**PHP学习笔记**

# 1 变量

## 1.1 变量类型

整型、浮点型、字符串、布尔类型、数组、对象、null、资源。

注意：浮点型运算会损失一点精度，原因运算时计算机在十进制转二进制时，是某些小数在十进制下是有限，转为二进制就为无限循环，计算机忽略循环部分，损失了一点进度。银行的做法通常是存款保存为整数，其单位是分，而不是元。

## 1.2 变量检测

通过isset函数判断一个变量是否存在，注意：变量被设置为null的话，isset会返回false。

|  |
| --- |
| if (isset($val)){  echo '变量$val存在'; }else{  echo '变量$val不存在'; } *// 结果：变量$val存在* |

通过gettype函数获取一个变量的类型。

|  |
| --- |
| $a = 12; echo *gettype*($a); *// 结果：integer* |

用is\_int, is\_float, is\_string等函数用来判断是否想要的类型

|  |
| --- |
| $vl = 1; if (*is\_int*($vl)){  echo '$vl是整型'; } else{  echo '$vl不是是整型'; } *// 结果：'$vl是整型'* |

## 1.3 变量打印

echo函数通常用来打印字符串，数字等。

print\_r函数通常用来打印数组。

var\_dump 可以打印所有类型，打印值和类型。

## 1.4 类型转换

变量的类型可以随意转变，比较灵活，最常用的是字符串和数字（整型）的转换。

**（1）字符串到数字转换转换规则**： 从左到右截取，碰到不合法的数字则停止，把已经截取出来的字符串转为数字，再参与运算。

|  |
| --- |
| $a = '10hello'; $a += 15; echo $a; *// 结果：25* |

**（2）数字到字符串的转换规则：**用.来拼接即可。

|  |
| --- |
| $a = 10; $a = $a.'hello'; echo $a; *// 结果：10hello* |

**（3）其他类型转为布尔类型**

除了以下值为false，其他值都为true

空字符串”” ，字符串”0”， 整型0，浮点数0.0，null， 空数组array()

## 1.5 变量赋值

变量赋值有两种，传值赋值和引用赋值。

传值赋值后，修改一个变量值，不影响另一个变量的值，两个变量相互独立，各自有自己独立的内存地址空间。

引用赋值后，修改一个变量值，另一个变量也同时修改，因为两个变量同时指向一个内存地址空间，指向的值都是一样。

## 1.6 销毁变量

使用unset函数销毁变量，注意：多个变量同时指向一个地址，销毁其中一个变量，另一个变量不影响，只有全部变量都销毁了，内存才会回收。

|  |
| --- |
| $val = 10; if (isset($val)){  echo '$val变量存在','<br>'; } *// 结果：$val变量存在* unset($val); *// 销毁变量* if (isset($val)){  echo '变量$val存在'; }else{  echo '变量$val不存在'; } *// 结果：变量$val不存在* |

## 1.7 动态变量名

变量名称的值可以做为另一个变量的变量名称。用重叠的$符号表示

|  |
| --- |
| $qinshihuang = '嬴政'; $emperor = 'qinshihuang'; $qinchao = 'emperor'; echo $qinchao,$$qinchao,$$$qinchao;  *// 结果： emperor qinshihuang 嬴政* |

## 1.8 字符串

### 1.8.1 字符串定义方式

有4种定义字符串方式，单引号和双引号的区别，heredoc和nowdoc的区别是一样，弄清楚单引号和双引号的区别，就知道heredoc和nowdoc的区别。

|  |
| --- |
| $str1 = "hello"; $str2 ='hello'; echo $str1, '<br>', $str2, '<br>';  *// 函数heredoc写大段文本* $str3 = <<<DOC 123 456 789 DOC; echo $str3,'<br>';  *// 函数nowdoc写大段文本* $str4 = <<<'DOC' abc def hij DOC; echo $str4, '<br>'; |

### 1.8.2 单双引号的对比、转义、解析和速度

**(1)转义**

|  |
| --- |
| $str1 = '\' \\ \n \t \v \$'; *// 单引号* $str2 = "\" \\ \n \t \v \$"; *// 双引号* echo $str1, '<br/>', $str2, '<br/>';  // 结果：*' \ \n \t \v \$*  *// " \ // $* |

**(2)解析**

单引号不解析里边内容，直接输出结果，双引号会解析里边的内容，所以双引号运行速度会比单引号慢，推荐使用单引号。

|  |
| --- |
| $color = 'Red'; $a = 'color is $color'; $b = "color is $color"; echo $a, ', ', $b; *// 结果：color is $color, color is Red* |

## 1.9 数组

数组是一个复合变量，可以存储多个值，各个值类型也可以不一样，每个值用不同的键来区分。

|  |
| --- |
| *// 初始化* $arr = **array**(); *// 空数组* $arr[**'name'**] = **'张三'**; $arr[**'gender'**] = **'男'**; $arr[**'age'**] = 26; *print\_r*($arr);*// 结果：Array ( [name] => 张三 [gender] => 男 [age] => 26 )* **echo '<br />'**; $arr2=**array**(**'name'**=>**'张三'**,**'gender'**=>**'男'**,**'age'**=>26); *print\_r*($arr2);*// 结果：Array ( [name] => 张三 [gender] => 男 [age] => 26 )* |

### 1.9.1 索引数组、关联数组、多维数组

|  |
| --- |
| *// (1)索引数组,它的键都为纯数字* $arr = **array**(1 => **'火车'**, 2 => **'飞机'**, 3 => **'汽车'**); *// (2)关联数组，它的键有实际意义的字符串* $arr2 = **array**(**'name'** => **'张三'**, **'gender'** => **'男'**, **'age'** => 26); *// (3)多维数组，数组单元又是一个数组* $arr3 = **array**(**'name'** => **'张三'**, **'score'** => **array**(**'chinese'** => 90, **'math'** => 95, **'english'** => 88)); *print\_r*($arr3);*// 结果： Array ( [name] => 张三 [score] => Array ( [chinese] => 90 [math] => 95 [english] => 88 ) )* |

### 1.9.2 键的规则

键可以是整数，也可以是字符串，主要有下面4条规则：

|  |
| --- |
| // (1) 如果不声明键，会从0开始递增自动生产数字键。 $arr = **array**(**'刘备'**, **'关羽'**, **'张飞'**); *print\_r*($arr);*// 结果：Array ( [0] => 刘备 [1] => 关羽 [2] => 张飞 )* **echo '<br />'**;  // (2) 如果存在一个或几个数字键，则从最大的数字键开始递增生成键。 $arr2 = **array**(4 => **'刘备'**, **'关羽'**, 8 => **'张飞'**); *print\_r*($arr2);*// 结果：Array ( [4] => 刘备 [5] => 关羽 [8] => 张飞 )* **echo '<br />'**;  // (3) 如果键声明重复，后面的值覆盖前面的值。 $arr3 = **array**(1 => **'刘备'**, **'关羽'**, 1 => **'张飞'**); *print\_r*($arr3);*// 结果：Array ( [1] => 张飞 [2] => 关羽 )* **echo '<br />'**;  // (4) 如果键是浮点数，会转成整数的键，如果键是字符串，这个字符串恰好是整数，也会转成整数的键。 $arr4 = **array**(3 => **'刘备'**, 4.5 => **'关羽'**, **'4.5'** => **'张飞'**, **'6'** => **'诸葛亮'**, **'7Eleven'** => **'赵云'**); *print\_r*($arr4);*// 结果：Array ( [3] => 刘备 [4] => 关羽 [4.5] => 张飞 [6] => 诸葛亮 [7Eleven] => 赵云 )* |

### 1.9.3 数组单元的操作

数组单元操作有增、读、改、删。

|  |
| --- |
| $arr = **array**(**'name'** => **'张三'**); $arr[**'class'**] = **'3（2）班'**;*// 增加年级单元* $arr[**'score'**] = **array**(**'Chinese'** => 90, **'Math'** => 95, **'English'** => 35);*// 增加成绩单元 print\_r*($arr);*// 结果：Array ( [name] => 张三 [class] => 3（2）班 [score] => Array ( [Chinese] => 90 [Math] => 95 [English] => 35 ) )* **echo '<br />'**;  **echo** $arr[**'score'**][**'English'**];*// 读取单元值，结果：35* **echo '<br />'**;  $arr[**'score'**][**'English'**] = 85; *// 修改单元值 print\_r*($arr);*// 结果：Array ( [name] => 张三 [class] => 3（2）班 [score] => Array ( [Chinese] => 90 [Math] => 95 [English] => 85 ) )* **echo '<br />'**;  **unset**($arr[**'score'**]); *// 删除单元值 print\_r*($arr);*// 结果：Array ( [name] => 张三 [class] => 3（2）班 )* |

### 1.9.4 循环遍历数组

for循环，只能用在索引数组中。

foreach as 可以用在索引数组、关系数组、多维数组中。

while each可以用在索引数组、关系数组、多维数组中。

|  |
| --- |
| *// (1) for循环*  $arr = **array**(**'a'**, **'b'**, **'c'**); **for** ($i = 0; $i < *count*($arr); $i++) {  **echo** $arr[$i], **' '**; *// 结果：a b c* } **echo '<br />'**;  *// (2) foreach as循环* $arr2 = **array**(**'R'** => 80, **'G'** => 200, **'B'** => 255); **foreach** ($arr2 **as** $k => $v) {  **echo** $k, **': '**, $v, **', '**; *// 结果：R: 80, G: 200, B: 255,* } **echo '<br />'**;  *// 简化版，只读取值，忽略键* **foreach** ($arr **as** $v) {  **echo** $v, **' '**; *// 结果：a b c* }  *// (3) while each循环*  $arr3 = **array**(0,1,2,3,4,**'false'**,0,1,2,3,4); **while** (**list**($k, $v) = *each*($arr3)) {  **echo** $v,**' '**; *// 结果0 1 2 3 4 false 0 1 2 3 4* } |

## 1.10 超级全局变量

在各个页面都可以使用的变量。

### 1.10.1 $\_GET, $\_POST, $\_REQUEST全局变量

$\_GET 从地址URL上获得数据。

$\_POST 从网页POST表单发送的数据。

$\_REQUEST 是变量$\_GET和$\_POST的集合。既有URL的数据，也有POST提交的数据。

注意：因为$\_REQUEST是$\_GET和$\_POST的组合，提交时数据名相同

的就会产生冲突，冲突优先选择是$\_GET还是$\_POST取决于php.ini配置文件设置选项request\_order，如果等于"GP"，则$\_REQUEST获取POST提交的数据，如果是"PG"，则$\_REQUEST获取的是GEt提交的数据。

|  |
| --- |
| *<!--test.html文件-->*  <!DOCTYPE html> <**html** lang=**"en"**> <**head**>  <**meta** charset=**"UTF-8"**>  <**title**>测试php的GET、 POST、 Request</**title**> </**head**> <**body**>  <**form** action=**"test.php?usr=Tom&pwd=123456"** method=**"post"**>  <**label** for=**"usr"**>用户名：<**input** id=**"usr"** type=**"text"** name=**"usr"**></**label**>  <**input** type=**"submit"** value=**"提交"**> </**form**>  </**body**> </**html**>  ------------------------------------------------------------------------  *// test.php文件 print\_r*($\_GET);*// 结果：Array ( [usr] => Tom [pwd] => 123456 ) print\_r*($\_POST);*// 结果：Array ( [usr] => 张三 ) print\_r*($\_REQUEST);*// 结果：Array ( [usr] => 张三 [pwd] => 123456 )* |

### 1.10.2 $\_ENV, $\_SERVER全局变量

$\_ENV 是指服务器操作系统的信息，如系统类型、系统环境变量，这些信息比较敏感，实际使用中是被禁止的，禁止方法：设置在配置文件php.ini的variables\_order = "GPCS"，重启服务软件即可。禁止后当然还可以使用函数getenv来获取系统信息。

$\_SERVER 是指运行该服务器的单一一个web服务信息。可以获取网站的域名SERVER\_NAME、来访者的IP地址REMOTE\_ADDR、当前运行脚本REQUEST\_URI等信息。

|  |
| --- |
| *// 禁止$\_ENV后可以使用getenv来获取系统信息* **echo** *getenv*(**'OS'**); **echo** *getenv*(**'Path'**); **echo** *getenv*(**'PATHEXT'**);  *//获取当前域名* **echo** $\_SERVER[**'HTTP\_HOST'**]; **echo** $\_SERVER[**'SERVER\_NAME'**];  *//获取当前ip* **echo** $\_SERVER[**'SERVER\_ADDR'**];  *//获取当前服务器标识的字串，在响应请求时的头部中给出* **echo** $\_SERVER[**'SERVER\_SOFTWARE'**];  *//获取请求页面时通信协议的名称和版本。例如，“HTTP/1.0”。* **echo** $\_SERVER[**'SERVER\_PROTOCOL'**];  *//当前正在执行脚本的文件名* **echo** $\_SERVER[**'PHP\_SELF'**];  *//获取来源网址,即点击来到本页的上页网址 //echo $\_SERVER['HTTP\_REFERER'];  //获取当前域名的后缀* **echo** $\_SERVER[**'REQUEST\_URI'**];  *//获取当前文件的物理路径* **echo** *dirname*(***\_\_FILE\_\_***);  *//获取当前文件的上一级物理路径* **echo** *dirname*(***\_\_FILE\_\_***) . **"/../"**; |

$GLOBALS 全局变量

$GLOBALS是对全局变量的花名的一个别名，函数内可以通过$GLOBALS来访问和修改函数外的全局变量，一般不推荐使用。

|  |
| --- |
| $a = 10; **echo** $a,**' '**; *// 结果：10* **function** f(){  $GLOBALS[**'a'**]=20; } f(); **echo** $a; *// 结果：20* |

## 1.11 常量

声明：define(常量名, 值)

常量名称：通常所有字母都是大写。

特点：一旦声明后，不能修改值、不能重复定义、不能销毁。

使用范围：类似超级全局变量，在所有页面都可以使用。

检测常量方法：使用defined函数。

动态常量名：用变量的值作为常量的名称。

|  |
| --- |
| *define*(**'PI'**,3.14); **echo *PI***,**'<br />'**; *// 结果：3.14  //PI = 3.14159; // 禁止修改 //define('PI',3.14159); // 禁止重定义 //unset(PI); // 禁止销毁* **function** f(){  **echo *PI***,**'<br />'**; *// 结果：3.14* } f();  **if** (*defined*(**'PI'**)){  **echo '常量PI已经存在'**,**'<br />'**; } *// 结果：常量PI已经存在* **if** (!*defined*(**'E'**)){  **echo 'E不存在，重新定义。'**,**'<br />'**;  *define*(**'E'**,2.72); } *// 结果：E不存在，重新定义。* $a = **'PI'**; **echo** *constant*($a);*// 使用动态变量名，结果：3.14* |

# 2运算符

## 2.1 算数，加、减、乘、除、求余

注意：整型加法或乘法，如果最终结果超出2^32,会自动转换为比整型范围更大范围浮点数；

除法运算时，除数不能为零。

求余运算时，结果的正负取决于被除数。

|  |
| --- |
| $a = 3000000000; $b = $a + $a; *var\_dump*($b); *// 6000000000, 浮点型* $c = 2 \* $a; *var\_dump*($c); *// 6000000000, 浮点型* $i = 10; $j = 0; *//echo $i/$j; // 报错  // 求余结果符号取决于被除数* echo $i % 3, $i % -3; *// 结果都是1* echo -$i % -3, -$i % -3; *// 结果都是-1* |

## 2.2 比较运算

比较运算的结果是布尔类型，只有真和假两种情况。

注意：全等于判断，要求类型相同并且值相等，结果才为真，否则为假。

|  |
| --- |
| $a = 123; $b = '123'; *var\_dump*($a == $b); *// 结果： true var\_dump*($a === $b); *// 结果： false* |

## 2.3 逻辑运算

|  |
| --- |
| $a = 5; $b = 12;  echo $a & $b, '<br />'; *// 逻辑与， 结果：4* echo $a | $b, '<br />'; *// 逻辑或， 结果：13* echo ~$a, ' ', ~$b, '<br />';*// 取反(公式：-x-1)， 结果：-6* echo $a ^ $b, '<br />'; *// 异或， 结果：9* echo $a << 1, '<br />'; *// 左移， 结果：10* echo $b >> 2, '<br />'; *// 右移， 结果：3* |

短路特性：

(1) 多个条件逻辑与时，从左到右判断，如果判断到一个为假，跳过后面的判断。

(2) 多个条件逻辑或时，从左到右判断，如果判断到一个为真，跳过后面的判断。

|  |
| --- |
| $a = false; $b = 1; if ($a && $b++) {  } echo $b; *// 结果：1* $a = true; $b = 1; if ($a || $b++) {  } echo $b; *// 结果：1  defined*('PI', 3.14) || *define*('PI', 3.14);*// 意思就是判断如果还没定义常量PI，在这里定义常量PI，这是利用逻运算的短路特性* |

## 2.4 三元运算

|  |
| --- |
| $a = 10; $b = 20; $c = 30; echo $a > $b ? $a : $b;*// 结果：20* $d = $a > $b ? $a : $b; echo $d > $c ? $d : $c; *// 获取三个数中的最大数，结果：30* |

## 2.5 自增子减运算

|  |
| --- |
| $a = 10; $b = $a++; *// 先赋值给变量$b,然后再自增1* $c = ++$a; *// 先自增1,然后再赋值给变量$c* echo $a, $b, $c; *// 结果：11,10,12* |

## 2.6 字符串运算符

字符串用.来拼接，也可以用来拼接数字。

|  |
| --- |
| $a = 'hello '; $b = 'world'; $c = 123; $d = $a . $b; $e = $d . $c; echo $d, ', ', $e; *// 结果：hello world, hello world123* |

# 3 控制结构

## 3.1条件控制

关键词：if —> if else —> else

|  |
| --- |
| $a = 75; if ($a<60){  echo '不及格'; }else if ($a>=60 && $a<80){  echo '及格'; }else{  echo '优秀'; } *// 结果：及格* |

## 3.2选择控制

**关键词：switch —> case —> break，一般使用在验证多个可能的值，不适合验证值的范围。**

|  |
| --- |
| $a = 3; switch ($a){  case 1:  echo '去北京';  break;  case 2:  echo '去上海';  break;  case 3:  echo '去广州';  break;  default:  echo '待在家里'; }  *// 结果：去广州* |

## 3.3循环控制结构

### 3.3.1 while和do while

关键词：while或do while，用自增自减作为判断条件时需要注意边界问题。

|  |
| --- |
| $i = 0; while ($i<10){ *// 有可能一次都不执行* echo $i++,' '; } *// 结果：0 1 2 3 4 5 6 7 8 9*  do{ *// 必须执行一次* echo $i++,' '; }while($i<20); *// 结果：10 11 12 13 14 15 16 17 18 19* |

### 3.3.2 for循环控制结构

|  |
| --- |
| for ($i = 0; $i < 10; $i++) {  echo $i, ' '; *// 结果：0 1 2 3 4 5 6 7 8 9* }  *// for 循环可以同时初始化多个变量* echo '<br />'; for ($i = 0, $j = 3; $i < 3; $i++, $j--) {  echo $i, ' ', $j, '，'; *// 结果： 0 3，1 2，2 1，* } |

## 3.4 break和continue控制

break在while、 for、switch控制结构使用，退出循环或选择结构。

continue在循环结构使用，跳过该条件。

|  |
| --- |
| *// 从一组数中取出数值大小在2到5之间的数* for ($i = 0; $i < 10; $i++) {  if ($i<2){  continue;  }  if ($i > 5){  break;  }  echo $i, ' '; *// 结果：2 3 4 5* } |

# 4 函数

一段封装好的代码，可以随时调用。

函数命名规则：不区分字母大小写，函数Add和add是一样的。而变量的名称是区分大小写的。注意：函数不能重名，变量可以。

|  |
| --- |
| function 函数名([参数1],[参数2]......){// 参数可省略  // 执行语句，可省略  // return，可省略，如果返回，只能返回一个值。  } |

函数传参有赋值传参和引用传参

赋值传参，不影响函数外的变量，因为是两个独立的变量，各自有自己的内存地址。

引用传参，会影响函数外的变量值，因为它们共同指向同一个地址。一般不推荐使用，会破坏函数的封装性。

|  |
| --- |
| function f($a, &$b) {  $a = $a\*$a;  $b = $b\*$b; } $i = 10; $j = 20; f($i, $j); echo $i, ', ', $j; *// 结果：10, 400* |

## 4.1 函数中变量的作用域

声明变量的作用域：

1. 局部变量：声明在函数内部。
2. php页面的普通全局变量：在函数外用$符号声明，只能在该页面函数外有效，函数内无效。
3. global全局变量：在整个页面内（包括函数体内）声明，整个页面都有效，不推荐声明使用。
4. 超级全局变量：系统内部声明，各个页面各个函数都有效。例如：$\_GET，$\_POST

|  |
| --- |
| $a = 10; function f() {  $b = 20; *// echo $a; // 报错，在函数内不能使用全局变量* }  *//echo $b; // 报错，在函数外不能使用局部变量* function f2() {  global $a; *// 不推荐使用* $a++;  echo $a;*// 结果：11* } f(); f2(); |

## 4.2 动态调用函数

可以用变量的值当做函数名，并调用函数。

|  |
| --- |
| function hello(){  echo 'hello world'; }  $a = 'hello'; if (*function\_exists*($a)){ *// 判断函数是否存在* $a(); *// 结果：hello world* } |

# 5 常用的系统函数

## 5.1 时间相关函数

### 5.1.1 时间格式化

|  |
| --- |
| echo *time*(); *// 结果：1469167206* echo '<br>'; echo *microtime*();*// 结果：0.21987700 1469167206* echo '<br>'; echo *microtime*(true); *// 结果： 1469167206.2199* echo '<br>';  *// 时间格式化* $time = *time*(); echo *date*('Y-m-d', $time); *// 输出年月日xxxx-xx-xx* echo '<br>'; echo *date*('h:i:s', $time); *// 输出时分秒xx:xx:xx* echo '<br>'; echo *date*('Y-m-d h:i:s 星期N', $time); *// 格式xxxx-xx-xx xx:xx:xx 星期x* echo '<br>'; echo *date*('Y-m-d h:i:s 星期N', 0); *// 返回1970时间戳的格式化，注意是08:00:00* echo '<br>'; echo *gmdate*('Y-m-d H:i:s 星期N', 0); *// 返回1970时间戳的格式化，是00:00:00* |

### 5.1.2 解释和检测日期

|  |
| --- |
| echo *mktime*('17', '14', '20', '7', '20', '2016'); *// 生成时间戳* echo '<br>'; echo *strtotime*('now'); *// 把当前日期时间生成时间戳* echo '<br>'; echo *strtotime*('now +1 hour'); *// 一小时后的时间戳 同时适用于day, minute等* echo '<br>'; echo *strtotime*('now -1 hour'); *// 一小时前的时间戳  // 检查输入日期是否合法 var\_dump*(*checkdate*(2, 29, 2015)); *// 不合法 var\_dump*(*checkdate*(2, 29, 2016)); *// 合法* |

## 5.2 字符串相关函数

### strlen 获取字符串长度函数

|  |
| --- |
| $str = 'zgr中国人'; echo *strlen*($str);*// 结果：12* echo *mb\_strlen*($str, 'utf-8');*// 结果：6* |

### strpos 查找字符串位置函数

|  |
| --- |
| $str = 'abcdefg'; $s = 'a'; $pos = *strpos*($str, $s); if ($pos !== false) {*// 注意必须用全不全等号，不能简单用if（）判断* echo $s, ' position at ', $pos, '<br>'; } else {  echo 'not fount<br>'; } *// 解果：a position at 0* |

### str\_replace 替换字符串

|  |
| --- |
| *// 只能替换一种字符串* $str = 'hello hello world'; echo *str\_replace*('hello', 'h\*\*\*o', $str), '<br>'; *// 结果：h\*\*\*o h\*\*\*o world  // 批量替换字符串* $str = 'aaAABBbb'; echo *strtr*($str, array('aa' => 'AA', 'AA' => 'aa', 'bb' => 'BB', 'BB' => 'bb')); *// 结果：AAaabbBB* |

### substr 截取字符串

|  |
| --- |
| $str = 'I am chinese'; echo *substr*($str, 0, 4) . '<br>';*// 结果：I am* echo *substr*($str, 5) . '<br>'; *// 结果：chinese* echo *substr*($str, 0, -1) . '<br>'; *// 结果：I am chines* echo *substr*($str, -7, -1) . '<br>';*// 结果：chines* |

### explode字符串拆分, implode数组内容合成字符串

|  |
| --- |
| $str = 'red,green,blue'; $arr = *explode*(',', $str);*// 字符串拆分成数组 print\_r*($arr);  *// 结果：Array ( [0] => red [1] => green [2] => blue )*  echo '<br>', *implode*($arr, ',');*// 数组元素连接成字符串，*  *// 结果：red,green,blue* |

### trim ltrim ltrim 去除字符串的首尾空白字符（\t\n\r\0\x0B）

|  |
| --- |
| $str = "\n\t abc \n\t \r"; echo *strlen*($str), '<br>'; *// 结果：10* echo *strlen*(*ltrim*($str)), '<br>'; *// 去除左边空白字符，结果：7* echo *strlen*(*ltrim* ($str)), '<br>'; *// 去除右边空白字符，结果：6* echo *strlen*(*trim*($str)), '<br>'; *// 去除首尾空白字符，结果：3* |

### hunk\_split 按长度分割并插入分割字符串

|  |
| --- |
| $str ='abcd'; echo *chunk\_split*($str,1,', '); *// 结果：a, b, c, d,* |

### str\_pad指定长度填充字符串

|  |
| --- |
| $str='hello'; echo *str\_pad*($str, 10), '<br>'; *// 结果："hello "* echo *str\_pad*($str, 10, "-=", *STR\_PAD\_LEFT*), '<br>';  *//结果："-=-=-hello"* echo *str\_pad*($str, 10, "\_", *STR\_PAD\_BOTH*), '<br>';  *//结果： "\_\_hello\_\_\_"*  echo *str\_pad*($str, 6 , "\_\_\_"), '<br>'; *//结果： "hello\_"* |

### addslashes字符转义与stripcslashes反转义

转义字符包括单引号（'）、双引号（"）、反斜线（\）与 NUL（NULL 字符）。

|  |
| --- |
| $str = "Tom's book? \"say\""; $s = *addslashes*($str); *// 转义，结果：Tom\'s book? \"say\"* echo $s, '<br>'; echo *stripcslashes*($s); *// 反转义，结果：Tom's book? "say"* |

### htmlspecialchars 把HTML实体转为字符(编码)，htmlspecialchars\_decode 将字符转为HTML实体（解码）

使用中文时没什么区别,但htmlentities会格式化中文字符使得中文输入时乱码。

|  |
| --- |
| $str = "<div>number:<input type='text'></div>"; $s = *htmlspecialchars*($str, *ENT\_QUOTES*); *// 把HTML实体转为字符 var\_dump*($s); $ss = *htmlspecialchars\_decode*($s, *ENT\_QUOTES*);*//将字符转为HTML实体 var\_dump*($ss);  *// 结果：string '&lt;div&gt;number:&lt;input type=&#039;text&#039;&gt;&lt;/div&gt;' (length=65)* *// 结果：string '<div>number:<input type='text'></div>' (length=37)* $str = "<div>数字:<input type='text'></div>"; $s = *htmlentities*($str, *ENT\_QUOTES*); *// 把HTML实体转为字符 var\_dump*($s);$ss = *html\_entity\_decode*($s, *ENT\_QUOTES*); *//将字符转为HTML实体 var\_dump*($ss);  *// 结果：string '&lt;div&gt;数字:&lt;input type=&#039;text&#039;&gt;&lt;/div&gt;' (length=65)* *// 结果：string '<div>数字:<input type='text'></div>' (length=37)* |

### strcmp 字符比较

|  |
| --- |
| $str1 = 'hello'; $str2 = 'Hello'; if (*strcmp*($str1, $str2) != 0) { *// 区分大小写比较* echo $str1, ' != ', $str2, '<br>'; *// 结果：hello != Hello* } if (*strcasecmp*($str1, $str2) == 0) { *// 不区分大小比较* echo $str1, ' = ', $str2, '<br>'; *// 结果：hello = Hello* } |

### strtolower strtoupper ucfirst ucwords字符串转换函数

|  |
| --- |
| $str = 'hi PHP, Hello world'; echo *strtolower*($str) . '<br>';  *// 所有字母为小写，结果：hi php, hello world* echo *strtoupper*($str) . '<br>';  *// 所有字母为大写，结果：HI PHP, HELLO WORLD* echo *ucfirst*($str) . '<br>';  *// 第一个字母转为大写，结果：Hi PHP, Hello world* echo *ucwords*($str) . '<br>';  *// 每个单词首字母转为大写，结果：Hi PHP, Hello World* |

## 5.3 数组相关函数

php的数组函数功能非常强大，开发使用频率较多，所以需要熟练掌握常用的操作数组的函数。

### current，next，prev， end，reset 数组游标操作函数

数组游标操作函数有current，next，prev， end，reset

注意：和do while循环一起用时，如果数组单元的值为0、空字符串、字符串false等会中断循环，需要用!==false作为while的判断条件。

|  |
| --- |
| $arr = array('A', 'B', 'C', 'D'); echo *current*($arr); *// 当前游标指向的数组的单元：A* echo *next*($arr); *// 指向下一个数组单元：B* echo *end*($arr); *// 最后一个数组单元：D* echo *prev*($arr); *// 前一个数组单元：C* echo *reset*($arr); *// 复位游标，即指向第一个单元：A* echo '<br />';  $arr2 = array(0, 1, 'false', '', '0', 0, 1); *reset*($arr); do {  echo *current*($arr2); *// 结果：01false001* } while (*next*($arr2) !== false); *// 顺序输出* echo '<br />';  *end*($arr2); do {  echo *current*($arr2); *// 结果：100false10* } while (*prev*($arr2) !== false); *// 倒序输出* |

### count 计算数组所有单元的个数

|  |
| --- |
| $arr = array('A','B','C','D'); echo *count*($arr); *// 结果：4* |

### array\_key\_exists 判断数组的键是否存在

|  |
| --- |
| $arr=array('usr'=>'张三','pwd'=>'123456','login'=>null); *// 注：不能用isset函数来判断值为null的变量*  if (*array\_key\_exists*('login',$arr)){  echo 'login 存在'; }else{  echo 'login不存在'; } *// 结果：login存在* |

### in\_array 判断数组中的值是否存在

|  |
| --- |
| $arr = array('张飞','刘备','关羽'); *var\_dump*(*in\_array*('关羽',$arr)); *// 结果：true var\_dump*(*in\_array*('曹操',$arr)); *// 结果：false* |

### array\_push, array\_unshift 插入一个或多单元

array\_push在数组末尾追加一个或多单元，键自动增加。

array\_unshift在数组开头插入一个或多单元，。注意单元是作为整体被插入的，因此传入单元将保持同样的顺序。所有的数值键名将修改为从零开始重新计数，所有的文字键名保持不变。

|  |
| --- |
| $arr=array('A', 'B', 'C'); *array\_push*($arr,'D'); *array\_push*($arr,'E','F'); *print\_r*($arr);*// 结果：Array ( [0] => A [1] => B [2] => C [3] => D [4] => E [5] => F )  array\_unshift*($arr,'php'); *array\_unshift*($arr,'java','C++');*// 注意是从参数右到左开始的，索引从新排列 print\_r*($arr);*// 结果：Array ( [0] => java [1] => C++ [2] => php [3] => A [4] => B [5] => C [6] => D [7] => E [8] => F)* |

### array\_pop, array\_shift 删除一个单元

array\_pop 删除数组最后一个单元。

array\_shift删除开头一个单元。其它单元向前移动一位，所有的数字键名将改为从零开始计数，文字键名将不变。

|  |
| --- |
| $arr=array('A', 'B', 'C', 'D', 'E'); *array\_pop*($arr); *print\_r*($arr);*// 结果：Array ( [0] => A [1] => B [2] => C [3] => D )  array\_shift*($arr); *print\_r*($arr);*// 结果：Array ( [1] => B [2] => C [3] => D )* |

### array\_keys, array\_values 获取数组的键和值

array\_keys获取数组中部分的或所有的键名。

array\_values 获取数组中所有的值。

|  |
| --- |
| $arr = array('小米5'=>2199,'iphone6s'=>4200,'三星S7'=>4200,'HTC10'=>3800); *print\_r*(*array\_keys*($arr));*// 获取所有键 // 结果：Array ( [0] => 小米5 [1] => iphone6s [2] => 三星S7 [3] => HTC10 )*  *print\_r*(*array\_keys*($arr,4200,true));*// 获取值为4200的键 // 结果：Array ( [0] => iphone6s [1] => 三星S7 )*  *print\_r*(*array\_values*($arr));*// 获取所有值 // 结果：Array ( [0] => 2199 [1] => 4200 [2] => 4200 [3] => 3800 )* |

### array\_sum 计算数组中所有值的和

|  |
| --- |
| $arr=*range*(5,15);*// 生成值为5到15的数组* echo *array\_sum*($arr);*// 结果：110* |

### range 生成规定范围的等差数组

|  |
| --- |
| foreach (*range*(1, 10, 2) as $v) {  echo $v, ' '; *// 结果：1 3 5 7 9* } foreach (*range*('D', 'A') as $v) {*// 如果是字符串，则取第一字符* echo $v, ' ';*// 结果： D C B A* } |

### array\_fill 用给定的值填充数组

|  |
| --- |
| *print\_r*(*array\_fill*(2, 3, 'apple'));*// 参数：起始索引 填充个数 值* *// 结果：Array ( [2] => apple [3] => apple [4] => apple )* |

### array\_filter 过滤数组单元

|  |
| --- |
| $arr=*range*(1,20);*// 生产1到20的数组* function isOdd($v){ *// 回调函数功能是判断奇数* return ($v&1); } foreach(*array\_filter*($arr,"isOdd") as $v){  echo $v,' '; *// 结果：1 3 5 7 9 11 13 15 17 19* } |

### array\_flip 交换数组中的键和值

|  |
| --- |
| $arr = array('HTC10' => 3800, 'iphone6s' => 4200, '小米5' => 2199, '三星S7' => 4200); *print\_r*(*array\_flip*($arr));*// 注意：翻转之后，当键有相同会覆盖掉 // 结果：Array ( [3800] => HTC10 [4200] => 三星S7 [2199] => 小米5 )* |

### array\_merge, array\_merge\_recursive 合并数组

array\_merge合并一个或多个数组。

array\_merge\_recursive递归地合并一个或多个数组。如果输入的数组中有相同的字符串键名，则这些值会被合并到一个数组中去，这将递归下去，因此如果一个值本身是一个数组，本函数将按照相应的条目把它合并为另一个数组。然而，如果数组具有相同的数组键名，后一个值将不会覆盖原来的值，而是附加到后面。

|  |
| --- |
| $arr = array('red' => 64, 'green' => 255, 'blue' => 128); $int = 99; $str = 'name'; $arr2 = array('red' => 1, 'green' => 2); $arr3 = *array\_merge*($arr, array($int), array($str), $arr2);*// 注：当键名相同会覆盖。 print\_r*($arr3); *// 结果：Array ( [red] => 1 [green] => 255 [blue] => 128 [0] => 99 [1] => name )* $arr4 = array("color" => $arr); $arr5 = array("color" => $arr3); *print\_r*(*array\_merge\_recursive*($arr4, $arr5)); *// 结果： Array ( [color] => Array ( [red] => Array ( [0] => 64 [1] => 1 ) [green] => Array ( [0] => 255 [1] => 2 ) [blue] => Array ( [0] => 128 [1] => 128 ) [0] => 99 [1] => name ) )* |

### sort, asort, usort 排序

|  |
| --- |
| $arr = array('HTC10' => 3800, 'iphone6s' => 4200, '小米5' => 2199); *sort*($arr);*// 值排序,注意：arr被统一转化为索引数组，去掉原来的键。 print\_r*($arr);  *// 结果：Array ( [0] => 2199 [1] => 3800 [2] => 4200 )* $arr2 = array('sansumg', 'apple', 'huawei'); *asort*($arr2); *// 键和值保持相对应 print\_r*($arr2);  *// 结果：Array ( [1] => apple [2] => huawei [0] => sansumg )*  $arr = array(8, 3, 7, 9, 1); function cmpInt($a, $b) { *// 用户自己定义的排序方法。*  if ($a === $b) {  return 0;  }  return $a > $b ? 1 : -1; } *usort*($arr, "cmpInt"); *print\_r*($arr);  *// 结果：Array ( [0] => 1 [1] => 3 [2] => 7 [3] => 8 [4] => 9 )* |

### array\_change\_key\_case获得字符串键名全为小写或大写的数组

|  |
| --- |
| $arr = array('Usr' => '张三', 'Pwd' => '123456'); *print\_r*(*array\_change\_key\_case*($arr));*// 结果：Array ( [usr] => 张三 [pwd] => 123456 ) print\_r*(*array\_change\_key\_case*($arr, *CASE\_UPPER*));*// 第二个参数 CASE\_LOWER（默认值）或CASE\_UPPER // 结果：Array ( [USR] => 张三 [PWD] => 123456 )* |

### array\_count\_values统计数组中所有的值出现的次数

|  |
| --- |
| $arr = array('A', 'B', 'C', 'A', 'A', 'C'); *print\_r*(*array\_count\_values*($arr));  *// 结果：Array ( [A] => 3 [B] => 1 [C] => 2 [b] => 1 )* |

### array\_diff, array\_diff\_assoc数组的差集

array\_diff该数组包括了所有在 array1 中但是不在任何其它参数数组中的值。注意键名保留不变。

array\_diff\_assoc 和 array\_diff() 不同的是键名也用于比较。

|  |
| --- |
| $arr1 = array('A','B','C'); $arr2 = array('B','C','D'); *print\_r*(*array\_diff*($arr1,$arr2)); *// 结果：Array ( [0] => A ) print\_r*(*array\_diff*($arr2,$arr1)); *// 结果：Array ( [0] => D )*  $arr1 = array('red' => 128, 'green' => 255); $arr2 = array('green' => 255, 'blue' => 64); *print\_r*(*array\_diff\_assoc*($arr1,$arr2)); *// 结果：Array([red]=>128) print\_r*(*array\_diff\_assoc*($arr2,$arr1)); *// 结果：Array([blue]=>64)* |

### array\_intersect, array\_intersect\_assoc数组的交集

array\_intersect 获取在所有比较的数组中出现的值，注意键名保留不变。

array\_intersect\_assoc 和 array\_intersect() 不同的是键名也用于比较。

|  |
| --- |
| $arr1 = array('A', 'B', 'C'); $arr2 = array('B', 'C', 'D'); *print\_r*(*array\_intersect*($arr1, $arr2));  *// 结果：Array ( [1] => B [2] => C )* $arr1 = array('red' => 128, 'green' => 255); $arr2 = array('green' => 255, 'blue' => 64); *print\_r*(*array\_intersect\_assoc*($arr1, $arr2));  *// 结果：Array ( [green] => 255 )* |

### array\_unique 删除数组中重复的值

|  |
| --- |
| $arr = array('A', 'B', 'C', 'A', 'b', 'C'); *print\_r*(*array\_unique*($arr));*// Array ( [0] => A [1] => B [2] => C [4] => b )* |

### array\_reverse 获取相反顺序的数组

|  |
| --- |
| $arr = array('A', 'B', 'C'); *print\_r*(*array\_reverse*($arr));  *// 结果：Array ( [0] => C [1] => B [2] => A ) print\_r*(*array\_reverse*($arr, true)); // 倒序保留原来对应的键  *// 结果：Array ( [2] => C [1] => B [0] => A )* |

### shuffle 将数组打乱（随机）

|  |
| --- |
| $arr = *range*(1, 6); *shuffle*($arr); *print\_r*($arr); *// 结果：Array ( [0] => 2 [1] => 5 [2] => 6 [3] => 4 [4] => 1 [5] => 3 )* |

### srand 播下随机数发生器种子

|  |
| --- |
| *srand*((double)*microtime*()\*1000000); $randval = *rand*(1000000,9999999); *// 参数表示输出范围* echo $randval; *// 随机结果：6413513* |

### array\_rand从数组中随机取出一个或多个单元

|  |
| --- |
| $arr = array('语文' => 80, '数学' => 90, '英语' => 85, '体育' => 92, '音乐' => 96); *print\_r*(*array\_rand*($arr, 2));*// 随机取出2个键 // 随机结果：Array ( [0] => 数学 [1] => 体育 )* |

## 5.4 进制转换相关函数

|  |
| --- |
| echo *decbin*(222), '<br />'; *// 十进制转换为二进制， 结果：11011110* echo *dechex*(222), '<br />'; *// 十进制转换为二进制， 结果：0xde* echo *hexdec*(222), '<br />'; *// 十六进制转换为十进制， 结果：546* echo hex2bin("48656c6c6f20576f726c6421"), '<br />';  *// 把十六进制值的字符串转换为 ASCII 字符，结果：Hello World!* echo *bin2hex*("Hello world!"), '<br />';  *// 把 ASCII 字符的字符串转换为十六进制值，结果：48656c6c6f20576f726c6421* echo *bindec*("11011110") . "<br>";  *// 把二进制转为十进制，结果：2222* |

## 5.5 文件操作相关函数

|  |
| --- |
| $fileName = **'./a.txt'**;  *// 写入内容* $file = *fopen*($fileName,**'a+'**); *// 以追加方式打开，文件不存在则创建* **if** ($file){  *fwrite*($file,**"1234567890\r\n"**);  *fwrite*($file,**"abcdefg\r\n"**);  *fclose*($file); }  *// 一次性把全部内容读取* $file=*fopen*($fileName,**'r'**); *// 只读打开* **if** ($file){  $fSize = *filesize*($fileName); *// 获取文件大小* $content=*fread*($file,$fSize); *// 读取文件内容* **echo** $content,**'<br>'**;*// 读取所有内容  fclose*($file); }  *// 按行读取内容* $file=*fopen*($fileName,**'r'**); *// 只读打开* **if** ($file){  **while** (($content=*fgets*($file))!==**false**){  **echo** $content,**'<br>'**;*// 读取所有内容* }  *fclose*($file); }  *// 把文件内容读取到数组中，按行分割。* $fArr=*file*($fileName,***FILE\_IGNORE\_NEW\_LINES*** | ***FILE\_SKIP\_EMPTY\_LINES***); *// 把文件读入数组中，第二个参数是去掉换行符和跳过空行* **foreach**($fArr **as** &$line) {  **echo** $line.**'<br />'**; } |

# 6 文件包含

作用：代码的重用。

include: 包含的文件不存在时，尽量执行下去。包含几次就重复执行几次文件的代码，速度比\_once快，推荐使用。

include\_once：包含的文件不存在时，尽量执行下去。和include的区别就是，无论包含多少次，都只执行一次。

require：包含的文件不存在时，停止执行，包含几次就重复执行几次文件的代码。

require\_once：包含的文件不存在时，停止执行，无论包含多少次，都只执行一次。

|  |
| --- |
| *// subtest1.php文件*  $a = 0; function fa(){  return $GLOBALS['a'] . '<br />'; }  ----------------------------------------------------------------------  *// subtest2.php文件*  $a++;  ----------------------------------------------------------------------  *// test.php 文件*  include('subtest1.php');  *//include('subtest2.php'); //include('subtest2.php'); //include('subtest2.php'); //echo fa(); // 结果：3  //include\_once('subtest2.php'); //include\_once('subtest2.php'); //include\_once('subtest2.php'); //echo fb(); // 结果：1  //require('subtest2.php'); //require('subtest2.php'); //require('subtest2.php'); //echo fa(); // 结果：3 //require('01.php'); // 导入文件不存在会发生致命错误* require\_once('subtest2.php'); require\_once('subtest2.php'); require\_once('subtest2.php'); echo fb(); *// 结果：1 //require('01.php'); // 导入文件不存在会发生致命错误* |

# 7 php错误报告

php的脚本错误分为多个等级，如致命错误、notice、warning等多个等级，如下图所示：



设置错误等级有两种方法，一是在配置文件php.ini修改选项error\_reporting，二是在程序开始时用函数error\_reporting。

注意：一般开发时报全部错误，上线时取消报错误，设置

error\_reporting(0)，因为报错会暴露具体网址路径，别人知道路径后容易攻击网址。

|  |
| --- |
| *error\_reporting*(***E\_ALL***); **echo** $a; *// Notice: Undefined variable: a* **include**(**'x.php'**);*// Warning: include(x.php): failed to open stream* **echo 'error\_reporting(E\_ALL)<hr /><br /><br />'**;  *error\_reporting*(***E\_ALL*** & ~***E\_NOTICE***);*// 取消Notice级别的错误。* **echo** $a; *// Notice: Undefined variable: a* **include**(**'x.php'**);*// Warning: include(x.php): failed to open stream* **echo 'E\_ALL & ~E\_NOTICE<hr /><br /><br />'**;  *error\_reporting*(***E\_ALL*** & ~***E\_WARNING***);*// 取消Warning级别的错误。* **echo** $a; *// Notice: Undefined variable: a* **include**(**'x.php'**);*// Warning: include(x.php): failed to open stream* **echo 'error\_reporting(E\_ALL & ~E\_WARNING)<hr /><br /><br />'**;  *error\_reporting*(0);*// 禁止报错* **echo** $a; *// Notice: Undefined variable: a* **include**(**'x.php'**);*// Warning: include(x.php): failed to open stream* **echo 'error\_reporting(0)<hr /><br /><br />'**; |

# 8 递归

递归的意思是先递后归。上一个递给下一个时，上一个操作暂停，等待下一个执行完毕返回，当递到最后一个，最后一个返回，逐级向上回归。所以递归函数必须要有结束条件，否则程序会进入死循环。

写递归函数技巧，首先写最外一层，然后判断符合自我调用条件，如果需要用到公共变量，需要用static声明变静态量。

## 递归实例1：f(n)=n+f(n-1) 从1累加到n

|  |
| --- |
| **function** sum($n){**if** ($n<=1){  **return** 1;  }  **return** $n+sum($n-1); } **echo** sum(100); |

## 递归实例2：用递归打印目录文件名和子目录。

|  |
| --- |
| **function** printItem($path, $lev = 0){  $dh = *opendir*($path);  **while** (($fn = *readdir*($dh)) !== **false**) {  **if** ($fn == **'.'** || $fn == **'..'**) { *// 忽略.和..* **continue**;  }  **echo** *str\_repeat*(**"&nbsp"**, $lev), $fn, **'<br />'**;  **if** (*is\_dir*($path . **'/'** . $fn)) {  printItem($path . **'/'** . $fn, $lev + 4);  }  } } printItem(**'.'**); |

## 递归实例3：计算数组所有单元之和

|  |
| --- |
| $arr = **array**(1,2,3,**array**(4,5,6,**array**(7,8,9))); **function** sumArr($arr){  **static** $sum = 0; *// 递归时用到公用变量，用static声明静态变量。*  **foreach** ($arr **as** $v){  **if** (*is\_array*($v)){  sumArr($v);  }**else**{  $sum += $v;  }  }  **return** $sum; } **echo** sumArr($arr); *// 结果：45* |

## 递归实例4：修改多维数组的单元值

如果是数字，将其修改为原来的2倍，在项目中有可能在POST、GET做递归安全转移时用到。

|  |
| --- |
| **function** chgArr(&$arr) *// 传入引用* {  **foreach** ($arr **as** $k => $v) {  **if** (*is\_int*($v)) {  $arr[$k] = $v \* 2;*// 修改引用值*  }  **if** (*is\_array*($v)) {  chgArr($v);  $arr[$k] = $v; *// 修改引用值*  }  } }  $arr = **array**(1, 2, **'A'**, **array**(3, 4, **'B'**), **array**(5, 6, **'C'**)); chgArr($arr); *print\_r*($arr);  *// 结果：Array ( [0] => 2 [1] => 4 [2] => A [3] => Array ( [0] => 6 [1] => 8 [2] => B ) [4] => Array ( [0] => 10 [1] => 12 [2] => C ) )* |

## 递归实例5：递归级联创建目录

如给定的目录"./2016/07/25",不存在则创建，项目中经常按年月日这种格式创建目录，让用户存放上传文件。

|  |
| --- |
| **function** mkDirs($dir){  **if** (!*is\_dir*($dir)){ *// 判断目录是否存在* $fdir=*dirname*($dir); *// 获取上一级目录，例如：./2016/07/25的上一级目录为./2016/07* **if** (!mkDirs($fdir)){ *// 递归，一直递归到根目录停止，因为根目录是存在的。* **return false**;  }  **if** (!*mkdir*($dir,0777)){ *// 创建目录，如果创建目录失败返回false* **return false**;  }  }  **return true**; }  $path=**'./2016/05/20'**; *var\_dump*(mkDirs($path)); |

## 递归实例6：删除目录下所有文件

例如给定路径"./test"，删除./test下所有文件和文件夹

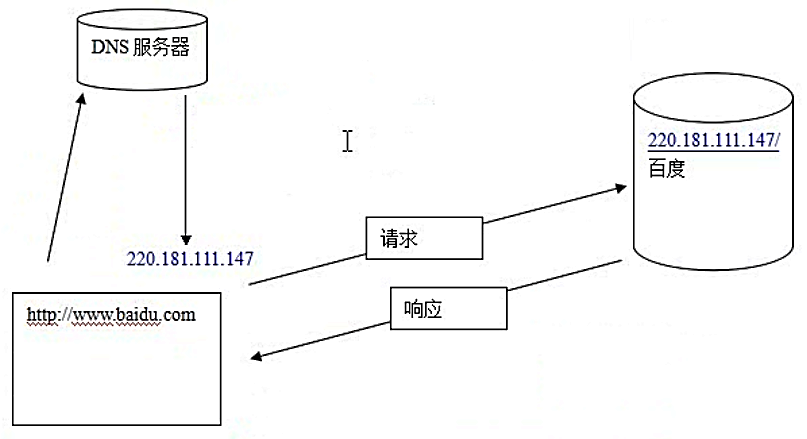
|  |
| --- |
| **function** delDir($path) {  **if** (!*is\_dir*($path)) {*// 判断目录是不是一个文件夹* **return null**;  }  $fh = *opendir*($path);  **while** (($row = *readdir*($fh)) !== **false**) {  **if** ($row == **'.'** || $row == **'..'**) {*//过滤掉虚拟目录* **continue**;  }   **if** (!*is\_dir*($path . **'/'** . $row)) {  *unlink*($path . **'/'** . $row);*// 删除文件* }  deldir($path . **'/'** . $row);  }  *closedir*($fh);*// 关闭目录句柄，否则出Permission denied* **if** (!*rmdir*($path)) {*// 删除文件之后再删除自身* **echo** $path . **'无权限删除<br>'**;  }  **return true**; }  $path = **'./2016'**; *var\_dump*(delDir($path)); |

## 递归实例7：无限级分类

|  |
| --- |
| $area = **array**(  **array**(**'id'** => 1, **'pid'** => 0, **'name'** => **'安徽省'**),  **array**(**'id'** => 2, **'pid'** => 0, **'name'** => **'浙江省'**),  **array**(**'id'** => 3, **'pid'** => 0, **'name'** => **'广东省'**),  **array**(**'id'** => 4, **'pid'** => 1, **'name'** => **'合肥市'**),  **array**(**'id'** => 5, **'pid'** => 4, **'name'** => **'长丰县'**),  **array**(**'id'** => 6, **'pid'** => 1, **'name'** => **'安庆市'**),  **array**(**'id'** => 7, **'pid'** => 2, **'name'** => **'杭州市'**),  **array**(**'id'** => 8, **'pid'** => 3, **'name'** => **'广州市'**),  **array**(**'id'** => 9, **'pid'** => 8, **'name'** => **'天河区'**) );  **function** getTree($arr, $pid, $lev = 0){  **static** $tree = **array**();  **foreach** ($arr **as** $k => $v) {  **if** ($v[**'pid'**] == $pid) {  $v[**'lev'**] = $lev;  $tree[] = $v;  *//unset($arr[$k]);*  *// 删除变量* getTree($arr, $v[**'id'**], $lev + 1); *// 找下一级* }  }  **return** $tree; }  $tree = getTree($area, 0); **foreach** ($tree **as** $v) {  **echo** *str\_repeat*(**'&nbsp'**, $v[**'lev'**]\*8), $v[**'name'**], **'<br/>'**; }  */\* 结果： 安徽省  合肥市  长丰县  安庆市 浙江省  杭州市 广东省  广州市  天河区  \*/* |

# 9 IP地址、域名和DNS

访问一个网站的过程：在浏览器输入域名，首先在本地是否找到该域名的IP地址，没找到就去DNS服务器去找，DNS服务器解析域名，把IP地址返回给浏览器，然后浏览器通过IP去访问网页。



# 10 window下安装apache

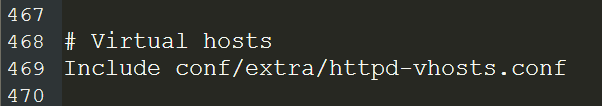
**(1)安装Apache**

安装完后在浏览器输入localhost，如果网页出现It works!，说明Apache启动成功了，如下图所示：



**(2)设置Apache配置文件httpd.conf**

在安装目录打开Apache2.2\conf\httpd.conf，找到Virtual hosts字段，去掉Include conf/extra/httpd-vhosts.conf前面的#号，就能支持虚拟主机的配置了，如下图所示：



**(3) 配置虚拟主机**

进入Apache2.2\conf\extra目录下，打开httpd-vhosts.conf文件，在文件后面追加下面内容：

|  |
| --- |
| <VirtualHost \*:80>  ServerAdmin webmaster@dummy-host.v.com  DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/www/www.yasen.com"  ServerName www.yasen.com  ServerAlias www.dummy-host.v.com  ErrorLog "logs/dummy-host.v.com-error.log"  CustomLog "logs/dummy-host.v.com-access.log" common  </VirtualHost> |

其中字段ServerName是域名，在浏览器输入的网址，字段DocumentRoot是访问的根目录。当浏览器输入域名时，Apache返回DocumentRoot目录下的index.html文件给浏览器，注意：DocumentRoot目录要真实存在，并且目录有index.html文件，如果不存在会报错403。

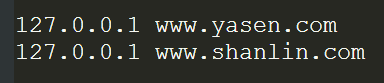
如果想设置多个虚拟主机，就重复复制上面代码，修改字段DocumentRoot和ServerName即可，例如下图设置了两个虚拟主机。



设置完毕重启Apache。

**(4)把域名解析到本地**

打开文件C:\WINDOWS\system32\drivers\etc\hosts，添加本地域名解析，如下图所示：



**(5)验证设置虚拟主机是否成功**

在浏览器上输入域名，如果出现403 Forbidden页面，如下图所示：



打开Apache配置文件，找到字段<Directory />，把deny from all修改为allow from all，保存，重启Apache。

刷新网页，如果能访问index.html文件（Apache默认），说明虚拟主机设置完成，例如下图所示：

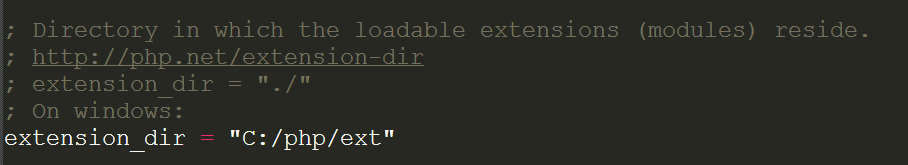


# 11 window下Apache和php整合

**(1)配置php.ini文件**

如果php安装目录下没有php.ini文件，复制一份php.ini-development文件，再把复件 php.ini-development文件名改为php.ini。

打开php.ini文件，找到字段extension\_dir，设置extension\_dir=php安装的目录下ext，如下图所示：

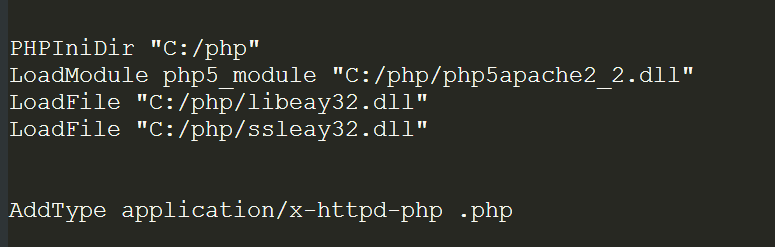


**(2)让Apache引入php解释引擎**

打开Apache配置文件httpd.conf，添加引入php解释器字段：

|  |
| --- |
| AddType application/x-httpd-php .php  PHPIniDir "C:/php"  LoadModule php5\_module "C:/php/php5apache2\_4.dll"  LoadFile "C:/php/ext/libeay32.dll"  LoadFile "C: /php/ext/ssleay32.dll" |

如下图所示：



字段AddType：让Apache碰到.php后缀文件就去找php解析器执行。

字段PHPIniDir：php安装目录。

字段LoadModule：Apache和php连接的库文件。

字段LoadFile：在Windows平台用到的两个库文件libeay32.dll、ssleay32.dll。

**(3)验证是否整合成功**

重启Apache软件，如果Apache启动失败，用终端查看失败的详细原因：进入Apache的bin目录，执行命令：httpd.exe -d start

在根目录新建一个info.php文件，输入内容：<?php phpinfo(); ?>

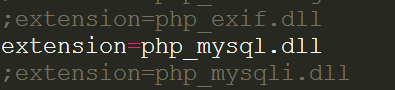
在浏览器输入：域名/info.php，如果能看到php运行结果，说明Apache和php整合成功，如果下图所示：



12 安装MySQL

安装完MySQL后，如果要想在php操作数据库，需要配置php.ini文件。

找到php\_mysql.dll，去掉前面的分号，如下图所示：



测试php是否成功连接MySQL，在根目录下新建mysql.php，输入内容

|  |
| --- |
| <?php  $sql= mysql\_connect('127.0.0.1','root','123456');  if ($sql){  var\_dump($sql);  mysql\_close($sql);  }  ?> |

在浏览器输入：域名/mysql.php，如果能没看到php运行错误，说明php连接数据成功，如果下图所示：



# 12 php操作mysql

在php5版本之前，一般是用php的mysql函数去驱动mysql数据库，例如mysql\_query()的函数，属于面向过程。在php5版本以后，增加了mysqli的函数功能，例如mysqli\_query()函数属于面向对象。某种意义上讲，它是mysql系统函数的增强版，更稳定更高效更安全。

两种驱动使用的官方文档：http://php.net/docs.php

**(1) mysql使用**

|  |
| --- |
| $conn = mysql\_connect('localhost', 'user', 'password');　//连接mysql数据库  $result = mysql\_query('select \* from data\_base'); // 执行sql语句  $row = mysql\_fetch\_assoc($result) // 返回数组，数组的key为表的列名  $row = mysql\_fetch\_array($result, MYSQL\_NUM) // 第二个参数决定输出的数组的key类型，MYSQL\_NUM(数字)，MYSQL\_ASSOC(列名) MYSQL\_BOTH(列名或数字)  $row = mysql\_fetch\_row($result) //只取一行数据  mysql\_close($conn) // 关闭连接 |

范例：

|  |
| --- |
| function mConn(){ // 初始化数据库，得到连接资源conn  static $conn=null;  if ($conn===null){  $conn=*mysql\_connect*("192.168.8.102","root","123456"); // 连接数据库  if (!$conn) {  echo 'Could not connect: ' . *mysql\_error*();  return null;  }  *mysql\_query*('use schedule',$conn); // 选择数据库  *mysql\_query*('set names utf8',$conn); // 设置字符编码  }  return $conn; }  $conn=mConn(); if ($conn===null){  return; }  $sql = "select \* from plan\_task where id<10";  $result=*mysql\_query*($sql,$conn); // 执行sql语句 while ($record = *mysql\_fetch\_assoc* ($result)){ // 输出结果  *print\_r* ($record); }  *mysql\_close*($conn); // 关闭连接 |

**(2) mysqli使用**

|  |
| --- |
| $conn = new mysqli('localhost', 'user', 'password','data\_base'); //连接数据库  $result = $conn -> query( 'select \* from data\_base' ); // 执行sql语句  $row = mysqli\_fetch\_assoc($result) // 返回数组，数组的key为表的列名  $row = mysqli\_fetch\_array($result, MYSQLI\_NUM) // 第二个参数决定输出的数组的key类型，MYSQLI\_NUM(数字)，MYSQLI\_ASSOC(列名) MYSQLI\_BOTH(列名或数字)  $row = mysqlI\_fetch\_row($result) //只取一行数据  $conn->close() // 关闭连接 |

范例：

|  |
| --- |
| function miConn(){ // 初始化数据库  static $conn=null;  if ($conn===null){  $conn=new mysqli("192.168.8.102","root","123456","schedule"); // 连接数据库  if ($conn->connect\_errno) {  echo "Connect failed: ".$conn->connect\_error;  return null;  }   $conn->query('set names utf8',$conn); // 设置字符编码  }  return $conn; }  $conn=miConn(); if ($conn===null){  return; }  $sql = "select \* from plan\_task where id<10"; $result = $conn->query($sql); while ($record = *mysqli\_fetch\_assoc* ($result)){  *print\_r* ($record); } $conn->close();// 关闭连接 |

# 13 面向对象

类是对一群有共同特征的事物或事件的集合，一种抽象的数据类型。

对象是真实世界中的实体，是类的实例。

例如人这个类，固有的熟悉有姓名、性别、年龄等属性，行为动作有喜、怒、哀、乐等方法。

类名不区分大小写，但是习惯要人为区分， 因为Linux严格区分大小写。

|  |
| --- |
| 类的语法：  声明类：class 类名  类属性：public 属性名  类方法：public(可省略) function 函数名  新建对象：new 类名(); // 内存开辟一块空间存放属性和方法名（不存放函数实体），当对象调用方法时，会直接调用类下面的方法。 |

在类的方法函数里要使用到类的属性时，需要用到伪变量this，注意this指向的不是类，指的是对象，谁实例化就指向谁。

|  |
| --- |
| **class** People{  *// 属性* **public $name** = **'刘备'**;*// 可以不赋值*  **public $gender** = **'男'**;  **public $age** = 45;   *// 方法* **public function** happy(){  **echo** $this->**name**, **"高兴了。<br>"**;  }   **public function** cry(){  **echo** $this->**name**, **"哭了。<br>"**;  } }  $p = **new** People(); *// 实例化* **echo** $p->**name**, $p->**gender**, $p->**age**, **"<br>"**; **echo** $p->happy(), $p->cry(); |