第2章 基本语法

1. PHP标记
   1. 一段代码，a.php。

<html>

<body>

<p>Hello HTML </p>

<p><?php echo "Hello,PHP"; ?></p>

</body>

</html>

* 1. “<?php”和“?>”是一种标记，php代码写在标记内部。标准标记，推荐。
  2. “<?php echo "Hello,PHP"; ?>”是一段PHP代码，它是嵌入到HTML结构中使用。
  3. echo是输出语句，用于输出字符串。

1. PHP注释
   1. 代码的解释，在解析时会被PHP解析器忽略。
   2. 单行注释

<?php

echo "Hello,php"; //输出一句话

?>

* 1. 多行注释“/\*……\*/”

<?php

/\*

echo "Hello,php";

$c = 10;

\*/

?>

* 1. 注释“#”

<?php

echo "Hello,php"; #输出一句话

?>

1. PHP标识符
   1. 自定义一些符号来标记一些名称，如变量名、函数名、类名等，这些符号被称为标识符。
   2. 规则
      1. 标识符可以是任意长度，只能由字母、数字、下划线组成。
      2. 标识符不能以数字开始。
      3. 标识符不能包含空格。
      4. 如果标识符由多个单词组成，那么应该使用下划线进行分隔（例如：user\_name）。
2. 关键字
   1. 标识符留给PHP专用，也称保留字，如class、public。
   2. PHP(5)中所有的关键字。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| and | or | xor | \_\_FILE\_\_ | exception |
| \_\_LINE\_\_ | array() | as | break | case |
| class | const | continue | declare | default |
| die() | do | echo | else | elseif |
| empty() | enddeclare | endfor | endforeach | endif |
| endswitch | endwhile | eval() | exit() | extends |
| for | foreach | function | global | if |
| include | include\_once | isset() | list() | new |
| print | require | require\_once | return | static |
| switch | unset() | use | var | while |
| \_\_FUNCTION\_\_ | \_\_CLASS\_\_ | \_\_METHOD\_\_ | final | php\_user\_filter |
| interface | implements | extends | public | private |
| protected | abstract | clone | try | catch |
| throw | this |  |  |  |

1. 常量的定义
   1. bool define ( string $name , mixed $value [, bool $case\_insensitive = false ] )
   2. 例

define("PAI", 3.14159, true);//不区分大小写

echo PAI.'<br/>';// 3.14159

echo pai.'<br/>';// 3.14159

1. 预定义常量
   1. PHP这些常量用于获取PHP中的信息。

|  |  |
| --- | --- |
| **常量名** | **功能描述** |
| \_\_FILE\_\_ | 默认常量，PHP程序文件名 |
| \_\_LINE\_\_ | 默认常量，PHP程序行数 |
| PHP\_VERSION | 内建常量，PHP程序的版本，如“3.0.8\_dev” |
| PHP\_OS | 内建常量，执行PHP解析器的操作系统名称，如“Windows” |
| TRUE | 该常量是一个真值（true） |
| FALSE | 该常量是一个假值（false） |
| NULL | 一个null值 |
| E\_ERROR | 该常量指到最近的错误处 |
| E\_WARNING | 该常量指到最近的警告处 |
| E\_PARSE | 该常量指到解析语法有潜在问题处 |
| E\_NOTICR | 该常量为发生不寻常，但不一定是错误处 |

* 1. 例

<?php echo \_\_FILE\_\_; //D:\site\hello\index.php

?>

1. 变量的定义
   1. 变量是由$和变量名组成。
   2. 例

$text;

$number;

$ABC\_123;

$\_book;

* 1. 变量的定义与赋值是同时进行。

<?php

$age=18;

$name="张三";

$salary=10000.5;

echo is\_double($salary);//1

?>

1. PHP的数据类型
   1. PHP的数据类型共有八种：字符串、整数、浮点数、逻辑、数组、对象、NULL。
   2. 弱语言PHP不像强类型语言那样，用类型名去声名变量。
2. 布尔类型
   1. 只有true和false，并且这两个值是不区分大小写。
   2. 例

<?php

$bool1 = true; //把true值赋给变量$bool1

$bool2 = false; //把false值赋给变量$bool2

echo is\_bool($bool1);//1

?>

* 1. 其他类型的数据也可以表示boolean值，了解。

<?php

header("Content-type: text/html; charset=utf-8");//否则会乱码

if("Helloworld")

echo "了解一下"; //了解一下

?>

1. 整型
   1. 当使用八进制表示时，数字前必须加上0（零），使用十六进制表示时，数字前必须加上0x（零x）。
   2. 例

<?php

$a = 123; //十进制数，数值为123

$b = -123; //十进制负数，数值-123

$c = 0123; //八进制数，等于十进制的83

$d = 0x123; //十六进制数，等于十进制的291

echo $d;//291

?>

1. 浮点型
   1. 标准格式

$a=3.1415

$b=3.5831

* 1. 科学计数法格式

$c=3.58E1

$d=849.52E-3

1. 字符串
   1. 字符串定义方式是单引号和双引号。
      1. $a='字符串'；
      2. $b="字符串"；
   2. 包含在双引号的字符串会被解析，而包含在单引号中的字符串不会解析，只会输出其字符本身。例：

<?php

$abc="123";

$a="{$abc}<br/>";

123

{$abc}

aaaa

echo $a;

$a='{$abc}<br/>';

echo $a;

echo "aaaa";

?>

* 1. 转义字符

|  |  |
| --- | --- |
| **序列** | **含义** |
| \n | 换行（ASCII 字符集中的 LF 或 0x0A (10)） |
| \r | 回车（ASCII 字符集中的 CR 或 0x0D (13)） |
| \t | 水平制表符（ASCII 字符集中的 HT 或 0x09 (9)） |
| \v | 垂直制表符（ASCII 字符集中的 VT 或 0x0B (11)） |
| \e | Escape（ASCII 字符集中的 ESC 或 0x1B (27)） |
| \f | 换页（ASCII 字符集中的 FF 或 0x0C (12)） |
| \\ | 反斜线 |
| \$ | 美元标记 |
| \" | 双引号 |

<?php echo "abc\ndef";?>，在命令窗口执行，不用浏览器。

1. 数组
   1. array() 函数用于创建数组。
   2. 例

<?php

$cars = array("Volvo", "BMW", "SAAB");

var\_dump($cars);//返回变量的数据类型和值

echo "<br/>";

$arrlength = count($cars);

for ($x = 0; $x < $arrlength; $x++) {

echo $cars[$x];

echo "<br>";

}

?>

输出：

array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(4) "SAAB" }

Volvo

BMW

SAAB

* 1. 例：

$aa = array("abc" => "123", "def" => "456");

echo $aa["def"];

1. 对象
   1. 定义类，类的实例化，得到对象。
   2. 例

<?php

header("content-type:text/html;charset=utf-8");

class Person

{

//人的成员属性

var $name; //人的名字

var $age; //人的年龄

//人的成员 say() 方法

function say()

{

echo "我的名字叫：" . $this->name . "<br />";

echo "我的网址是：" . $this->age;

}

} //类定义结束

//实例化一个对象

$p1 = new Person();

//给 $p1 对象属性赋值

我的名字叫：PHP中文网

我的网址是：www.php.cn

$p1->name = "PHP中文网";

$p1->age = 'www.php.cn';

//调用对象中的 say()方法

$p1->say();

?>

1. 检测变量的数据类型
   1. 变量的数据类型是在赋值的时候确定的，为了检测变量所赋的值是否符合期望的数据类型，在PHP中，提供了下面一组函数，括号里参数为要检测的变量。
   2. 相关函数

|  |  |
| --- | --- |
| **函数名称** | **功能描述** |
| is\_bool | 检测变量是否属于布尔类型 |
| is\_string | 检测变量是否属于字符串类型 |
| is\_float | 检测变量是否属于浮点类型 |
| is\_integer | 检测变量是否属于整型 |
| is\_null | 检测变量是否属于空值 |
| is\_array | 检测变量是否属于数组 |
| is\_resource | 检测变量是否属于资源 |
| is\_object | 检测变量是否属于对象类型 |
| is\_numeric | 检测变量是否属于数字或数字组成的字符串 |

* 1. 例

<?php

$a = array("zs", "ls", "ww");//定义数组a

print\_r($a);

echo ' < br />' . is\_array($a);//1

echo '<br/>' . $a[0];//zs

$a = null;

var\_dump($a);//NULL

echo is\_null($a);//1

?>

1. 可变变量
   1. 将一个变量的值作为另一个变量的名称。
   2. 例

<?php

$txt = 'abc';

$$txt = "Hello";

echo $abc; //输出hello

?>

1. 自动类型转换
   1. 转换成布尔型
      1. 有一些值会被转为false
         * 整型值0（零）
         * 浮点型值 0.0（零）
         * 空字符串，以及字符串 "0"
         * 不包括任何元素的数组
         * 不包括任何成员变量的对象（仅PHP 4.0 适用）
      2. 例

<?php

$a = array();

if ($a)//不成立

echo "1234";

?>

* + 1. 其他值会被转为true。
  1. 转换成整型
     1. 布尔型转换成整型：布尔值true，转换成整数1；布尔值false，转换成整数0。

<?php

$a=(2>1);

echo is\_bool($a);//1

var\_dump($a+3);//int(4)

?>

* + 1. 字符串型转换成整型：字符串的开始部分决定它的值。如果该字符串以合法的数值开始，则使用该数值；否则其值为 0（零）。例：

<?php

var\_dump(' 123fg456' >= 122);//bool(true)

var\_dump('some string' == 0);//bool(true)

var\_dump(123.0 == '123d456');//bool(true)

var\_dump(0 == "a");//bool(true)

var\_dump("1" == "01");//bool(true)

var\_dump("1" == "1e0");//bool(true)

当数字与字符串进行比较运算时，字符串会自动转换为数值，然后进行比较。

* 1. 转换成字符串型
     1. 数值将直接转换成数字字符串。
     2. 布尔型的true转换成字符"1"；false转换空字符串。

echo 123.5."abc";//字符串连接运算

echo true."2<br/>";//12

echo false=="";//1

1. 强制类型转换
   1. （int）、（integer）：转换成整型
   2. （float）、（double）、（real）：转换成浮点型
   3. （string）：转换成字符串
   4. （bool）、（boolean）：转换成布尔类型
   5. （array）：转换成数组
   6. （object）：转换成对象
   7. 例：

echo (int)123.99;//123

echo (integer)123.99;//123

$a = (float)123;

var\_dump($a);//float(123)

$a = (real)123;

var\_dump($a);//float(123)

$a = (double)123;

var\_dump($a);//float(123)

$a = (string)123;

var\_dump($a);// string(3) "123"

echo '<br/>';

$aa = (Object)array("abc" => "123", "def" => "456");

echo($aa->abc);

foreach ($aa as $key => $value) {

$newArray[$key] = $value;

}

var\_dump($newArray);//array(2) { ["abc"]=> string(3) "123" ["def"]=> string(3) "456" }

1. 用函数进行数据类型转换
   1. intval() floatval() strval()

例

$str="123.9abc";

$int=intval($str); //转换后数值：123

$float=floatval($str); //转换后数值：123.9

$str=strval($float); //转换后字符串："123.9"

* 1. settype()  
     例：

<?php

$num4=12.8;

$flg=settype($num4,"int");

var\_dump($flg); //输出bool(true)

var\_dump($num4); //输出int(12)

?>

1. PHP运算符与表达式

|  |  |
| --- | --- |
| **运算符类型** | **作用** |
| 算术运算符 | 用于处理四则运算 |
| 字符串运算符 | 用于连接字符串 |
| 赋值运算符 | 用于将表达式的值赋给变量 |
| 递增或递减运算符 | 用于自增或自减运算 |
| 比较运算符 | 用于表达式的比较，并返回一个真值或假值 |
| 逻辑运算符 | 用于根据表达式的值返回真值或假值 |
| 位运算符 | 用于处理数据的位运算 |
| 错误控制运算符 | 用于忽略因表达式运算错误而产生的错误信息 |
| instanceof | 用于判断一个对象是否是特定类的实例 |

1. 算数运算符

取模运算：结果和被除数的符号（正负号）相同。例：

<?php

echo (5 % 3) . "<br/>"; //2

echo (5 % -3) . "<br/>"; //2 取模运算符 % 的结果和被除数的符号相同

echo (-5 % -3) . "<br/>"; //-2

?>

再例：

<?php

$a=2;$b=3;

echo $a\*\*$b;//8 PHP 5.6，求a的b次方

?>

1. 赋值运算符

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **运算** | **范例** | **结果** |
| = | 赋值 | $a=3;$b=2; | $a=3;$b=2; |
| += | 加等于 | $a=3;$b=2;$a+=$b; | $a=5;$b=2; |
| -= | 减等于 | $a=3;$b=2;$a-=$b; | $a=1;$b=2; |
| \*= | 乘等于 | $a=3;$b=2;$a\*=$b; | $a=6;$b=2; |
| /= | 除等于 | $a=3;$b=2;$a/=$b; | $a=1.5;$b=2; |
| %= | 模等于 | $a=3;$b=2;$a%=$b; | $a=1;$b=2; |
| .= | 连接等于 | $a='abc';$a .= 'def'; | $a='abcdef' |

例：

<?php

$a = 3;

$b = &$a; // $b 是 $a 的引用

print "$a<br/>"; // 输出 3

print "$b<br/>"; // 输出 3

$a = 4; // 修改 $a

print "$a<br/>"; // 输出 4

print "$b<br/>"; // 也输出 4，因为 $b 是 $a 的引用，因此也被改变

$a;

$b;

$c;

$a = $b = $c = 5; //为三个变量同时赋值

echo $a . ' ' . $b . ' ' . $c . '<br/>';

?>

1. 递增递减运算符

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **运算** | **范例** | **结果** |
| ++ | 自增（前） | $a=2;$b=++$a; | $a=3;$b=3; |
| ++ | 自增（后） | $a=2;$b=$a++; | $a=3;$b=2; |
| -- | 自减（前） | $a=2;$b=--$a; | $a=1;$b=1; |
| -- | 自减（后） | $a=2;$b=$a--; | $a=1;$b=2; |

1. 比较运算符

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **运算** | **范例（$x=5）** | **结果** |
| == | 等于 | $x == 4 | false |
| != | 不等于 | $x != 4 | true |
| <> | 不等于 | $x <> 4 | true |
| === | 恒等 | $x === 5 | true |
| !== | 不恒等 | $x !== ’5’ | true |
| > | 大于 | $x > 5 | false |
| >= | 大于或等于 | $x >= 5 | true |
| < | 小于 | $x < 5 | false |
| <= | 小于或等于 | $x <= 5 | true |

===比较两个变量的值和类型；==比较两个变量的值，不比较数据类型。

<?php

header("Content-type: text/html; charset=utf-8");

$a=5;

if($a=="5")

echo "成立"; //成立

if($a==="5")

echo "成立"; //不会输出

?>

1. 逻辑运算符

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **运算** | **范例** | **结果** |
| && | 与 | $a && $b | $a和$b都为true，结果为true，否则为false |
| || | 或 | $a || $b | $a和$b中至少有一个为true，则结果为true，否则为false |
| ! | 非 | !$a | 若$a为false，结果为true，否则相反 |
| xor | 异或 | $a xor $b | $a和$b其一为true，但不同时是，结果为true，否则为false |
| and | 与 | $a and $b | 与&&相同，但优先级较低 |
| or | 或 | $a or $b | 与||相同，但优先级较低 |

<?php

$a = 3 or 1 ;

var\_dump($a);//int(3)，3先赋给a

$b=3||1;

var\_dump($b);//bool(true)

?>

1. 错误控制运算符
   1. PHP的错误控制运算符使用@符号来表示，把它放在一个PHP表达式之前，将忽略该表达式可能产生的任何错误信息。
   2. 例：

<?php

$a = @(4/0);

?>

* 1. 例：

<?php

header("Content-type: text/html; charset=utf-8");

//打开aaa.txt文件

$my\_file = @file ('aaa.txt') or

die ("打开文件失败: 错误信息： '$php\_errormsg'");//中止脚本执行函数

//下面语句不执行

echo "Hello";

?>

* 1. 例

<?php

header("Content-type: text/html; charset=utf-8");

$a=array("key1"=>"zs","key2"=>"ls");

$val = @$a["key3"] or die("元素不存在！");

print $val;

?>

1. 运算符优先级

|  |  |
| --- | --- |
| **结合方向** | **运算符** |
| 无 | new |
| 左 | [ |
| 右 | ++ -- ~ (int) (float) (string) (array) (object) @ |
| 无 | instance of |
| 右 | ! |
| 左 | \* / % |
| 左 | + - . |
| 左 | << >> |
| 无 | == != === !== <> |
| 左 | & |
| 左 | ^ |
| 左 | | |
| 左 | && |
| 左 | || |
| 左 | ? : |
| 右 | = += -= \*= /= .= %= &= |= ^= <<= >>= |
| 左 | and |
| 左 | xor |
| 左 | or |
| 左 | , |