成字符串。

3.4　PHP 的语句

PHP 的语句可分为顺序执行语句、条件控制语句、循环控制语句以及包含语句等。

3.4.1　条件控制语句

在 PHP 中，有 if 语句和 switch 两种条件语句。if 语句又可分为单分支选择 if 语句、双分支选择 if 语句和多分支选择 if 语句三种。

1. 单分支选择 if 语句

一般形式为：

```
if(条件表达式){
    语句块  }
```

它表示当条件表达式成立时(值为 true)，执行“语句块”，例如：

```
if( $sex ==1)echo "尊敬的先生";
```

如果语句块中包含多条语句，则要使用{}将这些语句包含起来，使它们构成一条复合语句，例如：

```
if( $a > $b)  {
    $temp = $a;
    $a = $b;
    $b = $temp;}
```

2. 双分支选择 if…else 语句

一般形式为：

```
if(条件表达式)
    {    语句块 1     }
else
    {    语句块 2     }
```

表示当条件表达式值为 true 时，执行“语句块 1”，否则执行“语句块 2”，例如：

```
if( $a) $a =0; else  $a =1;
```

该语句被称为“开关语句”，即如果 $a 的值为 true 或非零，则让 $a 的值为 0，否则让 $a 的值为 1，因此每执行一次都会使 $a 的值在 0 和 1 之间转换。if($a)是 if($a == true)的简写形式。

3. 多分支选择 if…elseif…else 语句

一般形式为：

```
if(表达式 1)     语句块 1
elseif(表达式 2)   语句块 2
elseif(表达式 3)   语句块 3
⋮
else 语句块 n
```

它首先会判断表达式 1 是否成立，如果成立，则执行语句块 1，执行完后，直接退出该选择结构，不再判断后面的表达式是否成立。如果表达式 1 不成立，则再依次判断表达式 2 ~ 表达式 *n* 是否成立，如果成立，则执行对应的语句块 *i*，如果所有表达式都不成立，则执行 else 后的语句块 *n*。例如要找出三个数中的最大数，程序如下：

```
if( $a < $b) $max = $b;
elseif( $a < $c) $max = $c;
else  $max = $a;
```

说明：

（1）if 语句还可以嵌套使用，也就是说“语句块”中还可以存在 if 语句。

（2）if(条件表达式)后一般没有“;”号，如果有“;”，表示 if 语句的语句块为空语句。

（3）语句块如果是一条语句则后面一定要有“;”，如果语句块是由{}包含的复合语句，则{}后不要有“;”。

4. switch/case 语句

switch 语句是多分支选择 if 语句的另一种形式，两者可互相转换。在要判断的条件有很多种可能的情况下，使用 switch 语句将使多分支选择结构更加清晰，一般形式为：

```
switch(变量或算术表达式)  {
    case(常量 1):    语句块 1
    case(常量 2):    语句块 2
    ⋮
    case(常量 n):    语句块 n
    default: 语句块 n+1
}
```

下面的程序根据时间显示不同的欢迎信息(3-5. php)。

```
<?   $a=date(G);                          //获取当前时间的小时数
$a=floor( $a/3);                          //将小时数除以 3 并取底
switch( $a)  {
    case 2:echo "早上好";break;
    case 3:echo "上午好";break;
    case 4:case 5: echo "下午好";break;
    case 6:echo "晚上好";break;
```

```
    default: echo "该睡觉了";break;
}    ? >
```

说明：

（1）case 语句后不能接表示范围的条件表达式，只能接常量。

（2）各个 case 中的常量必须不相同，如果相同，则满足条件时只会执行前面 case 语句中的内容。

（3）多个 case 可共用一组语句，此时必须写成“case 4：case 5：”的形式，不能写成“case 4，5：”。

（4）每个 case 后一般都要有一条 break 语句，这样执行完该 case 语句后就会跳出分支结构，否则，执行完该 case 语句后还会依次执行下面的 case 语句，直到遇到 break 或执行完。

（5）各个 case 和 default 语句的出现顺序可随意变动。

3.4.2　循环控制语句

循环结构通常用于重复执行一组语句，直到满足循环结束条件时才停止。在 PHP 中，主要有 4 种循环语句，即 for 循环、foreach 循环、while 循环和 do…while 循环。

1. for 循环

for 循环语句是不断地执行循环体中的语句，直到相应条件不满足，并且在每次循环后处理计数器。for 语句的一般形式为：

```
for(初始表达式; 循环条件表达式; 计数器表达式)
    {  循环体语句块 }
```

其执行过程为：①执行初始表达式（通常是给循环变量赋初值）；②判断循环条件表达式是否成立，若成立，则执行循环体，否则跳出循环；③执行一遍循环体语句块；④执行计数器表达式（通常是给循环变量计数）；⑤转到第②步判断是否继续循环。

循环可以嵌套，例如要用 for 循环画金字塔，有下面两种写法，运行效果如图 3-4 所示。

图 3-4　画金字塔程序

① 写法 1（3-6. php）。

```
<div align = "center" >
< ?
for( $i = 0; $i < 5; $i ++){
  for( $j = 0; $j < = $i; $j ++)
    echo "* ";
  echo "< br/ >";
}
? > < /div >
```

② 写法 2(3-7. php)。

```
<div align = "center" >
<?
for( $i = 0; $i < 5; $i ++){
  $a = $a ." * ";
 echo $a ." <br/ >";
}
? >
< /div >
```

提示：在对矩阵进行操作时，通常需要双重循环嵌套。

2. foreach 循环

foreach 语句通常用来对数组或对象中的元素进行遍历操作，例如数组中的元素个数未知，则很适合使用 foreach 语句，其一般形式为：

```
foreach(数组名 as $value)
    {  循环体语句块 }
```

或者

```
foreach(数组名 as $key => $value)
    {  循环体语句块 }
```

foreach 语句遍历数组时首先指向数组中的第一个元素。每次循环时，将当前数组元素值赋给 $ value，将当前数组索引值赋给 $ key，再让数组指针向后移动直到遍历结束。示例程序如下，运行效果如图 3-5 所示。

```
<?                                                    //3 - 8.php
$sports = array("网球","游泳","短跑","柔道");  //定义并初始化一个数组
echo "我校开展的运动项目如下 <br / >";
foreach( $sports as $key => $value)
       echo $key .":". $value . " ";          ? >
```

3. while 循环

while 语句是前测式循环，即是否终止循环的条件判断是在执行循环体之前，因此循环体可能一次都不会执行，其一般形式为：

```
while(条件表达式){
    循环体语句块 }
```

图 3-5　foreach 示例程序

图 3-6　while 语句的应用

例如,要输出一个有三行的 html 表格,程序(3-9. php)如下,运行效果如图 3-6 所示。

```
<table border="1" width="300" align="center">
<?   $i=0;
while($i<3){
    echo "<tr><td>这是第 $i 行</td></tr>";   //输出表格行
    $i++;
}  ?>  </table>
```

4. do…while 循环

do…while 语句是后测式循环,它将条件判断放在循环之后,这就保证了循环体中的语句块至少会被执行一次,在某些时候这是非常有用的,其一般形式为:

```
do {
    循环体语句块 }
while(条件表达式);                                    //注意 while(…)后有;号
```

例如下面的程序会输出段落 <p> 元素一次:

```
<?   $i=0;
do{
    echo "<p>不满足循环条件,仍然会输出一次</p>";
    $i++;    }
while($i>1);    ?>
```

想一想:如果将 while($i>1)改成 while($i>0),程序会循环多少次?

5. break 循环

break 语句用来提前终止循环,它可以出现在 while、do…while、for、foreach 或 switch 语句内部,用来跳出循环语句或 switch 语句。在用“穷举法”解题时,通常找到解后就用 break 终止循环。例如要输出一个字符串,各元素之间用“,”号隔开,最后一个元素后没有“,”号,代码如下(3-10. php):

```
<?   $sports=array("网球","游泳","短跑","柔道");
for($i=0;$i<4;$i++){
    echo $sports[$i];
    if($i==3)break;              //最后一个元素不输出",",换成 continue 试试
    echo ",";
}    ?>
```

输出结果为:

网球,游泳,短跑,柔道

提示:在 PHP 中,break 后还可带参数 *n*,表示跳出 *n* 层循环,如“break 2;”会跳出 2 层循环,而其他语言的 break 语句一般不能带参数,只能跳出最近的一层循环。