第一章 基本语法

第一节 PHP 代码书写

PHP 代码我们以<?php 开始，以?>结束。如：

```
<?php
    echo"我正在学习PHP";
?>
```

在浏览器中将输出：

```
我正在学习 PHP
```

这种风格我们称之为标准风格，我们也可省去“PHP”三个字母，即我们常常说的简短风格。如：

```
<?php
    echo"我正在学习PHP";
?>
```

同样我们在浏览器中可以看到：

```
我正在学习 PHP
```

此外，还有 Script 风格，如：

```
<script language="php">
    echo"我正在学习PHP";
</script>
```

以及 ASP 风格，如：

```
<%
    echo"我正在学习PHP";
%>
```

ASP 风格在默认情况下是被禁止的，如果需要运行，需修改 asp_tags 选项，在 PHP.int 里。

我们一般用标准风格及简短风格，Script 风格及 ASP 风格基本不用，但是，为了达到最好的兼容性，我们推荐使用标准风格，而不使用简短风格。

第二节 PHP 文本注释

在 PHP 中，我们使用“//”来编写单行注释，或者使用 “/* … */ ”来编写多

行注释，也可用“#”来注释，但这个不太常用。

文本的注释要写在代码的上方或右边，不要写在代码的下方。

```
<?php
    //echo() 函数输出一个或多个字符串。
    /*echo() 实际上不是一个函数，因此您无需对其使用括号。
    不过，如果您希望向 echo() 传递一个或多个参数，
    那么使用括号会发生解析错误。*/
    echo"我正在学习PHP";
?>
```

在浏览器中只输出：

```
我正在学习 PHP
```

而注释了的文本却没有显示。

第三节 PHP 输出函数

PHP 输出函数有 echo()函数与 print()函数以及格式化输出函数 printf()函数和 sprintf()函数。

一、echo()函数

echo()函数输出一个或多个字符串，可以用圆括号，也可不用，在实际应用中，我们一般不用圆括号；echo 更象一条语句，无返回值。

```
<?php
    echo("今天天气很好，我们出去玩！")."<br>";
    echo"今天天气很好，我们出去玩！";
?>
```

浏览器中输出为：

```
今天天气很好，我们出去玩！
今天天气很好，我们出去玩！
```

二、print()函数

print()函数输出一个或多个字符串，可以用圆括号，也可不用，在实际应用中，我们一般不用圆括号；print()函数有返回值，其返回值为 1，当其执行失败（比如断线）时返回 flase。

```php
<?php
    print("今天天气很好，我们出去玩！").<br>";
    print"今天天气很好，我们出去玩！"."<br>";
    echo print"今天天气很好，我们出去玩！";
?>
```

浏览器中输出为：

今天天气很好，我们出去玩！
今天天气很好，我们出去玩！
今天天气很好，我们出去玩！1

第一句、第二句都输出了“今天天气很好，我们出去玩!”，第三除了输出了“今天天气很好，我们出去玩!”外，还输出了返回值“1。

echo()函数与 print()函数两者的功能几乎完全一样，有一点不同就是 echo()函数无返回值，print()函数有返回值。且 echo()函数稍快于 print()函数。

三、printf()函数

printf()函数输出格式化的字符串。其中格式化字符串包括两部分内容：一部分是正常字符，这些字符将按原样输出；另一部分是格式化规定字符，以“%”开始，后跟一个或几个规定字符，用来确定输出内容格式。

参量表是需要输出的一系列参数，其个数必须与格式化字符串所说明的输出参数个数一样多，各参数之间用“,”分开，且顺序一一对应，否则将会出现意想不到的错误。

常用类型转换符

%b 整数转二进制

%c 整数转 ASCII 码

%d 整数转有符号十进制

%f 倍精度转浮点

%o 整数转八进制

%s 整数转字符串

%u 整数转无符号十进制

%x 整数转十六进制(小写)

%X 整数转十六进制(大写)

```php
<?php
    $a = "今天";
    $b = 10;
    printf("%s我买了%u本书",$a,$b);
?>
```

浏览器输出的内容是：

```
今天我买了 10 本书
```

printf()函数有返回值，其返回值为字符串的长度。

```php
<?php
    $a = "今天";
    $b = 10;
    echo printf("%s我买了%u本书",$a,$b);
?>
```

此时浏览器输出的内容是：

```
今天我买了 10 本书 16
```

其中“今天我买了 10 本书”为 printf()函数格式化后的字符串，“16”为 printf()函数的返回值——字符串的长度 16，需要通过 echo 才能输出。

四、sprintf()函数

sprintf()函数与 printf()函数类似，printf()函数的返回值是字符串的长度，而 sprintf()函数的返回值侧是字符串的本身。因此，sprintf()函数必须通过 echo 才能输出。

```php
<?php
    $a = "今天";
    $b = 10;
    echo sprintf("%s我买了%u本书",$a,$b);
?>
```

我们在浏览器中可以看到输出为今天我买了 10 本书，如果省略掉了 echo，那么浏览器中输出为空。

sprintf()和 printf()的用法和 C 语言中的 printf()非常相似。我们经常用 sprintf()将

十进制转换为其它进制。如：

```
<?php
    $a = 12;
    echo sprintf ("%b",$a);
?>
```

在浏览器中输出为：

```
1100
```

即将 12 转为二进制为 1100。

第四节 PHP 变量

变量用于存储值，比如数字、文本字符串或数组。一旦设置了某个变量，我们就可以在脚本中重复地使用它。PHP 的变量必须以 $ 符开始，然后再加上变量名。

一、变量的命名

1.变量名必须以字母或者下划线 "_" 开头，后面跟上任意数量的字母、数字或者下划线。

2.变量名不能以数字开头，中间不能有空格及运算符。

3.变量名严格区分大小写，即$UserName 与$username 是不同的变量。

4.为避免命名冲突，不允许使用与 PHP 内置的函数相同的名称。

5.在为变量命名时，尽量使用有意义的字符串。

$name;

$_password;

$book1;

二、变量的赋值

为变量赋值有两种方式：传值赋值和引用赋值，这两种赋值方式在对数据的处理上存在很大差别。

1、传值赋值

这种赋值方式使用“=”直接将一个变量（或表达式）的值赋给变量。使用这种赋值方式，等号两边的变量值互不影响，任何一个变量值的变化都不会影响到另一个变量。从根本上讲，传值赋值是通过在存储区域复制一个变量的副本来实现的。应用传值赋值的示例代码如下。

```php
<?php
    $a = 33;
    $b = $a;
    $b = 44;
    echo "变量a的值为".$a."<br>";
    echo "变量b的值为".$b;
?>
```

在上面的代码中，执行“$a = 33”语句时，系统会在内存中为变量 a 开辟一个存储空间，并将“33”这个数值存储到该存储空间。

执行“$b = $a”语句时，系统会在内存中为变量 b 开辟一个存储空间，并将变量 a 所指向的存储空间的内容复制到变量 b 所指向的存储空间。

执行“$b = 44”语句时，系统将变量 b 所指向的存储空间保存的值更改为“44”，而变量 a 所指向的存储空间保存的值仍然是“33”。

因此，我们在浏览器上看到的内容为：

变量 a 的值为 33

变量 b 的值为 44

2、引用赋值

引用赋值同样也是使用“=”将一个变量的值赋给另一个变量，但是需要在等号右边的变量前面加上一个“&”符号。实际上这种赋值方式并不是真正意义上的赋值，而是一个变量引用另一个变量。在使用引用赋值的时候，两个变量将会指向内存中同一存储空间。因此任何一个变量的变化都会引起另外一个变量的变化。应用引用赋值的示例代码如下。

```php
<?php
    $a = 33;
    $b = &$a;
    $b = 44;
    echo "变量a的值为".$a."<br>";
    echo "变量b的值为".$b;
?>
```

在上面的代码中执行“$a = 33”语句时，对内存进行操作的过程与传值赋值相同，这里就不再介绍了。执行“$b = &$a”语句后，变量 b 将会指向变量 a 所占有的存储空间。

执行“$b = 44”语句后，变量 b 所指向的存储空间保存的值变为“44”。此时由于变量 a 也指向此存储空间，所以变量 a 的值也会变为“44”。

因此，我们在浏览器上看到的内容为：

```
变量 a 的值为 44
变量 b 的值为 44
```

三、变量的作用域

在使用 PHP 语言进行开发的时候，我们几乎可以在任何位置声明变量。但是变量声明位置及声明方式的不同决定了变量作用域的不同。所谓的变量作用域，指的是变量在哪些范围内能被使用，在哪些范围内不能被使用。PHP 中的变量按照作用域的不同可以分为局部变量和全局变量。

1、局部变量

局部变量是声明在某一函数体内的变量，该变量的作用范围仅限于其所在的函数体的内部。如果在该函数体的外部引用这个变量，则系统将会认为引用的是另外一个变量。

应用局部变量的示例代码如下。

```php
<?php
    function local(){
        $a = "“这是内部函数”";//在函数内部声明一个变量a并赋值
        echo "函数内部变量a的值为".$a."<br>";
    }
    local();//调用函数local()，用来打印出变量a的值
    $a = "“这是外部函数”"; //在函数外部再次声明变量a并赋另一个值
    echo "函数外部变量a的值为".$a;
?>
```

在浏览器中输出为：

```
函数内部变量 a 的值为“这是内部函数”
```

函数外部变量 a 的值为“这是外部函数”

2、全局变量

全局变量可以在程序的任何地方被访问，这种变量的作用范围是最广泛的。要将一个变量声明为全局变量，只需在该变量前面加上“global”关键字，不区分大小写，也可以是“GLOBAL”。使用全局变量，我们能够实现在函数内部引用函数外部的参数，或者在函数外部引用函数内部的参数。

应用全局变量的示例代码如下。

在函数内部引用函数外部的参数：

```
<?php
    $a = "“这是外部函数”";//在外部定义一个变量a
    function local(){
        global $a;//将变量a声明为全局变量
        echo "在local函数内部获得变量a的值为".$a."<br>";
    }
    local(); //调用函数local()，用于输出local函数内部变量a的值
?>
```

在浏览器中输出为：

在 local 函数内部获得变量 a 的值为“这是外部函数”

在函数外部引用函数内部的参数：

```
<?php
    function local(){
        global $a;//将变量a声明为全局变量
        $a = "“这是内部函数”";//在内部对变量a进行赋值
    }
    local(); //调用函数local()，用于输出local函数内部变量a的值
    echo "在local函数外部获得变量a的值为".$a;//在函数local外部输出变量
?>
```

在浏览器中输出为：

在 local 函数外部获得变量 a 的值为“这是内部函数”

应用全局变量虽然能够使我们更加方便地操作变量，但是有的时候变量作用域的扩大，会给开发带来麻烦，可能会引发一些预料不到的问题。

将一个变量声明为全局变量，还有另外一种方法，就是利用$GLOBALS[]数组。

3、静态变量

函数执行时所产生的临时变量，在函数结束时就会自动消失。当然，因为程序需要，函数在循环过程中不希望变量在每次执行完函数就消失的话，那么我们就要采用静态变量，静态变量是指用 static 声明的变量，这种变量与局部变量的区别是，当静态变量离开了它的作用范围后，它的值不会自动消亡，而是继续存在，当下次再用到它的时候，可以保留最近一次的值。

应用静态变量的示例代码如下。

```
<?php
    function add()
    {
        static $a = 0;
        $a++;
        echo $a."<br >";
    }
    add ();
    add ();
    add ();
?>
```

在浏览器中输出为：

```
1
2
3
```

这段程序中，主要定义了一个函数 add()，然后分 3 次调用 add()。

如果用局部变量的方式来分工这段代码，3 次的输出应该都是 1。但实际输出却是 1、2 和 3。

这是因为，变量 a 在声明的时候被加上了一个修饰符 static，这就标志着 a 变量

在 add()函数内部就是一个静态变量了，具备记忆自身值的功能，当第一次调用 add 时，a 由于自加变成了 1，这个时候，a 就记住自己不再是 0，而是 1 了，当我们再次调用 add 时，a 再一次自加，由 1 变成了 2，…。由此，我们就可以看出静态变量的特性了。

4、可变变量

可变变量是一种独特的变量，它可以动态的改变一个变量的名称，方法就是在该变量的前面加一个变量符号“$”。

```php
<?php
    $a= 'hello';    //普通变量
    $$a = 'world';  //可变变量，相当于$hello='world';
    echo $a."<br>";
    echo $$a."<br>";
    echo $hello."<br>";
    echo "$a {$$a}"."<br>";
    echo "$a $hello";//这种写法更准确并且会输出同样的结果
?>
```

在浏览器中输出为：

```
hello
world
world
hello world
hello world
```

5、预定义变量

预定义变量又称为超级全局变量数组，是 PHP 系统中自带的变量，不需要开发者重新定义，它可让你的程序设计更加的方便快捷。在 PHP 脚本运行时，PHP 会自动将一些数据放在超级全局数组中。

PHP 预定义变量	
变量	作用
$GLOBALS[]	储存当前脚本中的所有全局变量，其 KEY 为变量名，

	VALUE 为变量值
$_SERVER[]	当前 WEB 服务器变量数组
$_GET[]	存储以 GET 方法提交表单中的数据
$_POST[]	存储以 POST 方法提交表单中的数据
$_COOKIE[]	取得或设置用户浏览器 Cookies 中存储的变量数组
$_FILES[]	存储上传文件提交到当前脚本的数据
$_ENV[]	存储当前 WEB 环境变量
$_REQUEST[]	存储提交表单中所有请求数组，其中包括$_GET、$_POST、$_COOKIE 和$_SESSION 中的所有内容
$_SESSION[]	存储当前脚本的会话变量数组

四、变量的数据类型

数据类型是具有相同特性的一组数据的统称。PHP 早就提供了丰富的数据类型，PHP 5 中又有更多补充。数据类型可以分为 3 类：标量数据类型、复合数据类型和特殊数据类型。

标量类型（四种）：

整型（int，integer）

浮点型（float，double，real）

布尔型（bool，boolean）

字符串（string）

复合类型（两种）：

数组（array）

对象（object）

特殊类型（两种）：

资源（resource）

空值（NULL）

1、整型（integer）

PHP 中的整型指的是不包含小数部分的数据。在 32 位操作系统中，整型数据的有效范围在“-2147483648～+2147483647”之间。整型数据可以用十进制（基数为 10）、八进制（基数为 8，以 0 作为前缀）或十六进制（基数为 16，以 0x

作为前缀）表示，并且可以包含“+”和“-”。整型数据的用法如下面代码所示。

```
<?php
    $a = 100;    //十进制整型数据
    $b = -034;   //八进制整型数据
    $c = 0xBF;   //十六进制整型数据
    echo $a."<br>";
    echo $b."<br>";
    echo $c;
?>
```

在浏览器中输出为：

```
100
-28
191
```

如果给定的数字超出了整型数据规定的范围，则会产生数据溢出。对于这种情况，PHP 会自动将整型数据转化为浮点型数据。

2、浮点型（float）

浮点型数据就是通常所说的实数，可分为单精度浮点型数据和双精度浮点型数据。浮点数主要用于简单整数无法满足的形式，比如长度、重量等数据的表示。浮点型数据的用法如下面代码所示。

```
<?php
    $a = 1.2;
    $b = -0.34;
    $c = 1.8e4;   //该浮点数表示1.8×10^4
    echo $a."<br>";
    echo $b."<br>";
    echo $c;
?>
```

在浏览器中输出为：

```
1.2
-0.34
18000
```

3、布尔型（boolean）

布尔型数据是在 PHP 4 中开始出现的，一个布尔型的数据只有“true”和“false”两种取值，分别对应逻辑“真”与逻辑“假”。布尔型变量的用法如下面代码所示。在使用布尔型数据类型时，“true”和“false”两个取值是不区分大小写的。也就是说“TRUE”和“FALSE”同样是正确的。

```php
<?php
    $a = true;
    $b = false;
    echo $a;
    echo $b;
?>
```

在浏览器中输出为：

```
1
```

当布尔值为“true”时，输出为 1，当布尔值为“false”时，输出为空。

4、字符串（string）

字符串是一个字符的序列。组成字符串的字符是任意的，可以是字母、数字或者符号。在 PHP 中没有对字符串的最大长度进行严格的规定。在 PHP 中定义字符串有 3 种方式：使用单引号（'）定义、使用双引号（"）定义和使用定界符（<<<）定义。下面是一个使用字符串的例子。

```php
<?php
    $var = "中国人";
    echo "我是$var"."<br>";
    echo '我是$var'.'<br>';
    echo "今天天气很好！"."<br>";
    echo '我们去图书馆。'."<br>";
    echo <<<Eof
```

```
                我是一个{$var}
Eof;
?>
```

在浏览器中输出为：

```
我是中国人
我是$var
今天天气很好！
我是中国人
我是$var
今天天气很好！
我们去图书馆。
我是一个中国人
```

php 中单引号和双引号的最大区别就是，双引号比单引号多一步解析过程。双引号会把双引号中的变量及转义字符解析出来。而单引号则不管它的内容是什么，都作为字符串输出。

在双引号中，中文和变量一起使用时，变量最好要用{}括起来，或变量前后的字符串用双引号，再用“.”与变量相连。

```
<?php
    $str = "年轻人";
    echo "我们都是$str，应该多学习。"."<br>";
    echo "我们都是{$str}，应该多学习。"."<br>";
    echo "我们都是".$str."，应该多学习。";
?>
```

在浏览器中输出为：

```
我们都是
我们都是年轻人，应该多学习。
我们都是年轻人，应该多学习。
```

第一句输出因为变量没用{}括起来，或者没有将字符串分开，再用“.”与变量相连，因此变量及其后面的字符串不能输出，第二、三句输出都正常。

在一般情况下，我们尽量使用单引号，因为在理论上，单引号的运行速度要快些，如果遇到有变量及转义字符需要解析时，我们才用双引号。

下面是一些常用的转义字符：

转移序列	描述
\n	换行符
\r	回车符
\t	制表符
\\	反斜线
\$	美元符
\"	双引号

值得注意的是，“\n”，“ \r”和“ \t”三个转义字符在浏览器中不能反应出来，只能在源文件看到。

PHP 定界符的作用就是按照原样，包括换行格式什么的，输出在其内部的东西；在 PHP 定界符中的任何特殊字符都不需要转义；PHP 定界符中的 PHP 变量会被正常的用其值来替换。使用定界符应注意以下几点：

（1）在<<<之后的字符 Eof 是自己定义的，随便什么都是可以的，但是结尾处的字符一定要和他一样，他们是成对出现的；

（2）结尾的 Eof;，一定要另起一行，并且除了 Eof;这个定界符结尾标识之外不能有任何其他字符，前后都不能有，包括空格；

（3）如果在定界符中间出现有 PHP 的变量，你只需要像在其它字符串中输出一样写就行了，变量$var 之所以要用{}括起来是要告诉 PHP 解析器这是一个 PHP 变量，其实不用也是可以的，但是有可能会产生歧义。

5、数组（array）

数组是一系列相关的数据以某种特定的方式进行排列而组成的集合。组成这个集合的各个数据可以是基本数据类型，也可以是复合数据类型；可以是相同的数据类型，也可以是不同的数据类型。

数组里的每一个数据元素都有其唯一的编号，称为索引。索引用于指定数组中特定的数据元素。在有的语言中数组的索引必须是数字编号，而在 PHP 中，索引可以是数字编号，也可以是字符串。

一个简单的 PHP 数组的应用示例代码如下。

```
<?php
    $network = array(1=>"how",2=>are,'three'=>"you");
    echo $network[2];
    echo $network[three];
?>
```

在浏览器中输出为：

```
areyou
```

6、对象（object）

对象是面向对象语言中的一个核心概念，对象就是类的一个实例。在了解对象之前我们先简单介绍一下什么是“类”。在面向对象语言中，人们把各个具体事物的共同特征抽取出来，形成一个一般的概念，也就构成了一个“类”。

在 PHP 中类的定义方式如下。

```
class  类名 {
    类里包含的内容;
}
```

在 PHP 中，通过“new”关键字来实例化一个类并得到该类的一个对象。类和对象的应用示例如下面代码所示。

```
<?php
    class Book {
        function getBookName($book_name){
            return $book_name;
        }
    }
    $book1 = new Book(); //实例化一个Book类的对象book1
    echo $book1->getBookName("PHP")."<br>";
    $book2 = new Book(); //实例化一个Book类的对象book2
    echo $book2->getBookName("JSP");
?>
```

在浏览器中输出为：

```
PHP
JSP
```

7、资源（resource）

资源是 PHP 提供的一种特殊数据类型，该数据类型用于表示一个 PHP 的外部资源，比如一个数据库的访问操作，或者一个网络流的处理等。虽然资源也是一种数据类型，但是我们却不能直接对它进行操作。PHP 提供了一些特定的函数，用于建立和使用资源。比如“mysql_connect()”函数用于建立一个 MySQL 数据的连接，“fopen()”函数用于打开一个文件等。

应用资源数据类型的示例代码如下。

```php
<?php
    $cn = mysql_connect('localhost','root');
    echo get_resource_type($cn)."<br>";
    $fp = fopen("foo","w");
    echo get_resource_type($fp);
?>
```

在浏览器中输出为：

```
mysql link
stream
```

8、空值（NULL）

NULL 是 PHP 4 开始引入的一个特殊的数据类型，这种数据类型只有一个值 NULL。在 PHP 中，如果变量满足以下几种情况，那么该变量的值就为 NULL。

变量未被赋予任何值。

变量被赋值为 NULL。

被 unset()函数处理后的变量。

下面是使用 NULL 数据类型的一个示例。

```php
<?php
    $a ;          //变量$a未被赋予任何值，$a的值为NULL
    $b = NULL ; //变量$b被赋值为NULL
```

```
    $c = 1 ;
    unset($c);   //使用unset()函数处理后，$c的值为NULL
?>
```

五、变量类型的转换

PHP 中的类型转换包括两种方式，即自动类型转换和强制类型转换。下面我们分别介绍这两种类型转换的实现方式及应用过程。

1、自动类型转换

自动类型转换是指，在定义变量时不需要指定变量的数据类型，PHP 会根据引用变量的具体应用环境将变量转换为合适的数据类型。

如果所有运算数都是数字，则将选取占用字节最长的一种运算数的数据类型作为基准数据类型；如果运算数为字符串，则将该字符串转型为数字然后再进行求值运算。字符串转换为数字的规定为如果字符串以数字开头，则只取数字部分而去除数字后面部分，根据数字部分构成决定转型为整型数据还是浮点型数据；如果字符串以字母开头，则直接将字符串转换为 0。

```
<?php
    $a = 1 + 1.23;
    $b = 2 + "3.14";
    $c = 3 + "abc";
    echo $a."<br>";
    echo $b."<br>";
    echo $c."<br>";
?>
```

在浏览器中输出为：

```
2.23
5.14
3
```

在第 1 个赋值运算式中，运算数包含了整型数字“1”和浮点型数字“1.23”， 首先取浮点型数据类型作为基准数据类型。赋值后变量 a 的数据类型为浮点型。

在第 2 个赋值运算式中，运算数包含了整型数字“2”和字符串型数据“3.14”，

首先将字符串转换为浮点型数据“3.14”，然后进行加法运算。赋值后变量 b 的数据类型为浮点型。

在第 3 个赋值运算式中，运算数包含了整型数字“3”和字符串型数据“abc”，首先将字符串转换为整型数字 0，然后进行加法运算。赋值后变量 c 的数据类型为整型。

2、强制类型转换

强制类型转换允许我们手动将变量的数据类型转换成为指定的数据类型。PHP 强制类型转换与 C 语言或者 Java 语言中的类型转换相似，都是通过在变量前面加上一个小括号，并把目标数据类型填写在括号中来实现的。

在 PHP 中强制类型转换的具体实现方式如下表所示。

PHP 强制类型转换的实现方式		
转换格式	转换结果	实现方式
(int),(integer)	将其他数据类型 强制转换为整型	$a = "3"; $b = (int)$a; // 也可写为$b = (integer)$a;
(bool),(boolean)	将其他数据类型 强制转换为布尔型	$a = "3"; $b = (bool)$a; // 也可写为$b = (boolean)$a;
(float),(double),(real)	将其他数据类型 强制转换为浮点型	$a = "3"; $b = (float)$a; $c = (double)$a; $d = (real)$a;
(string)	将其他数据类型 强制转换为字符串	$a = 3; $b = (string)$a;
(array)	将其他数据类型 强制转换为数组	$a = "3"; $b = (array)$a;
(object)	将其他数据类型 强制转换为对象	$a = "3"; $b = (object)$a;

虽然 PHP 提供了比较宽泛的类型转换机制，为开发者提供了很大便利，但同时

也存在着一些问题，比如将字符串型数据转换为整型数据该如何转换、将整型数据转换为布尔型数据该如何转换等。如果没有对上述类似的情形做出明确规定，那么在处理类型转换问题时就会出现一些问题，幸运的是 PHP 为我们提供了相关的转换规定。

1）、其他数据类型转换为整型的规则

其他数据类型转换为整型		
原类型	目标类型	转换规则
浮点型	整型	向下取整，即不会四舍五入而是直接去掉浮点型数据小数点后边的部分，只保留整数部分
布尔型	整型	TRUE 转换成整型数字“1”，false 转换成整型数字“0”
字符串	整型	（1）字符串为纯整型数字，转换成相应的整型数字 （2）字符串为带小数点的数字，转换时去除小数点后面的部分，保留整数部分 （3）字符串以整型数字开头，转换时去除整型数字后面的部分，然后按照规则 1 进行处理 （4）字符串以带小数点的数字开头，转换时去除小数点后面的部分，然后按规则 2 进行处理 （5）字符串内容以非数字开头，直接转换为 0

将其他数据类型转换为整型的示例代码如下所示。

```
<?php
    $a = "123";
    $b = "123sunyang";
    $c = "2.32";
    $d = "2.32abc";
    $e = "sunyang123";
    $f = TRUE;
    $g = FALSE;
    $h = 3.1415926;
    echo (int)$a."<br>";
```

```
    echo (int)$b."<br>";
    echo (int)$c."<br>";
    echo (int)$d."<br>";
    echo (int)$e."<br>";
    echo (int)$f."<br>";
    echo (int)$g."<br>";
    echo (int)$h."<br>";
?>
```

在浏览器中输出为：

```
123
123
2
2
0
1
0
3
```

浮点型数据向整型数据转换的时候，需要注意以下两种情况。

如果几个浮点型数据相乘，应将大于 1 的数放在最前面，并将整个式了括起来，不然的话容易出错。以下示例代码中，第一及第四个输出正确，其他三个输出都出现了错误。

```
<?php
    echo (int)(46.86*0.26*0.74)."<br>";//46.86*0.26*0.74=9.015864
    echo (int)46.86*0.26*0.74."<br>";//其结果为46*0.26*0.74=8.8504
    echo (int)0.26*0.74*46.86,"< br>";
    echo (int)(100*0.1*0.7)."<br>";
    echo (int)(0.1*0.7*100);
?>
```

在浏览器中输出为：

```
9
8.8504
0
7
6
```

如果浮点型数据相除时，也应将整外除式括起来，以免出现错误。以下示例代码中，第一及第三个输出正确，其他两个输出都出现了错误。

```php
<?php
    echo (int)(7.8/3.2)."<br>";//7.8/3.2=2.4375
    echo (int)7.8/3.2."<br>";//其结果为7/3.2=2.1875
    echo (int)(3.2/7.8)."<br>";//3.2/7.8=0.410256410256
    echo (int)3.2/7.8;//其结果为3/7.8=0.384615384615
?>
```

在浏览器中输出为：

```
2
2.1875
0
0.384615384615
```

2）、其他数据类型转换为浮点型的规则

其他数据类型转换为浮点型		
原类型	目标类型	转换规则
整型	浮点型	将整型数据直接转换为浮点型，数值保持不变
布尔型	浮点型	TRUE 转换成浮点型数字“1”，false 转换成浮点型数字“0”
字符串	浮点型	（1）字符串为整型数字，直接转换成相应的浮点型数字 （2）字符串以数字开头，转换时去除数字后面的部分，然后按照规则 1 进行处理 （3）字符串以带小数点的数字开头，转换时直接去除数字后面的部分，只保留数字部分

		（4）字符串以非数字内容开头，直接转换为 0

将其他数据类型转换为浮点型的示例代码如下。

```
<?php
    $a = "123";
    $b = "123sunyang";
    $c = "2.32";
    $d = "2.32abc";
    $e = "sunyang123";
    $f = TRUE;
    $g = FALSE;
    $h = 1234;
    echo (float)$a."<br>";
    echo (float)$b."<br>";
    echo (float)$c."<br>";
    echo (float)$d."<br>";
    echo (float)$e."<br>";
    echo (float)$f."<br>";
    echo (float)$g."<br>";
    echo (float)$h."<br>";
?>
```

在浏览器中输出为：

```
123
123
2.32
2.32
0
1
0
1234
```

3)、其他数据类型转换为布尔型的规则

其他数据类型转换为布尔型		
原类型	目标类型	转换规则
整型	布尔型	0 转换为 false，非零的其他整型数字转换为 true
浮点型	布尔型	0.0 转换为 false，非零的其他浮点型数字转换为 true
字符串	布尔型	空字符串或字符串内容为零转换为 false，其他字符串转换为 true
NULL	布尔型	直接转换为 false
数组	布尔型	空数组转换为 false，非空数组转换为 true

将其他数据类型转换为布尔型的示例代码如下所示。

```php
<?php
    $a = 0;
    $b = 123;
    $c = 0.0;
    $d = 3.14;
    $e = "";
    $f = "0";
    $g = "TRUE";
    $h = array();
    $i = array("a","b","c");
    $j = NULL;
    echo var_dump((boolean)$a)."<br>";
    echo var_dump((boolean)$b)."<br>";
    echo var_dump((boolean)$c)."<br>";
    echo var_dump((boolean)$d)."<br>";
    echo var_dump((boolean)$e)."<br>";
    echo var_dump((boolean)$f)."<br>";
    echo var_dump((boolean)$g)."<br>";
    echo var_dump((boolean)$h)."<br>";
```

```
    echo var_dump((boolean)$i)."<br>";
    echo var_dump((boolean)$j);
?>
```

在浏览器中输出为：

```
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
```

4）、其他数据类型转换为字符串的规则

其他数据类型转换为字符串		
原类型	目标类型	转换规则
整型	字符串	转换时直接在整型两边加上双引号作为转换后的结果
浮点型	字符串	转换时直接在浮点型两边加上双引号作为转换后的结果
布尔型	字符串	true 转换为字符串“1”，false 转换为字符串“0”
数组	字符串	直接转换为字符串“Array”
对象	字符串	直接转换为字符串“Object”
NULL	字符串	直接转换为空字符串

将其他数据类型转换为字符串的示例代码如下。

```
<?php
    $a = 123;
    $b = 3.14;
    $c = TRUE;
    $d = FALSE;
```

```php
    $e = array("abc");
    $f = NULL;
    echo (string)$a."<br>";
    echo (string)$b."<br>";
    echo (string)$c."<br>";
    echo (string)$d."<br>";
    echo (string)$e."<br>";
    echo (string)$f;
?>
```

在浏览器中输出为：

```
123
3.14
1

Array
```

5）、其他数据类型转换为数组的规则

其他数据类型转换为数组		
原类型	目标类型	转换规则
整型 浮点型 布尔型 字符串	数组	将这几个数据类型强制转换为数组时， 得到的数组只包含一个数据元素， 该数据就是未转换前的数据， 并且该数据的数据类型也与未转换前相同
对象	数组	转换时将对象的成员变量的名称作为各数组元素的 key， 而转换后数组每个 key 的 value 都为空 （1）如果成员变量为私有的（private）， 则转换后 key 的名称为“类名+成员变量名” （2）如果成员变量为公有的（public）， 则转换后 key 的名称为“成员变量名”

		（3）如果成员变量为受保护的（protected）， 则转换后 key 的名称为“*+成员变量名”
NULL	数组	直接转换为一个空数组

将其他数据类型转换为数组的示例代码如下。

```php
<?php
    $a = 123;
    $b = 3.14;
    $c = TRUE;
    $d = "Hello";
    class A {
        private $private;
        public $public;
        protected $protected;
    }
    $e = new A();
    $f = NULL;
    echo var_dump((array)$a)."<br>";
    echo var_dump((array)$b)."<br>";
    echo var_dump((array)$c)."<br>";
    echo var_dump((array)$d)."<br>";
    echo var_dump((array)$e)."<br>";
    echo var_dump((array)$f);
?>
```

在浏览器中输出为：

```
array(1) { [0]=> int(123) }
array(1) { [0]=> float(3.14) }
array(1) { [0]=> bool(true) }
array(1) { [0]=> string(5) "Hello" }
array(3) { ["Aprivate"]=> NULL    ["public"]=> NULL    ["*protected"]=> NULL }
```

```
array(0) { }
```

6）、其他数据类型转换为对象的规则

其他数据类型转换为对象		
原类型	目标类型	转换规则
整型 浮点型 布尔型 字符串	对象	将其他类型变量转换为对象时， 将会新建一个名为“scalar”的属性， 并将原变量的值存储在这个属性中
数组	对象	将数组转换为对象时，数组的 key 作为对 象成员变量的名称，对应各个 key 的 value 作为对象成员变量保存的值
NULL	对象	直接转换为一个空对象

将其他数据类型转换为对象的示例代码如下。

```php
<?php
    $a = 123;
    $b = 3.14;
    $c = TRUE;
    $d = "Hello";
    $e = array("a"=>aaa,"b"=>bbb,"c"=>ccc);
    $f = NULL;
    echo var_dump((object)$a)."<br>";
    echo var_dump((object)$b)."<br>";
    echo var_dump((object)$c)."<br>";
    echo var_dump((object)$d)."<br>";
    echo var_dump((object)$e)."<br>";
    echo var_dump((object)$f);
?>
```

在浏览器中输出为：

```
object(stdClass)#1 (1) { ["scalar"]=> int(123) }
```

```
object(stdClass)#1 (1) { ["scalar"]=> float(3.14) }
object(stdClass)#1 (1) { ["scalar"]=> bool(true) }
object(stdClass)#1 (1) { ["scalar"]=> string(5) "Hello" }
object(stdClass)#1 (3) { ["a"]=> string(3) "aaa"  ["b"]=> string(3) "bbb"  ["c"]=>
string(3) "ccc" }
object(stdClass)#1 (0) { }
```

六、变量的常用函数

1、变量转换函数

在 PHP 强制转换中，除了上述方法外，还可应用函数进行转换，常用的函数有以下几种。

1）、settype()函数

settype()函数将变量设置为指定类型，当某个变量用 settype()函数设定后，该变量的类型就发生改变，其语法如下：

```
bool settype (mixed $var, string $type)
```

将变量 var 的类型设置成 type。type 的可能值为：“boolean”(或为“bool”)，“integer”(或为“int”)，“float”，“string”，“array”，“object”，“null”。如果成功则返回 TRUE，失败则返回 FALSE。

使用settype()函数的示例代码如下。

```php
<?php
    $a = "5.6kg";
    $b = true;
    settype($a,"float")."<br>";
    settype($b,"string")."<br>";
    echo $a."<br>";
    echo $b;
?>
```

在浏览器中输出为：

```
5.6
1
```

可以看出，原来为字符串的$a 经 settype 设置后，转换为浮点数“5.6”。原来为布尔值的$b 经 settype 设置后，true 转换为字符串“1”。

2）、intval()函数、floatval()函数、strval()函数

这三个函数是将原变量通过转换后得到新类型的新变量，原变量的类型和值都不变，括号中放入原变量。

使用intval()函数、floatval()函数、strval()函数的示例代码如下。

```
<?php
    $a = "5.6kg";
    $b = 2001;
    $c = intval($a);
    $d = floatval($a);
    $e = strval($b);
    echo $a."<br>";
    echo $b."<br>";
    echo $c."<br>";
    echo $d."<br>";
    echo $e;
?>
```

在浏览器中输出为：

```
5.6kg
2001
5
5.6
2001
```

可以看出，原变量 a，b 并没有改变，变量 c 为整数 5，变量 d 为浮点数 5.6，变量 e 为字符串 2001，而原变量 b 为整数 2001。

2、变量检查函数

1）、isset()函数

isset()函数用于检查某个变量是否存在，如果存在则返回 TRUE，否则返回

FALSE。

使用isset()函数的示例代码如下。

```
<?php
    $a = "2001年";
    $c = 3.14;
    echo isset($a)."<br>";
    echo isset($b)."<br>";
    echo isset($c);
?>
```

在浏览器中输出为：

```
1


1
```

因$a，$c 真实存在，它返回布尔值 TRUE，在浏览器是显示为“1”，而$b 并不存在，它返回布尔值 FALSE，在浏览器是显示为空。

1）、empty ()函数

isset()函数用于检查某个变量的值是否为空（""、0、"0"、NULL、FALSE、array()、var $var 以及没有任何属性的对象都将被认为是空），如果为空则返回 TRUE，否则返回 FALSE。

使用empty ()函数的示例代码如下。

```
<?php
    $a = "";
    $b = 3.14;
    $c = 0;
    echo empty($a)."<br>";
    echo empty($b)."<br>";
    echo empty($c);
?>
```

在浏览器中输出为：

```
1


1
```

因$a，$c 为空，它返回布尔值 TRUE，在浏览器是显示为“1”，而$b 并不为空，它返回布尔值 FALSE，在浏览器是显示为空。

3、变量判断函数

PHP 中有一些函数可以判断变量的类型，下面将一些常用的变量判断函数总结如下。

判断变量类型的函数		
函数名	作用	判断结果
is_int() is_integer()	检测变量是否是整数	若变量为整数类型则返回 true，否则返回 false
is_float() is_double()	检测变量是否是浮点型	若变量为浮点型则返回 true，否则返回 false
is_bool()	检测变量是否是布尔型	若变量为布尔型则返回 true，否则返回 false
is_string()	检测变量是否是字符串	若变量为字符串则返回 true，否则返回 false
is_array ()	检测变量是否是数组	若变量为数组则返回 true，否则返回 false
is_object()	检测变量是否是一个对象	若变量为对象则返回 true，否则返回 false
is_resource()	检测变量是否为资源类型	若变量为资源类型则返回 true，否则返回 false
is_null()	检测变量是否为 NULL	若变量为 NULL 则返回 true，否则返回 false

使用判断变量类型函数的示例代码如下。

```
<?php
    $a = "8";
```

```
    $b = 3.14;
    $c = 9;
    $d = array(2,4,6);
    echo is_string($a).'<br>';
    echo is_float($b).'<br>';
    echo is_ string ($c).'<br>';
    echo is_array($d);
?>
```

在浏览器中输出为：

```
1
1

1
```

4、变量获取函数

1）、gettype()函数

本函数用来取得变量的类型。返回的类型字符串可能为下列字符串其中之一：boolean、integer、double、string、array、object、resource、NULL、unknown type。

使用gettype ()函数的示例代码如下。

```
<?php
    $a = "大家好！ ";
    $b = 3.14;
    $c = 9;
    $d = array(2,4,6);
    echo gettype($a).'<br>';
    echo gettype($b).'<br>';
    echo gettype($c).'<br>';
    echo gettype($d);
?>
```

在浏览器中输出为：

```
string
double
integer
array
```

我们一般不要使用 gettype()来测试某种类型，因为其返回的字符串在未来的版本中可能需要改变。此外，由于包含了字符串的比较，它的运行也是较慢的。使用 is_* 函数代替。

2）、var_dump()函数

var_dump()函数打印变量的相关信息，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。数组将递归展开值，通过缩进显示其结构。

使用var_dump()函数的示例代码如下。

```php
<?php
    $a = "大家好！ ";
    $b = 3.14;
    $c = 9;
    $d = array(2,4,6);
    echo var_dump($a).'<br>';
    echo var_dump($b).'<br>';
    echo var_dump($c).'<br>';
    echo var_dump($d);
?>
```

在浏览器中输出为：

```
string(8) "大家好！ "
float(3.14)
int(9)
array(3) { [0]=> int(2) [1]=> int(4) [2]=> int(6) }
```

3）、var_export()函数

var_export()函数输出或返回一个变量的字符串表示，此函数返回关于传递给该

函数的变量的结构信息，它和 var_dump()类似，不同的是其返回的表示是合法的 PHP 代码。您可以通过将函数的第二个参数设置为 TRUE，从而返回变量的表示。

使用 var_export()函数的示例代码如下。

```
<?php
    $a = "8";
    $b = 3.14;
    $c = 9;
    $d = array(2,4,6);
    echo var_export($a).'<br>';
    echo var_export($b).'<br>';
    echo var_export($c).'<br>';
    echo var_export($d);
?>
```

在浏览器中输出为：

```
'大家好！'
3.14
9
array ( 0 => 2, 1 => 4, 2 => 6, )
```

一般在调试数组的时候，多用 var_export()，因为 var_dump()没有格式，而 var_export()是有换行的，看起来比较舒服一些。而在调试单个变量的时候多用 var_dump()，因为 var_dump()可以打印出变量类型和长度。

4、变量销毁函数

unset()函数

php 中 unset()函数是用来销毁变量的，但很多时候，这个函数只把变量给销毁了，内存中存放的该变量的值仍然没有销毁，也就是没能达到我们想要的释放内存的效果。

如果在函数中 unset()一个通过引用传递的变量，则只是局部变量被销毁，而在调用环境中的变量将保持调用 unset()之前一样的值。

使用 unset ()函数的示例代码如下。

```php
<?php
    $a = "大家好！ ";
    $b = 3.14;
    $c = 9;
    $d = array(2,4,6);
    $e = true;
    unset($b);
    unset($d);
    echo ($a).'<br>';
    echo ($b).'<br>';
    echo ($c).'<br>';
    echo ($d).'<br>';
    echo ($e);
?>
```

在浏览器中输出为：

```
大家好！


9


1
```

变量 b 和 d 被销毁，所以没有被输出。

第五节 PHP 常量

一、自定义常量

PHP 中使用 define()函数定义常量。define(常量名,常量值)，常量命名方法与变量命名相同，以字母或下划线开头，常量名称区分大小写，但按照惯例常量名称全部大写，如果设置为 true 则不区分大小写，如果设置为 false 则区分大小写，如果没有设置该参数，则取默认值 false。一个常量一旦被定义，就不能再改变或者取消定义。

```php
<?php
    define("CONSTANT", "你好！");
    echo CONSTANT."<br>";
    define("CONSTANT", "你在干什么？");
    echo CONSTANT."<br>";
    echo Constant;
?>
```

在浏览器中输出为：

```
你好！
你好！
Constant
```

我们在 PHP 中定义了一个常量名“CONSTANT”，并给予值“你好！”，因此在浏览器中输出了“你好！”，而我们又给常量“CONSTANT”另一值 “你在干什么？”，因常量定义后其值不能被改变，因此浏览器中第二个输出仍为“你好！”，而不是“你在干什么？”。我们将“echo CONSTANT”改成“echo Constant”，因常量区分大小写，“Constant”并不是常量“CONSTANT”，因此“Constant”被当作内容输出来，即第三个输出为“Constant”。

二、预定义常量

与预定义变量一样，PHP 也提供了一些默认的预定义常量供使用。在程序中可以随时应用这些预定义常量，但是我们不能任意更改这些常量的值。PHP 中常用的一些默认预定义常量及其作用如下表所示。

预定义常量	
常量	作用
__FILE__	存储当前脚本的绝对路径及文件名称
__LINE__	存储该常量所在的行号
__FUNCTION__	存储该常量所在的函数名称
__CLASS__	存储该常量所在的类的名称
__METHOD__	存储该常量所在的类的方法名称
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

第六节 运算符

一个复杂的 PHP 程序往往是由大量的表达式所构成的，而运算符则是表达式的核心，也称作操作符。只有掌握了 PHP 表达式和运算符的用法，才能够更好地使用 PHP 语言进行开发设计。PHP 中常用的运算符包括算术运算符、赋值运算符、比较运算符、逻辑运算符、位运算符、字符串运算符和数组运算符，下面将分别介绍。

一、算术运算符

算术运算符号，就是用来处理四则运算的符号，这是最简单，也最常用的符号，尤其是数字的处理，几乎都会使用到算术运算符，其中取模就是取余数的意思。PHP 提供的算术运算符及其作用如下表所示。

算术运算符		
算术运算符	名称	应用格式
+	加法运算符	$a + $b
-	减法运算符	$a - $b
*	乘法运算符	$a * $b
/	除法运算符	$a / $b
%	取模运算符	$a % $b

使用算术运算符的示例代码如下。

```
<?php
    $a = 8;
    $b = 3;
    echo $a+$b."<br>";
    echo $a-$b."<br>";
    echo $a*$b."<br>";
    echo $a/$b."<br>";
    echo $a%$b;
?>
```

在浏览器中输出为：

```
11
5
24
2.66666666667
2
```

二、递增 / 递减运算符

递增 / 递减运算符是可以对操作系统（可以是数字或字符）进行递增、递减操作的一种运算符。PHP 提供的递增 / 递减运算符及其作用如下表所示。

递增 / 递减运算符		
示例	名称	说明
$i++	后加	返回$i，然后将$i 的值加 1
++$i	前加	$i 的值加 1，然后返回$i
$i--	后减	返回$i，然后将$i 的值减 1
--$i	前减	$i 的值减 1，然后返回$i

使用递增 / 递减运算符的示例代码如下。

```php
<?php
    $a = 8;
    $b = 8;
    $c = 3;
    $d = 3;
    echo $a++."<br>";
    echo ++$b."<br>";
    echo $c--."<br>";
    echo --$d;
?>
```

在浏览器中输出为：

```
8
9
3
```

2

三、赋值运算符

基本的赋值运算符是“=”。它并不是我们一直理解的“等于”号。它实际上意味着把右边表达式的值赋给左边的变量。如$a=3，并不是$a 等于 3，而是将整数 3 赋给$a。然而在 PHP 中不仅仅只有这一种赋值运算符，PHP 提供的赋值运算符及其用法如下表所示。

赋值运算符		
赋值运算符	用法	等价格式
=	$a = 10	$a = 10
+=	$a += 10	$a = $a + 10
-=	$a -= 10	$a = $a - 10
*=	$a *= 10	$a = $a* 10
/=	$a /= 10	$a = $a / 10
%=	$a %= 10	$a = $a % 10
.=	$a .= “abc”	$a = $a.”abc”

使用赋值运算符的示例代码如下。

```
<?php
    $a = 6;
    $b = 8;
    $c = 7;
    $d = 5;
    $e = 4;
    $f = "大家";
    echo($a+=3)."<br>";
    echo($b-=3)."<br>";
    echo($c*=3)."<br>";
    echo($d/=3)."<br>";
    echo($e%=3)."<br>";
    echo($f.="好!");
```

```
?>
```

在浏览器中输出为：

```
9
5
21
1.66666666667
1
大家好!
```

四、比较运算符

比较运算符也称条件运算符或关系运算符，用于比较两个数据的值并返回一个布尔类型的结果。PHP 提供的比较运算符及其用法如下表所示。

比较运算符			
比较运算符	名称	用法	说明
==	等于	$a == $b	如果变量 a 等于变量 b，则返回 true
===	全等	$a === $b	如果变量 a 等于变量 b，并且它们的数据类型也相同，则返回 true
!=或<>	不等	$a != $b 或$a<>$b	如果变量 a 不等于变量 b，则返回 true
!==	非全等	$a !== $b	如果变量 a 不等于变量 b，或者它们的数据类型不同，则返回 true
<	小与	$a < $b	如果变量 a 小于变量 b，则返回 true
>	大于	$a > $b	如果变量 a 大于变量 b，则返回 true
<=	小于等于	$a <= $b	如果变量 a 小于或等于变量 b，则返回 true
>=	大于等于	$a>=$b	如果变量 a 大于或等于变量 b，则返回 true

使用比较运算符的示例代码如下。

```
<?php
    $a = 5;
    $b = 3;
    $c = "5";
```

```php
    $d =5.0;
    echo var_dump($a==$b)."<br>";
    echo var_dump($c==$d)."<br>";
    echo var_dump($a===$c)."<br>";
    echo var_dump($a!=$b)."<br>";
    echo var_dump($a!=$c)."<br>";
    echo var_dump($a!==$d)."<br>";
    echo var_dump($a<$b)."<br>";
    echo var_dump($a>$b)."<br>";
    echo var_dump($a<=$b)."<br>";
    echo var_dump($a>=$b);
?>
```

在浏览器中输出为：

```
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
```

五、逻辑运算符

逻辑运算符用于处理逻辑运算操作，只能操作布尔型值。PHP 提供的逻辑运算符及其用法如下表所示。

逻辑运算符			
逻辑运算符	名称	用法	说明
and 或&&	逻辑与	$a and $b	如果$a 和$b 两个都为 true 时返回 true

		或 $a&&$b	
or 或||	逻辑或	$a or $b 或$a||$b	如果$a 和$b 任何一个为 true 时返回 true
xor	逻辑异或	$a xor $b	如果$a 和$b 只有一个为 true 时返回 true
!	逻辑非	!$a	如果$a 为 false 时返回 true

使用逻辑运算符的示例代码如下。

```php
<?php
    $a = true;
    $b = true;
    $c = false;
    echo var_dump($a&&$b)."<br>";
    echo var_dump($a&&$c)."<br>";
    echo var_dump($a||$b)."<br>";
    echo var_dump($a||$c)."<br>";
    echo var_dump($a xor $b)."<br>";
    echo var_dump($a xor $c)."<br>";
    echo var_dump(!$a)."<br>";
    echo var_dump(!$c);
?>
```

在浏览器中输出为：

```
bool(true)
bool(false)
bool(true)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
```

六、位运算符

位运算符主要应用于整型数据的运算过程。当表达式包含位运算符时，运算时会先将各个整型运算数转换为相应的二进制格式，然后再进行位运算。PHP 提供的位运算符及其用法如下表所示。

位运算符			
位运算符	名称	用法	说明
&	与操作符	$a & $b	将在$a 和$b 中都为 1 的位设为 1
|	或操作符	$a | $b	将在$a 或者$b 中为 1 的位设为 1
^	异或操作符	$a ^ $b	将在$a 和$b 中不同的位设为 1
~	非操作符	~$a	将$a 中为 0 的位设为 1，1 的位设为 0
<<	左移操作符	$a << 2	将$a 中的位向左移动 2 次，空出的位置用 0（每一次移动都表示“乘以 2”）
>>	右移操作符	$a >> 2	将$a 中的位向右移动 2 次，多出的位置截掉（每一次移动都表示“除以 2”）

使用位运算符的示例代码如下。

```php
<?php
    $a = 7;   //二进制为00000111
    $b = 2;   //二进制为00000010
    echo ($a&$b)."<br>"; //与操作后为00000010，转十进制为2
    echo ($a|$b)."<br>"; //或操作后为00000111，转十进制为7
    echo ($a^$b)."<br>"; //异或操作后为00000101，转十进制为5
    echo (~$a)."<br>";   //非操作后为11111000，转十进制为-8
    echo ($a<<$b)."<br>"; //向左位移2个单位后为00011100，转十进制为28
    echo ($a>>$b)."<br>"; //向右位移2个单位后为00000001，转十进制为1
?>
```

在浏览器中输出为：

```
2
7
5
```

```
-8
28
1
```

其他都好理解，对于非操作后为 11111000，转十进制为什么-8，做一下解释。

电脑中一般 32 位，为了方便，我们按 8 位说明（8 的十进制为 1000，其 32 位为 00000000000000000000000000001000，其 8 位为 00001000）。当我们指定一个数是无符号类型时，那么其最高位的 1 或 0，和其它位一样，用来表示该数的大小。当我们指定一个数是有符号类型时，最高数称为“符号位”。为 1 时，表示该数为负值，为 0 时表示为正值。

负数如何转换成二进制？负数转为二进制的步骤为：第一步，求出其正数的二进制，如-8 的正数（8）的二进制为 00001000。第二步，求出其反码，即 1 变 0，0 变 1，00001000 的反码是 11110111。第三步，得出其补码，即反码加 1，要记住逢 2 进 1，11110111 的补码为 11111000，因此-8 的二进制为 11111000。因为我们讲的是 8 位，而实际上是 32 位，前 24 位全都是 1，上面讲的 8 的二进制 00001000 前 24 位都是 0。

为什么 11111000 是-8 而不是 248？我们将 248 转为二进制是 11111000，但其前 24 位都为 0，而-8 的二进制前 24 位都为 1。如果单从 8 位来讲，248 的二进制是无符号类型，它没有负数，取值范围为 0～255（00000000～11111111）共 256 个数，而-8 的二进制是有符号类型，取值范围为-128～127（负数 10000000～11111111，00000000，正数 00000001～01111111）共 256 个数。

在 32 位操作系统中使用位运算符编程的时候，右移不要超过 32 位，左移结果不要超过 32 位，否则会发生数据溢出。

如果在开发过程中一定要使用位运算符，则建议开发人员保证所有参与位运算的数据都为整型数据，否则运算结果可能产生错误。

位运算符也可用于包含字符串的表达式，但是这种情况很少见。

七、字符串运算符

字符串运算符也称连接运算符，用于处理字符串的相关操作。在 PHP 中提供了两个字符串运算符。第一个是连接运算符（“.”），它返回其左右参数连接后的字符串。第二个是连接赋值运算符（“.=”），它将右边参数附加到左边的参数后。

例如下面的例子。

```
<?php
    $a = "今天";
    $b = $a . "是星期一，";
    echo $b."<br>";
    $c = "明天";
    $c .= "是星期二。";
    echo ($c);
?>
```

在浏览器中输出为：

```
今天是星期一，
明天是星期二。
```

八、数组运算符

数组运算符应用于数组的一些相关操作。PHP 提供的数组运算符及其用法如下表所示。

数组运算符			
数组运算符	名称	用法	说明
+	联合	$a+$b	$a 与$b 保存的数组联合
==	相等	$a==$b	如果$a 与$b 保存的数组具有相同键值，则返回 true
===	全等	$a===$b	如果$a 与$b 保存的数组具有相同键值，且顺序和数据类型一致则返回 true
!=或<>	不等	$a!=$b 或 $a<>$b	如果$a 与$b 保存的数组不具有相同键值，则返回 true
!==	不全等	$a!==$b	如果$a 与$b 保存的数组不具有相同键值，且顺序和数据类型也不一致，则返回 true

应用数组运算符的示例程序如下。

```
<?php
    $a = array("1"=>3,"2"=>5);
    $b = array("color"=>"red","shape"=>"round");
```

```
    $c = array("1"=>"3","2"=>"5");
    echo var_dump($a+$b)."<br>";
    echo var_dump($a==$c)."<br>";
    echo var_dump($a===$c)."<br>";
    echo var_dump($a!=$b)."<br>";
    echo var_dump($a!==$c);
?>
```

在浏览器中输出为：

```
array(4) { [1]=> int(3) [2]=> int(5) ["color"]=> string(3) "red" ["shape"]=> string(5) "round" }
bool(true)
bool(false)
bool(true)
bool(true)
```

九、错误抑制运算符

当 PHP 表达式产生错误而我们又不想将错误信息显示在页面上时，可使用错误抑制运算符。当表达式的前面被加上“@”这个运算符以后，该表达式可能产生的任何错误信息都会被忽略。使用错误抑制运算符的示例代码如下。

```
<?php
    $a = 5;
    $b = 0;
    echo ($a/$b);
?>
```

在浏览器中输出为：

```
Warning: Division by zero in D:\wamp\www\myweb\001.php on line 4
```

浏览器出现了错误提示，如果将在($a/$b)前面加上“@”这个符号，则再次运行这个程序的时候，就不会得到任何错误信息。

```
<?php
    $a = 5;
```

```
    $b = 0;
    echo @($a/$b);
?>
```

在程序的开发调试阶段，不应该使用错误抑制运算符，以便能够快速地发现错误信息。而在程序的发布阶段，可加上错误抑制运算符，以防止程序出现不友好的错误信息。

十、类型运算符

PHP5 提供了一个类型运算符“instanceof”，在 PHP 5 之前通过 is_a()实现，现在已经不推荐使用了。这个运算符用于判断指定对象是否来自于指定的类。应用类型运算符的示例代码如下。

```
<?php
    class A{}   //定义一个类A
    $a = new A();   //实例化一个类A的对象a
    var_dump($a instanceof A);   //使用类型运算符判断a是否为类A的实例
?>
```

在浏览器中输出为：

```
bool(true)
```

十一、执行运算符

执行运算符使用“`”（键盘数字 1 左边的按键）符号。使用了这个运算符以后，该运算符内的字符串将会被当做 DOS 命令行来处理。应用执行运算符的示例代码如下。

```
<?php
    $a = `dir c:\\AppServ`;
    echo $a;
?>
```

在浏览器中输出为：

```
驱动器 C 中的卷没有标签。 卷的序列号是 3C42-3B82 c:\ 的目录
```

十二、三元运算符

三元运算符的功能与“if....else”流程语句一致，它在一行中书写，代码精练、

执行效率高。在 PHP 程序中恰当地使用三元运算符能够让脚本更为简洁、高效。代码的格式如下：

```
表达式 1?表达式 2:表达式 3
```

如果表达式 1 的值为 true 则计算表达式 2，否则计算表达式 3。应用判断运算符的示例代码如下。

```php
<?php
    $a = 90;
    $b = $a>80?'成功':'失败';
    echo $b;
?>
```

在浏览器中输出为：

```
成功
```

应该注意的是：在使用三元运算符时，建议使用 print 语句替换 echo 语句，经测试，PHP4 环境下，在使用三元运算时若用 echo 语句打印内容，PHP 会报错。

十三、运算符的优先级

一个复杂的表达式往往包含了多种运算符，各个运算符优先级的不同决定了其被执行的顺序也不一样。高优先级的运算符所在的子表达式会先被执行，而低优先级的运算符所在的子表达式会后被执行。

下面表格从高到低列出了 PHP 运算符的优先级。同一行中的运算符具有相同的优先级，此时它们的结合方向决定求值顺序。左联表示表达式从左向右求值，右联相反。

运算符优先级		
结合方向	运算符	附加信息
无方向性	new	new
左	[	array()
无方向性	++ –	递增 / 递减运算符
无方向性	! ~ - (int) (float) (string) (array) (object) @	类型
左	* / %	算数运算符

左	+ - .	算数运算符和字符串运算符
左	<< >>	位运算符
无方向性	< <= > >=	比较运算符
无方向性	== != === !==	比较运算符
左	&	位运算符和引用
左	^	位运算符
左	|	位运算符
左	&&	逻辑运算符
左	||	逻辑运算符
左	? :	三元运算符
右	= += -= *= /= .= %= &= |= ^= <<= >>=	赋值运算符
左	and	逻辑运算符
左	xor	逻辑运算符
左	or	逻辑运算符

如果在开发过程中需要使用复杂的表达式运算，则可以通过添加“()”来限制各子表达式运算的优先级。

第七节 表达式

表达式是常量、变量和运算符的组合。表达式是 PHP 最重要的基石。在 PHP 中，几乎所写的任何东西都是一个表达式。简单但却最精确的定义一个表达式的方式就是“任何有值的东西”。

一、简单表达式

简单表达式是由一个单一的赋值符或一个单一函数调用组成的。由于这些表达式很简单，所以也没必要过多讨论。下面是一些例子：

```
<?php
    $a = 20;
    $a>$b;
    $a>5?'休息':'上班';
    $usenames = array('John','Marie ');
```

```
?>
```

二、复杂表达式

复杂表达式可以以任何顺序使用任意数量的数值、变量、操作符和函数。尽可能使用简短的表达式，这意味着它们更容易维护。以下是一些例子：

```
<?php
    ((10+2)/count_fishes() * 114)
?>
```

包含有三个操作符和一个函数调用的复杂表达式。

```
<?php
    initialize_count(20 -($int_page_number -1) * 2)
?>
```

有一个复杂表达式参数的简单函数调用。

有时很难分清左括号和右括号的数目是否相同。从左到右，当左括号出现时，就加 1，当右括号出现时，就从总数中减 1。如果在表达式的结尾，总数为零时，左圆括号和右圆括号的数目就一定相同了。