

PHP 基础

1. PHP 在 Web 开发中的应用
2. 第一个 PHP 脚本程序
3. PHP 语言标记
4. 指令分割符 “分号”
5. 程序注释
6. 在程序中使用空白符的处理
7. 变量
8. 变量的类型
9. 数据类型之间相互转换
10. 常量

一、PHP 语法

1、PHP 在 Web 开发中的应用

PHP(Hypertext Preprocessor 缩写)超级文本预处理器。

PHP 是一种在服务器端执行的嵌入 HTML 文档的脚本语言。

PHP 是目前最流行的网站开发语言 (B/S 结构) 。

PHP 特点：

是开放源代码的，服务器端的脚本语言。

独立于操作系统，可以运行在几乎所有系统中。

支持大部分的服务器，如 apache,IIS

支持大量的数据库

可以创建图象

还有一些其他功能在后面的高级技术详细介绍。

2、第一个 PHP 脚本语言

PHP 的开发步骤：

使用编辑器创建一个包含源代码的磁盘文件

将文件上传到 web 服务器上

通过浏览器访问 Web 服务器运行程序

3、PHP 语言标记

1、我们用<?php 来表示 PHP 标识符的起始，然后放入 PHP 语句并通过加上一个终止标识符?>来退出 PHP 模式。可以根据自己的需要在 HTML 文件中像这样开启或关闭 PHP 模式。大多数的嵌入式脚本语言都是这样嵌入到 HTML 中并和 HTML 一起使用，例如 CSS、JavaScript、PHP、ASP 以及 JSP 等。



2、PHP 语言嵌入 HTML 中的位置



3、使用不同的四对标记

- 》以<?php 开始和以?>结束标记是标准风格，这是 PHP 推荐使用的标记风格。
- 》以<script language="php">开始和<script>结束是长风格标记，这种标记最长，总是可用的，但我们并不常用。
 - 》以<?开始和以?>结束标记是简短风格的标记，是最简单的，但是系统管理员偶尔会禁用掉它，因为它会干扰 XML 文档的声明。只用通过 php.ini 配置文件中的指令 short_open_tag 打开后就可以使用。
 - 》以<%开始和以%>结束标记是 ASP 风格的标记，可以在 php.ini 配置文件设定中启用 asp_tags 选项就可以使用它（默认是禁用的），习惯了 ASP 风格的可以使用它。

4. 指令分割符”分号”

一种是在程序中使用结构定义语句例如流程控制、函数与类的定义等，是用大括号来标记代码块，在大括号后面不要用分号。

另一种是在程序中使用功能执行语句，如变量的声明、内容的输出、函数的调用等，是用来在程序中执行某些特定功能的语句，这种语句也可称为指令，PHP 需要在每个指令后用分号结束。

和其他语言不一样的是，在 PHP 中右括号(?)前的分号不是必选的。

5. 程序注释

对于阅读代码的人来说，注释其实就相当于代码的解释和说明。注释可以用来解释脚本的用途、脚本编写人、为什么要按如此的方法编写代码、上一次修改的时间等等。

PHP 支持 C、C++ 和 Shell 脚本风格的注释，如下：

```
//... ... 单行注释
/* ... ... */多行注释      (注意：不能嵌套)
# ... ... 脚本注释
```

程序员在编程时使用注释是一种良好的习惯，优点：

写帮助文档

调试程序

注意： 注释要写在代码的上面或是右边

6. 在程序中使用空白的处理

一般来说，空白符（空格、Tab制表符、换行）在 PHP 中无关紧要。可以将一个语句展开成任意行，或者将语句紧缩在一行。

使用两个空行

一个源文件的两个代码段

两个类的声明

在以下情况使用一个空行

两个函数声明之间

函数内的局部变量和函数的第一条语句之间

注释或者单行注释之前

一个函数的两个逻辑代码段

7、变量

变量的声明

变量的命名

可变变量

变量的引用赋值

1、变量的声明

变量是用于临时存储值的容器。这些值可以是数字、文本、或者复杂得多的排列组合。是用于跟踪几乎所有类型信息的简单工具。

PHP 是一种非常弱的类型语言。

2、PHP 变量的声明：

注意：

是以\$符开始的，后面跟大小写字母，数字和下划线，但不能以数字开头。

(字母数字下划线但不能以数字开头作为变量名)

可以使用函数 `unset()` 释放指定的变量，`isset()` 函数检测变量是否设置，`empty()` 函数检查一个变量是否为空。

`isset()` 是判断变量是否存在，是否定义 `empty()` 是判断变量的值是否为空

3、变量的命名

变量名与 PHP 中其它的标签一样遵循相同的规则。一个有效的变量名由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。

变量的名称是对大小写敏感的。

但内置结构和关键字以及用户自定义的类名和函数名都是不区分大小写的。如：`echo`、`while`、`function` 名称等。

```
<?php
echo "this is a test";
Echo "this is a test";
$name="tarzan";
$Name="skygao";
echo $name.$Name; //输出: tarzanskygao
?>
```

这两行输出是一样的

这是两个不同的变量

8、变量的类型

类型介绍

标量	布尔型(boolean) 整型(integer) 浮点型(float 或 double) 字符串(String)
复合	数组(Array) 对象(Object)
特殊	资源类型(Resource) NULL 类型 伪类型介绍

布尔型(boolean)

这是最简单的类型。boolean 表达了真值，可以为 TRUE 或 FALSE，即“真”或“假”。

当其他类型转换为 boolean 类型时，以下值被认为是 FALSE：

布尔值 FALSE

整型值 0 (零)

浮点型值 0.0 (零)

空白字符串和字符串"0"

没有成员变量的数组

没有单元的对象（仅适用于 PHP 4）

特殊类型 NULL（包括尚未设定的变量）

所有其它值都被认为是 TRUE（包括任何资源）。

整型(integer)

整型值可以用十进制，十六进制或八进制符号指定，前面可以加上可选的符号（- 或者 +）代表数值的正负。（最左边的位数，0 表示正数，1 表示负数）充当符号位

//php 中整数是 4 个字节 (32 位的二进制)

//1 字节=8 位 (8 个 0 到 8 个 1) 1024 字节=1k 1024k=1M

//十进制 256 个

//\$a=123; //将 123 整数赋给 a 变量

//\$a=0123; //将八进制的 0123 整数赋给 a 变量

//\$a=0x123; //将十六进制的 0X123 整数赋给 a 变量

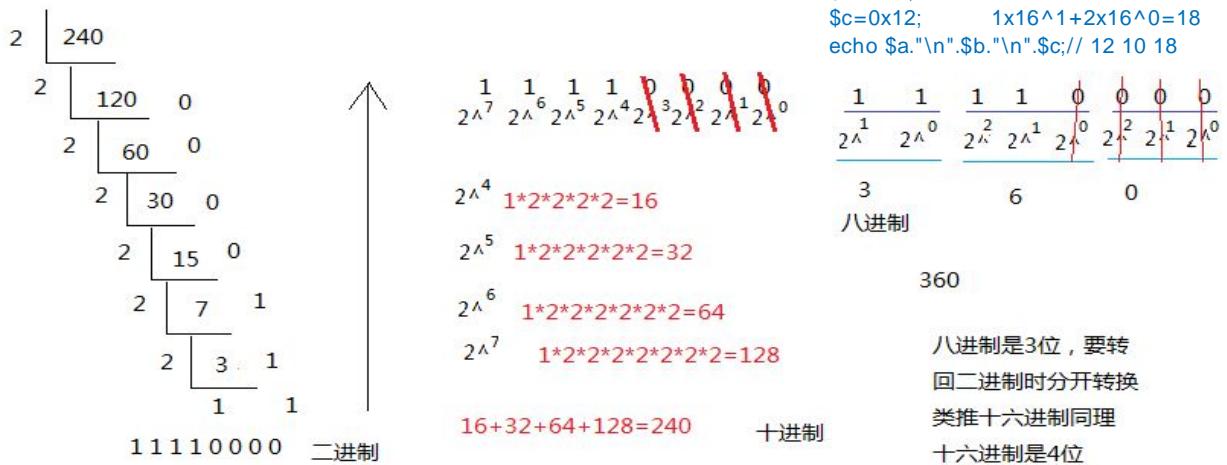
//echo 10; //10

//echo 010; //8 (八进制)

//echo 0X10; //16 (十六进制)

整数值有最大的使用范围，这与平台有关，对于 32 位系统而言范围：-2147483648~2147483647,PHP 不支持无符号整数。如果超出了则变成了 float 型。

进制之间转换:



浮点型(float 或 double)

浮点数（也叫双精度数或实数）是包含小数部分的数。通常用来表示整数无法表示的数据，如金钱值、距离值、速度值等。浮点数的字长和平台相关，尽管通常最大值是 1.8e308 并具有 14 位十进制数字的精度。

注意事项：例：floor((0.1+0.7)*10) 通常会返回 7 而不是预期中的 8，因为该结果内部的表示其实是 7.9。就是不可能精确的用有限位数表达某些十进制分数。所以永远不要相信浮点数结果精确到了最后一位，也永远不要比较两个浮点数是否相等。如果确实需要更高的精度，应该使用任意精度数学函数或者 gmp 函数。 floor 向下取整函数

字符串(String)

字符串的定义

string 是一系列字符。在 PHP 中，字符和字节一样，也就是说，一共有 256 种不同字符的可能性。这也暗示 PHP 对 Unicode 没有本地支持。

注：一个字符串变得非常巨大也没有问题，PHP 没有给字符串的大小强加实现范围，所以完全没有理由担心长字符串。

语法：

字符串可以用三种字面上的方法定义：

单引号 ''

双引号 ""

定界符 <<<

单引号：指定一个简单字符串的最简单的方法是用单引号（字符 '）括起来。

》要表示一个单引号，需要用反斜线 (\) 转义，和很多其它语言一样。如果在单引号之前或字符串结尾需要出现一个反斜线，需要用两个反斜线表示。注意如果试图转义任何其它字符，反斜线本身也会被显示出来！所以通常不需要转义反斜线本身。

》注：单引号字符串中出现的变量不会被变量的值替代。

```

<?php
echo 'this is a simple string';           //输出: this is a simple string
echo 'this is a \'simple\' string';        //输出: this is a 'simple' string
echo "this \n is \r a \t simple string\\";   //输出: this \n is \r a \t simple string\
$str=100;
echo "this is a simple $str string";      //输出: this is a simple $str string
?>

```

双引号: 如果用双引号 ("") 括起字符串, PHP 懂得更多特殊字符的转义序列:

注: 双引号字符串最重要一点是其中的变量名会被变量值替代。此外, 如果试图转义任何其它字符, 反斜线本身也会被显示出来! 转义字符如下表所示:

序列	含义
\n	换行 (LF 或 ASCII 字符 0x0A (10))
\r	回车 (CR 或 ASCII 字符 0x0D (13))
\t	水平制表符 (HT 或 ASCII 字符 0x09 (9))
\\"	反斜线
\\$	美元符号
\"	双引号
\[0-7] {1, 3}	此正则表达式序列匹配一个用八进制符号表示的字符
\x[0-9A-Fa-f] {1, 2}	此正则表达式序列匹配一个用十六进制符号表示的字符

在 php 中字串的定义方式:

单引号: '' 不支持变量解析, 支持单引号和\的转义。

双引号: "" 支持变量解析, 为了兼容使用{}将变量括起来。

支持转义: \n \s \t \" \\ \\$

定界符: <<< 注意结束符的使用

定界符: 另一种给字符串定界的方法使用定界符语法 (“<<<”）。应该在 <<< 之后提供一个标识符, 然后是字符串, 然后是同样的标识符结束字符串。

定界符中标识符的命名规则与变量的命名规则一样。只能包含字母数字下划线, 而且必须以下划线或非数字字符开始。

》注: 结束标识符所在的行不能包含任何其它字符, 可能除了一个分号 (;) 之外。这尤其意味着该结束标识符不能被缩进, 而且在分号之前和之后都不能有任何空格或制表符。如果破坏了这条规则使得结束标识符不“干净”, 则它不会被视为结束标识符, PHP 将继续寻找下去。如果在这种情况下找不到合适的结束标识符, 将会导致一个在脚本最后一行出现的语法错误。

》不能用定界符语法初始化类成员。用其它字符串语法替代。

》定界符文本的表现和双引号字符串一样, 只是没有双引号

实例:

```

<?php
header('Content-type:text/html;charset="utf-8"');
//字符串
//定义字符串的方式有三种: 单引号、双引号、定界符
$a=10;

```

```

$sl = 'hello world!$a'; //不支持变量解析
$s2 = "hello world!$a"; //支持变量解析

echo $sl;
echo "<br/>";
echo $s2;
echo "<br/>";
echo "$acm"; //变量名默认使用最长字符名
echo "$a cm".<br>; //变量名中包括字母,数字,下划线但不与数字开头,空格不解释
echo "{$a}cm"; //为了防止变量名混淆, 建议使用{}包括起来

echo "<h3>字串中的转义字符:\t\r\n\\\'\\\"\\$\n</h3>";

echo '单引号支持: \\\' 不支持: \t\r\n<br/>;//本身不能嵌套需要时必须转义
echo "双引号全支持: \\\" \t\r\n<br/>";

echo "<h3>定界符: <<<</h3>";

$str=<<<mystr
    asdflkja
    asdf;ljasdglkjaegrmystr
        ;adsjgasdfasdf{$a}d
mystr;
mystr;
echo $str;
输出结果:

hello world!$a
hello world!10

```

(!) Notice: Undefined variable: acm in D:\wamp\wamp\www\test.php on line 10			
Call Stack			
#	Time	Memory	Function
1	0.0000	143368	{main}()
10 cm			
10cm			

字串中的转义字符:\t \r \n \\ \' \\ \" \\\$

单引号支持: \\\' 不支持: \t\r\n
双引号全支持: \\\"

定界符: <<<

asdflkja asdf;ljasdglkjaegrmystr ;adsjgasdfasdf10d mystr;

数组(Array)

》 PHP 中的数组实际上是一个有序图。图是一种把 values 映射到 keys 的类型。

》 PHP 中可以使用多种方式构建一个数组，在这里我们只用 array() 语言结构来新建一个 array。它接受一定数量用逗号分隔的 key => value 参数对。

语法结构：

```
array( [key =>] value , ... )  
// key 可以是integer或者string类型  
// value 可以是任何值
```

```
<?php
```

```
$arr = array(  
    "foo" => "bar", 12 => true);  
?>
```

对象(Object)

在 PHP 中，对象和数组一样都是一种复合数据类型。但对象是一种更高级的数据类型。一个对象类型的变量，是由一组属性值和一组方法构成，其中属性表明对象的一种状态，方法通常用来表明对象的功能。

```
<?php  
class foo{           //类的定义  
    function do_foo(){ //类中方法的定义  
        echo "Doing foo.";  
    }  
}  
$bar = new foo;      //初始化类foo创建一个对象bar  
$bar->do_foo();    //通过对象bar调用方法do_foo输出： Doing foo.  
?>
```

资源类型(Resource)

资源是一种特殊变量，保存了到外部资源的一个引用。资源是通过专门的函数来建立和使用的。由于资源类型变量保存有为打开文件、数据库连接、图形画布区域等的特殊句柄，因此无法将其它类型的值转换为资源，无法长时间维持一种状态，什么时候使用什么时候就打开，用完了立即断开。类似于拨打电话不聊就挂断不然形成浪费资源。

```
<?php  
$file_handle=fopen("info.txt","w");  
var_dump($file_handle);           //resource(3) of type (stream)  
$link_mysql=mysql_connect("localhost","root","root");  
var_dump($link_mysql);           //resource(4) of type (mysql link)  
?>
```

NULL 类型

特殊的 NULL 值表示一个变量没有值。NULL 类型唯一可能的值就是 NULL，表示一个变量的值为空，NULL 不区分大小写。

在下列情况下一个变量被认为是 NULL：

被赋值为 NULL 值的变量。

尚未被赋值的变量。

被 unset() 函数销毁的变量。

伪类型介绍

伪类型并不是 PHP 语言中的基本数据类型，作为描述性类型。

mixed: 说明一个参数可以接受多种不同的（但并不必须是所有的）类型。

number: 说明一个参数可以是 integer 或者 float。

callback: 有些诸如 call_user_function() 或 usort() 的函数接受用户自定义的函数作为一个参数。

Callback 函数不仅可以是一个简单的函数，它还可以是一个对象的方法，包括静态类的方法

一个 PHP 函数用函数名字符串来传递。可以传递任何内置的或者用户自定义的函数，除了 array(), echo(), empty(), eval(), exit(), isset(), list(), print() 和 unset()。

9、数据类型之间相互转换

》自动转换 对运算时自动转化成合适类型

字符串转换为数值取决于引号最左字符，遇到若字符串则后面的字符全部舍去。`echo 1+2+'3+4+5'; //6`

》强制转换

PHP 中的类型强制转换和 C 中的非常像：在要转换的变量之前加上用括号括起来的目标类型。

允许的强制转换有：

(int), (integer) - 转换成整型

(bool), (boolean) - 转换成布尔型

(float), (double), (real) - 转换成浮点型

(string) - 转换成字符串

(array) - 转换成数组

(object) - 转换成对象

注意在括号内允许有空格和制表符，为了将一个变量还原为字符串，还可以将变量放置在双引号中。

变量类型的测试函数：

is_bool(): 判断是否是布尔型

is_int()、**is_integer()**和**is_long()**: 判断是否为整型。

is_float()、**is_double()**和**is_real()**: 判断是否为浮点型

is_string(): 判断是否为字符串

is_array(): 判断是否为数组

is_object(): 判断是否为对象

is_resource(): 判断是否为资源类型

is_null(): 判断是否为 null

is_scalar(): 判断是否为标量

is_numeric(): 判断是否是任何类型的数字和数字字符串

is_callable(): 判断是否是有效的函数名

函数：

Bool settype (mixed var, string type)

是将变量 var 的类型设置成 type。

10、常量

常量的定义与使用

常量与变量

预定义常量

1. 常量

》常量是一个简单值的标识符（名字）。如同其名称所暗示的，在脚本执行期间一个常量一旦被定义，就不能再改变或取消定义。常量默认为大小写敏感。按照惯例常量标识符总是大写的。

》常量名和其它任何 PHP 标签遵循同样的命名规则。合法的常量名以字母或下划线开始，后面跟着任何字母，数字或下划线。

》常量的范围是全局的。不用管作用域就可以在脚本的任何地方访问常量。

》我们可以用 `define()` 函数来定义常量。

2. 常量的定义与使用

使用 `define()` 函数来定义常量。一个常量一旦被定义，就不能再改变或者取消定义。

重复定义也是无效的。

```
<?php
    define("CON_INT",100);
    echo CON_INT;                                //输出: 100
    define("GREETING","Hello you",true);
    echo GREETING;                               //输出: Hello you
    echo constant ("Greeting");                  //输出: Hello you
?>
```

3. 常量与变量

常量和变量不同：

常量前面没有美元符号（\$）；

常量只能用 `define()` 函数定义，而不能通过赋值语句；

常量可以不用理会变量范围的规则而在任何地方定义和访问；

常量一旦定义就不能被重新定义或者取消定义；

常量的值只能是标量。

可以用函数 `constant()` 来读取常量的值。

用 `get_defined_constants()` 可以获得所有已定义的常量列表

4. 预定义常量

名称	说明
<code>__LINE__</code>	文件中的当前行号。
<code>__FILE__</code>	文件的完整路径和文件名。如果用在包含文件中，则返回包含文件名。自 PHP 4.0.2 起， <code>__FILE__</code> 总是包含一个绝对路径，而在此之前的版本有时会包含一个相对路径。
<code>__FUNCTION__</code>	函数名称（PHP 4.3.0 新加）。自 PHP 5 起本常量返回该函数被定义时的名字（区分大小写）。在 PHP 4 中该值总是小写字母的。
<code>__CLASS__</code>	类的名称（PHP 4.3.0 新加）。自 PHP 5 起本常量返回该类被定义时的名字（区分大小写）。在 PHP 4 中该值总是小写字母的。
<code>__METHOD__</code>	类的方法名（PHP 5.0.0 新加）。返回该方法被定义时的名字（区分大小写）。

二、PHP 中的运算符

算数运算符
字符串运算符
赋值运算符
比较运算符
逻辑运算符
位运算符（二进制操作）
其他运算符
运算符的优先级

1. 定义运算符 双引号中是无法进行运算操作的，是可以使用切分引号连接方法解决

运算符是可以通过给出的一或多个值（用编程行话来说，表达式）来产生另一个值（因而整个结构成为一个表达式）的东西。所以可以认为函数或任何会返回一个值（例如 print）的结构是运算符，而那些没有返回值的（例如 echo）是别的东西。

有三种类型的运算符：（几个操作就叫几元运算符）

一元运算符，只运算一个值，例如 !（取反运算符）或 ++（加一运算符）。(!false), \$a++

二元运算符，有两个操作数，PHP 支持的大多数运算符都是这种。(10+20)

三元运算符：?:。它应该被用来根据一个表达式在另两个表达式中选择一个，而不是用来在两个语句或者程序路线中选择。把整个三元表达式放在扩号里是个很好的主意。

运算符

运算符	意义	示例	结果
+	加法运算	\$a+\$b	\$a 和\$b 的和
-	减法/取负运算	\$a-\$b	\$a 和\$b 的差
*	乘法运算	\$a*\$b	\$a 和\$b 的积
/	除法运算	\$a/\$b	\$a 和\$b 的商
%	求余运算符(取模运算)	\$a%\$b	\$a 和\$b 的余数
++	累加 1	\$a++ 或 ++\$a	\$a 的值加 1
--	递减 1	\$a--	

++\$a 先++运算再赋值给其他变量，\$a++ 先赋值给其他变量再进行++运算。

\$a = 10; \$b = 10 + \$a++; echo \$a."：“.\$b 结果： 11:20

常用的求余运算符

\$num = rand(0,10000); //随机一个 0 到 10000 的数

echo \$num%10; //将 num 数值对 10 取模获得 0~9 的随机数

echo (\$num%51)+50; //根据 num 的值随机出一个 50 至 100 的数。

2.字符串运算符

有两个字符串运算符：

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。

```
<?php  
    $a = "Hello ";  
    $b = $a . "World!";      // 现在$b的值: Hello World!  
  
    $a = "Hello ";  
    $a .= "World!";        // 现在$a的值: Hello World!  
?>
```

3.赋值运算符

基本的赋值运算符是“=”。一开始可能会以为它是“等于”，其实不是的。它实际上意味着把右边表达式的值赋给左边的运算数。

运算符	意义	示例
=	将一个值或表达式的结果赋给变量	\$x=3
+=	将变量与所赋的值相加后的结果赋给该变量	\$x+=3 等价于\$x=\$x+3
-=	将变量与所赋的值相减后的结果赋给该变量	\$x-=3 等价于\$x=\$x-3
=	将变量与所赋的值相乘后的结果赋给该变量	\$x=3 等价于\$x=\$x*3
/=	将变量与所赋的值相除后的结果赋给该变量	\$x/=3 等价于\$x=\$x/3
%=	将变量与所赋的值求模后的结果赋给该变量	\$x%=3 等价于\$x=\$x%3
.=	将变量与所赋的值相连后的结果赋给该变量	\$x .= "H" 等价于\$x=\$x. "H"

4.比较运算符（字符串按照 ASCII 码值规则比较）

运算符	描述	说明	示例
>	大于	当左边大于右边时返回 true，否则返回 false	\$a>\$b
<	小于	当左边小于右边时返回 true，否则返回 false	\$a<\$b
>=	大于等于	当左边大于等于右边时返回 true，否则 false	\$a>=\$b
<=	小于等于	当左边小于等于右边时返回 true，否则 false	\$a<=\$b
==	等于	两边操作数的值相等时返回 true，否则 false	\$a==\$b

string **chr** (int \$ascii) chr — 返回指定的字符
int **ord** (string \$string) ord — 返回字符的 ASCII 码值

==	全等于	两边值相等并且类型相等返回 true, 否则 false	\$a==\$b
<>或!=	不等于	两边值不等时返回 true, 否则返回 false	\$a>\$b \$a!=\$b
!==	非全等于	两边值与类型都相同时返回 false, 否则 true	\$a!==\$b

输出所有 ASCII 码值

```
for($i=1; $i<=128;$i++){
    printf("%d=>%c<br/>",$i);
}
```

5.逻辑运算符

运算符	描述	说明	示例
and 或 &&	逻辑与	当两边操作数都为 true 时, 返回 true, 否则返回 false	\$a and \$b \$a && \$b
or 或 	逻辑或	当两边操作数都为 false 时, 返回 false, 否则返回 true	\$a or \$b \$a \$b
not 或 !	逻辑非	当操作数为 true 时返回 false, 否则返回 true (一元运算符, 一个操作数)	not \$b !\$b
xor	逻辑异或	当两边操作数只有一个为 true 时, 返回 true, 否则返回 false	\$a xor \$b

//逻辑短路与

```
$a=10;
if($a>20 && ++$a){
}
echo $a; //表达式前面满足条件就不会继续往后执行 // 10
```

//逻辑短路或

```
$a=10;
if($a>5 || ++$a){
}
echo $a; // 10
```

```
<?php
$a=5;
$b=6;
if($a=6 || $b=7){ /* 逻辑||运算符优先赋值=, 逻辑运算符返回true/false同时
逻辑运算符有短路现象所以$b=7赋值运算是不会执行*/
    echo $a.'--';//true 在输出自动转换为1
    $a++; //bool类型 true不能进行自增运算
    echo $a.'--';
    $b++;
    echo $b.'--';
}
echo $a.'-'.$b;//输出1--1--7--1|7
```

//用法: 先判断是否登陆再进行判断是否拥有权限 同时应用数组输出先有长度值再输出

```
if(已登录 && 拥有权限){
    //通行
} else{
    //报错: 你没有登录或没有权限
}
```

```
$a = 3; //赋值
$b = 5;
if ($a=5 || $b = 7)//运算符优先级 逻辑运算符(||)高于赋值运算符(=) , 即$a = (bool)ture;$b=5;
{
    $a++; // $a++ 值不变为1
    $b++; // $b++ 值为5+1
}
echo $a . " " . $b;
var_dump($name);可以看到变量类型 (逻辑运算符返回的是布尔值)
```

6.位运算符（将值先转换为二进制再进行运算）

运算符	描述	说明	示例
&	按位与	只有参与运算的两位都为 1 时，运算结果才为 1，否则为 0.	\$a & \$b
	按位或	只有参与运算的两位都为 0 时，运算结果才为 0，否则为 1.	\$a \$b
^	按位异或	只有参与运算的两位不同，运算结果才为 1，否则为 0.	^\$b
~	按位非	将用二进制表示的操作数中的 1 变成 0，0 变成 1.	~\$a
<<	左移	将左边的操作数在内存中的二进制数据右移右边操作数指定的位数，右边移空的部分补上 0	\$a<<\$b
>>	右移	将左边的操作数在内存中的二进制数据左移右边操作数指定的位数，左边移空的部分补上 0	\$a>>\$b

4 位二进制刚好满足 10 数值的十进制表示

十进制	二进制	位与运算 1&3 //1	位或运算 3 4 //7	按位取反	echo ~7; echo '<hr/>'; printf("%b",-8);
0	0000	0001	0011		
1	0001				
2	0010	0011	0100	-8	
3	0011	&			
4	0100				
5	0101	0001	0111	11111111111111111111111111000	
6	0110				
7	0111				
8	1000				
9	1001				
10	1010				

按位左移	按位右移	三元运算符	\$a = 10; \$b = 20; echo (\$a>\$b)?\$a:\$b; 表达式 ? 真 : 假
echo 1<<2 //4	echo 8>>2 //2		

echo isset(\$name)? "你好！{\$name}": "你好！游客" ;
--

7.其他运算符

运算符	描述	示例
? :	三元运算符，可以提供简单的逻辑判断。	\$a<\$b?\$c=1:\$c=0
``	反引号(` `)是执行运算符，PHP 将尝试将反引号中的内容作外壳命令来执行，并将其输入信息返回	\$a=` ls -al`
@	错误控制运算符，当将其放置在一个 PHP 表达式之前，该表达式可能产生的任何错误信息都被忽略掉。	@表达式

=>	数组下标指定符号，通过此符号指定数组的键与值。	键=>值
->	对象成员访问符号，访问对象中的成员属性或成员方法。	对象->成员
instanceof	类型运算符，用来测定一个给定的对象是否来自指定的对象类。	对象 instanceof 类名

8. 运算符的优先级（利用小括号）

表格 15-1. 运算符优先级

结合方向	运算符	附加信息
非结合	<code>new</code>	new
左	<code>[</code>	array()
非结合	<code>++ --</code>	递增 / 递减运算符
非结合	<code>! ~ - (int) (float) (string) (array) (object) @</code>	类型
左	<code>* / %</code>	算数运算符
左	<code>+ - .</code>	算数运算符和字符串运算符
左	<code><< >></code>	位运算符
非结合	<code>< <= > >=</code>	比较运算符
非结合	<code>== != === !==</code>	比较运算符
左	<code>&</code>	位运算符和引用
左	<code>^</code>	位运算符
左	<code> </code>	位运算符
左	<code>&&</code>	逻辑运算符
左	<code> </code>	逻辑运算符
左	<code>? :</code>	三元运算符
右	<code>= += -= *= /= .= %= &= = ^= <<= >>=</code>	赋值运算符
左	<code>and</code>	逻辑运算符
左	<code>xor</code>	逻辑运算符
左	<code>or</code>	逻辑运算符
左	<code>,</code>	多处用到

左联表示表达式从左向右求值，右联相反。

三、流程控制概述

任何 PHP 脚本都是由一系列语句构成的。一条语句可以是一个赋值语句，一个函数调用，一个循环，甚至一个什么也不做的（空语句）条件语句。语句通常以分号结束。此外，还可以用花括号将一组语句封装成一个语句组。语句组本身可以当作是一行语句。本章讲述了各种语句类型。

在任何一门程序设计语言中，都需要支持满足程序结构化所需要的三种基本结构：

顺序结构

分支结构（选择结构）

循环结构

顺序结构：在程序结构中，最基本的就是顺序结构。程序会按照自上而下的顺序执行。由于结构简单所以这里我就不多介绍

（一）分支结构

单一条件分支结构(if)

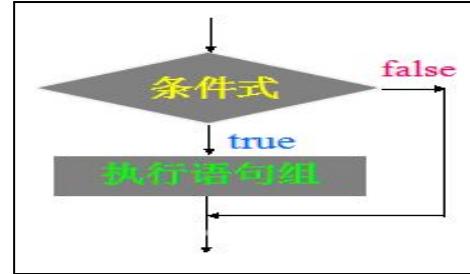
双向条件分支结构(else 从句)
多向条件分支结构(elseif 子句)
多向条件分支结构(switch 语句)
巢状条件分支结构

1、单一条件分支结构(if)

单个 if 语句：

基本格式：

```
if(条件表达式){  
    语句组;  
    //语句组为单条语句时可省略“{}”。  
}
```



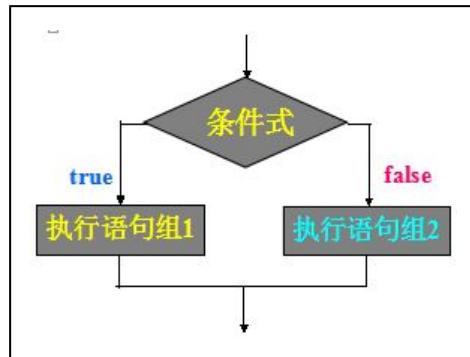
当条件表达式的值为真 (true) 时，PHP 将执行语句组，相反条件表达式的值为假 (false) 时，PHP 将不执行语句组，忽略语句组执行下面的语句。

2、双向条件分支结构(else 从句)

if…else 语句：（范围性判断）

格式如下

```
if(条件表达式){  
    语句组 1  
}else{  
    语句组 2  
    //语句组为单条语句时可省略“{}”。  
}
```



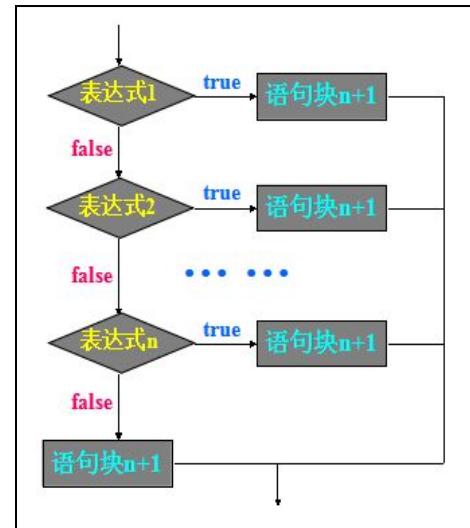
if-else 条件判断与 if 条件判断类似，所不同的是，if-else 语句的条件表达式值为真(true)时，会执行 if 的本体语句(语句组 1)，而条件表达式值为假(false)时，则执行 else 的本体语句(语句组 2)。

3、多向条件分支结构(elseif 子句)

elseif 子句：

格式如下

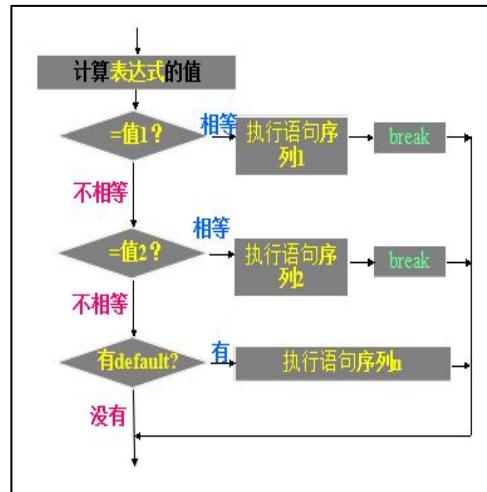
```
if(条件表达式 1){  
    语句块 1  
}elseif(条件表达式 1){  
    语句块 2  
}... ...  
}elseif(条件表达式 n){  
    语句块 n  
}else{  
    语句块 n+1  
}
```



4、多向条件分支结构(switch 子句)

switch-case 语句语法：（具体某一个值判断）

```
switch (表达式) {  
    case 值 1:  
        语句序列 1; break;  
    case 值 2:  
        语句序列 2; break;  
    ...  
    default:  
        语句序列 n; break;  
}
```



当程序执行碰到 switch 条件判断时，它会取出键值，然后与语句体中的 case 所列出的值逐一比较，如果数值不符合，则将数值往下一个 case 传递，如果数值符合，则执行 case 中的语句，然后再碰到 break 语句即跳出 switch 条件判断，如果所有的值比对都不符合，则会执行 default 中的语句。

switch 语句使用注意事项：

switch 语句与 if 语句不同，它仅能判断一种关系：是否恒等。

switch 语句中 case 子句的常量可以是整型常量、字符型常量、表达式或变量。

在同一个 switch 中，case 子句的常量不能相同，否则第二个值永远无法匹配到。

case 和 default 子句后面的语句序列允许由多个可执行语句组成，且不必用“{}”括起来，也可以为空语句。

switch 语句中可省略 break 语句和 default 子句。但省略后会改变流程。

例子：

```
switch($i) { //条件表达式是一个变量$i  
    case 2: //和值 2 匹配时，没有 break，将控制转移到下一个 case 中的语句  
    case 3: //和值 3 匹配时，执行下面的语句块  
        echo "$i 和值 2 或 3 任一个匹配";  
        break; //退出 switch 语句  
    case 4: //和值为 3 匹配上时，执行下面的语句块  
        echo "$i 和值 4 匹配时，才会执行";  
        break; //退出 switch 语句  
    default: //匹配任何和其他 case 都不匹配的情况，要放在最后一个 case 之后  
        echo "$i 没有匹配的值时，才会执行";  
}
```

5、巢状条件分支结构

◆ 语法：

```
if(表达式1){  
    if(表达式2){  
        ...  
    } else{  
        ...  
    } else{  
        if(表达式3){  
            ...  
        }  
    }  
}
```

◆ 巢状条件分支结构就是if语句的嵌套，即指if或else后面的语言块中又包含if语句。if语句可以无限层地嵌套在其他if语句，这给程序的不同部分的条件执行提供了充分的弹性。

(二) 循环结构

while 语句

do...while 循环

for 语句

特殊的流程控制语句

1、while 循环语法：（数据迭代）

```
while(表达式){  
    语句或语句序列  
    ....  
}
```

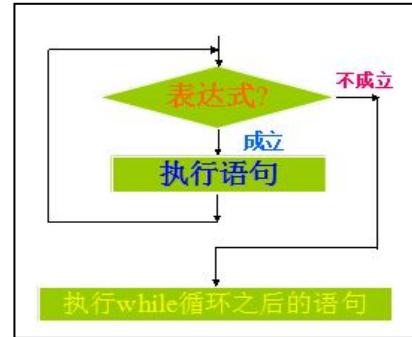
当 while 循环语句中表达式（循环控制语句）的结果为真时，程序将反复执行同一段程序：循环体（while 中的语句或语句序列），直到满足一定的条件（表达式的结果为假时）后才停止执行该段程序。

擅长循环迭代的数据（未知容器里面不断取出直到没有为止）

2、while 循环实例

```
<?php  
    //求1到100的累加。  
    $i=1;  
    $sum=0;  
    while($i<=100)  
    {  
        $sum+=$i;  
        $i++;  
    }  
    echo $sum;  
?>
```

```
<html>  
    <head><title>使用while循环嵌套输出表格</title></head>  
    <body>  
        <table align="center" border="1" width=600>  
            <caption><h1>使用while循环嵌套输出表格</h1></caption>  
            <?php  
                $out = 0; //外层循环需要计数的累加变量  
                while( $out < 10 ) { //指定外层循环，并且循环次数为10次  
                    $bgcolor = $out%2 == 0 ? "#FFFFFF" : "#DDDDDD";  
                    echo "<tr bgcolor=\"$bgcolor\">"; //执行一次则输出一行并指定背景颜色  
                    $in = 0; //内层循环需要计数的累加变量  
                    while( $in < 10 ) { //指定内层循环，并且循环次数为10次  
                        echo "<td>".($out*10+$in)."</td>"; //执行一次，输出一个单元格  
                        $in++; //内层的计数变量累加  
                    }  
                    echo "</tr>"; //输出行关闭标记  
                    $out++; //外层的计数变量累加  
                }  
            ?>  
        </table>  
    </body>  
</html>
```

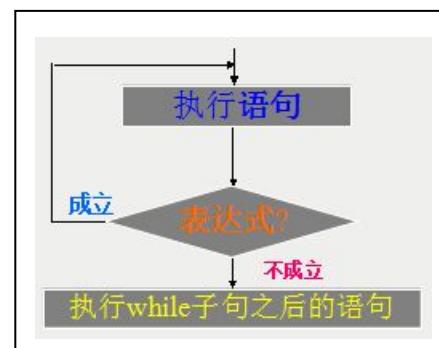


3、do...while 循环（特点是先执行循环体在做判断）

基本格式：

```
do{  
    语句或语句序列  
    ....  
}while(表达式);
```

程序会先执行 do 语句体中的语句（循环体），然后再检查表达式（循环控制语句）的值，如果符合条件式（值为真），就再进行 do 语句体中的语句，直到条件不符合为假时结束循环。



4、for语句（计数循环）

基本格式：

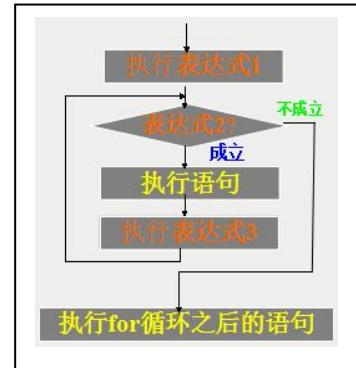
```
for(表达式 1;表达式 2;表达式 3)
{
    语句或语句序列;
}
```

for 循环语句中表达式 1 为循环初始条件；表达式 2 为循环控制条件；表达式 3 为控制变量递增；语句或语句序列为循环体。

表达式 1 是初始化循环条件的，执行次数 1 次

表达式 2 是循环的判断语句（决定是否进行循环体的执行），次数：n+1

表达式 3 是用于影响表达式 2 判断结果语句，如递增和递减操作，次数：n（循环次数）



<?php

```
//使用花括号{}将代码块括起来，通常代码块为一条时可以不加花括号
for( $i = 1; $i <= 10; $i++ ){
    echo "这是第<b> $i </b>次循环执行输出的结果<br>";
}

$i = 1; //将for语句中第一部分初始化提出来，放到for前面执行，但for语句中的分号要保留
for( ; $i <= 10; $i++ ){
    echo "这是第<b> $i </b>次循环执行输出的结果<br>";
}

$i = 1; //再将第三部分的增量提出来，放到for语句的执行体最后，但也要将分号保留
for( ; $i <= 10; ){
    echo "这是第<b> $i </b>次循环执行输出的结果<br>";
    $i++;
}

$i = 1;
for( ; ){
    if( $i > 10 )
        break;
    echo "这是第<b> $i </b>次循环执行输出的结果<br>";
    $i++;
}
```

```
<?php
$a = array(100,80,600,15,34,200,5);
print_r($a);
echo '<hr/>';
echo count($a); //数值下标从0开始索引
echo "<hr/><br/>";

//排序的循环次数
for($i=0;$i<count($a)-1;$i++){
    //执行排序，负责一次找出一个
    for($j=0;$j<count($a)-$i-1;$j++){
        //确定顺序（大：从小到大）
        if($a[$j]>$a[$j+1]){
            //执行交换位置
            $tmp = $a[$j];
            $a[$j] = $a[$j+1];
            $a[$j+1] = $tmp;
        }
    }
    print_r($a);
    echo "<br/><br/>";
}
```

5、特殊流程控制语句

1. break

我们之前在 switch 条件判断中已经使用过 break 关键字，它会使得程序流程离开 switch 本体中的语句，如果 break 使用在 for、while 或 do-while 循环结构中时，将会使得程序离开该层循环。

2. continue

continue 的作用与 break 有点类似，continue 若使用在 for、while 或 do-while 循环结构中，当程序执行至 continue 时，之后的语句将直接被略过，而直接执行下一次的循环动作。

3. exit

当前的脚本中只要执行到 exit 语句，而不管它在哪个结构中都会直接退出当前脚本。exit 是一个函数，当前使用过的 die() 函数就是 exit() 的别名。可以带参数输出一条消息，并退出当前脚本。

程序思路：（循环分清变化的值）

```
<?php
//循环结构
for($i=1;$i<=9;$i++){
    echo $i." ";
}
echo "<hr/>";
//输出9行，1到9的值
for($j=1;$j<=9;$j++){
    //输出1到9的值
    for($i=1;$i<=9;$i++){
        |   echo $i." ";
    }
    echo "<br/>";
}

echo "<hr/>";
//输出9行，1到9的值(三角形)
for($j=1;$j<=9;$j++){
    //输出1到9的值
    for($i=1;$i<=$j;$i++){
        |   echo $i." ";
    }
    echo "<br/>";
}

echo "<hr/>";
//输出9行，1到9的值(99乘法表)
for($j=1;$j<=9;$j++){
    //输出1到9的值
    for($i=1;$i<=$j;$i++){
        |   echo $i."X".$j."=".($i*$j)." ";
    }
    echo "<br/>";
}

echo "<hr/>";
//输出9行，1到9的值(表格式的99乘法表)
echo "<table width='500' border='1'>";
for($j=1;$j<=9;$j++){
    echo "<tr>";
    //输出1到9的值
    for($i=1;$i<=$j;$i++){
        |   echo "<td>{$i}X{$j}=".($i*$j)."</td>";
    }
    echo "</tr>";
}
echo "</table>";

echo "<hr/>";
//输出9行，1到9的值(表格式的99乘法表)
echo "<table width='500' border='1'>";
for($j=9;$j>=1;$j--){
    echo "<tr>";
    //输出空单元格
    for($i=0;$i<9-$j;$i++){
        |   echo "<td>&nbsp;</td>";
    }
    //输出1到9的值
    for($i=$j;$i>=1;$i--){
        |   echo "<td>{$i}X{$j}=".($i*$j)."</td>";
    }
    echo "</tr>";
}
echo "</table>";
```

1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9

1X1=1
1X2=2 2X2=4
1X3=3 2X3=6 3X3=9
1X4=4 2X4=8 3X4=12 4X4=16
1X5=5 2X5=10 3X5=15 4X5=20 5X5=25
1X6=6 2X6=12 3X6=18 4X6=24 5X6=30 6X6=36
1X7=7 2X7=14 3X7=21 4X7=28 5X7=35 6X7=42 7X7=49
1X8=8 2X8=16 3X8=24 4X8=32 5X8=40 6X8=48 7X8=56 8X8=64
1X9=9 2X9=18 3X9=27 4X9=36 5X9=45 6X9=54 7X9=63 8X9=72 9X9=81

| | | | | | | | | | |
|------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|--|
| $1 \times 1 = 1$ | | | | | | | | | |
| $1 \times 2 = 2$ | $2 \times 2 = 4$ | | | | | | | | |
| $1 \times 3 = 3$ | $2 \times 3 = 6$ | $3 \times 3 = 9$ | | | | | | | |
| $1 \times 4 = 4$ | $2 \times 4 = 8$ | $3 \times 4 = 12$ | $4 \times 4 = 16$ | | | | | | |
| $1 \times 5 = 5$ | $2 \times 5 = 10$ | $3 \times 5 = 15$ | $4 \times 5 = 20$ | $5 \times 5 = 25$ | | | | | |
| $1 \times 6 = 6$ | $2 \times 6 = 12$ | $3 \times 6 = 18$ | $4 \times 6 = 24$ | $5 \times 6 = 30$ | $6 \times 6 = 36$ | | | | |
| $1 \times 7 = 7$ | $2 \times 7 = 14$ | $3 \times 7 = 21$ | $4 \times 7 = 28$ | $5 \times 7 = 35$ | $6 \times 7 = 42$ | $7 \times 7 = 49$ | | | |
| $1 \times 8 = 8$ | $2 \times 8 = 16$ | $3 \times 8 = 24$ | $4 \times 8 = 32$ | $5 \times 8 = 40$ | $6 \times 8 = 48$ | $7 \times 8 = 56$ | $8 \times 8 = 64$ | | |
| $1 \times 9 = 9$ | $2 \times 9 = 18$ | $3 \times 9 = 27$ | $4 \times 9 = 36$ | $5 \times 9 = 45$ | $6 \times 9 = 54$ | $7 \times 9 = 63$ | $8 \times 9 = 72$ | $9 \times 9 = 81$ | |

注意：单双引号中不能解析运算符！

for 循环的特殊写法

```
//for循环的特殊写法  
//(理解for循环的执行顺序)  
  
//循环输出1到10的值  
for($i=1;$i<=10;$i++){  
    echo $i." ";  
}
```

```
//循环输出1到10的值  
$i=1;  
for(; $i<=10; $i++){  
    echo $i." ";  
}
```

```
//循环输出1到10的值  
$i=1;  
for(; $i<=10;){  
    echo $i." ";  
    $i++;  
}
```

```
//循环输出1到10的值  
$i=1;  
for(;;){  
    if($i>10){  
        break; //退出循环  
    }  
    echo $i." ";  
    $i++;  
}
```

| |
|----------------------|
| 1 2 3 4 5 6 7 8 9 10 |
| 1 2 3 4 5 6 7 8 9 10 |
| 1 2 3 4 5 6 7 8 9 10 |
| 1 2 3 4 5 6 7 8 9 10 |

Do...while 循环实例（注意步骤）

```
//do....while循环的使用(特点先执行后判断)  
  
//输出1到10的值  
$i=20;  
do{  
    echo $i." ";  
    $i++;  
}while($i<=10);
```

```
//输出1到100中被3整除值。  
$i=1;  
do{  
    if($i%3==0){  
        echo $i." ";  
    }  
    $i++;  
}while($i<=50);
```

```
//输出9行，1到9值  
$j=1;  
do{  
    $i=1;  
    do{  
        echo $i." ";  
        $i++;  
    }while($i<=9);  
    echo "<br/>";  
    $j++;  
}while($j<=9);
```

```
//输出9行，1到9值  
$j=1;  
do{  
    $i=1;  
    do{  
        echo $i." ";  
        $i++;  
    }while($i<=$j);  
    echo "<br/>";  
    $j++;  
}while($j<=9);
```

```
//输出9行，1到9值  
$j=1;  
do{  
    $i=1;  
    do{  
        echo "{$i}*{$j}=".($i*$j)." ";  
        $i++;  
    }while($i<=$j);  
    echo "<br/>";  
    $j++;  
}while($j<=9);
```

```
2 3 4 5 6 7 8 9 10
```

```
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48
```

```
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8 9
```

```
1  
1 2  
1 2 3  
1 2 3 4  
1 2 3 4 5  
1 2 3 4 5 6  
1 2 3 4 5 6 7  
1 2 3 4 5 6 7 8  
1 2 3 4 5 6 7 8 9
```

```
1*1=1  
1*2=2 2*2=4  
1*3=3 2*3=6 3*3=9  
1*4=4 2*4=8 3*4=12 4*4=16  
1*5=5 2*5=10 3*5=15 4*5=20 5*5=25  
1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36  
1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49  
1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64  
1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

写程序要有思路：先循环遍历需要数据再进行判断处理需要的数据。

```
//输出1到100中被3整除值。  
$i=1;  
do{  
    if($i%3==0){  
        echo $i." ";  
    }  
    $i++;  
}while($i<=100);
```

```
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48
```

While 循环实例（理解行数与星星个数的规律）

» do-while 后面必须加上分号作为结束

» do-while 与 while 的最大区别是先执行循环体然后再判断循环的控制条件

```

<?php
//while循环的使用

//循环输出1到10的值
$i=1;
for($i<=10){
    echo $i." ";
    $i++;
}
echo "<hr/>";

//循环输出1到10的值
$i=1;
while($i<=10){
    echo $i." ";
    $i++;
}
echo "<hr/>";

//一次输出一个星，输出10次
$i=0;
while($i<10){
    echo "* ";
    $i++;
}
echo "<hr/>";

//一次输出一个星，输出10次,共10行
$j=0;
while($j<10){

    $i=0;
    while($i<10){
        echo "* ";
        $i++;
    }
    echo "<br/>";
    $j++;
}

echo "<hr/>";
//一次输出一个星，输出10次,共10行 (三角型的)
$j=0;
while($j<10){

    $i=0;
    while($i<$j){
        echo "* ";
        $i++;
    }
    echo "<br/>";
    $j++;
}

echo "<hr/>";
//一次输出一个星，输出10次,共10行 (三角型的)
$j=0;
while($j<10){

    $z=0;
    while($z<10-$j){
        echo " ";
        $z++;
    }

    $i=0;
    while($i<$j*2+1){
        echo "*";
        $i++;
    }
    echo "<br/>";
    $j++;
}

```

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

* * * * * * * * *

* * * * * * * * *

* * * * * * * * *

* * * * * * * * *

* * * * * * * * *

* * * * * * * * *

* * * * * * * * *

* * * * * * * * *

*

* *

* * *

* * * *

* * * * *

* * * * * *

* * * * * * *

特殊流程控制语句实例：

```
<?php
//特殊的流程控制:
for($i=1;$i<=10;$i++){
    if($i==6){
        break; //退出当前循环
    }
    echo $i." ";
}
echo "<hr/>";

for($i=1;$i<=10;$i++){
    if($i==6){
        continue; //退出本次循环, 继续下一循环
    }
    echo $i." ";
}

echo "<hr/>";
//循环输出1到9的值, 然后是9行
for($j=1;$j<=9;$j++){
    for($i=1;$i<=9;$i++){
        if($i==6){
            //break; //退出当前循环
            break 2; //退出2层循环
        }
        echo $i." ";
    }
    echo "<br/>";
}

echo "<hr/>";
//循环输出1到9的值, 然后是9行
for($j=1;$j<=9;$j++){
    for($i=1;$i<=9;$i++){
        if($i==6){
            //continue;
            continue 2; //跳出2层本次循环, 继续下一次
        }
        echo $i." ";
    }
    echo "<br/>";
}
```

输出结果：

```
1 2 3 4 5
-----
1 2 3 4 5 7 8 9 10
-----
1 2 3 4 5
-----
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

6、可变变量和引用赋值

1. 可变变量：就是变量的名字还是一个变量。

```
$a="name";
$name="zhangsan";
echo $$a; //变量的名字还是一个变量时就是可变变量， $a 等价于 name,即 $$a 等价于$name
```

2. 引用赋值：(相当于起别名) 指针赋值，使用"&"符号 (Linux 软链接)

```
$m=10; //定义一个变量 m 值为 10;
$n=&$m; //将 m 的值所在的内存地址给了变量 n (引用方式赋值，也叫起别名)
$n=20; //将变量 n 的值改为 20
echo $m; //输出变量 m 的值： 20
```

//值的复制 (值传递) 普通赋值

```
$a=10;
$b=$a; //将 a 的值复制一份赋给变量 b
$b=20;
echo $a; //10
```

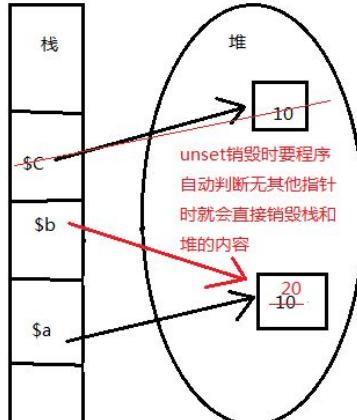
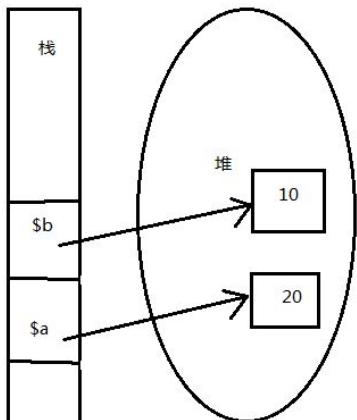
//引用赋值 (引用传递)

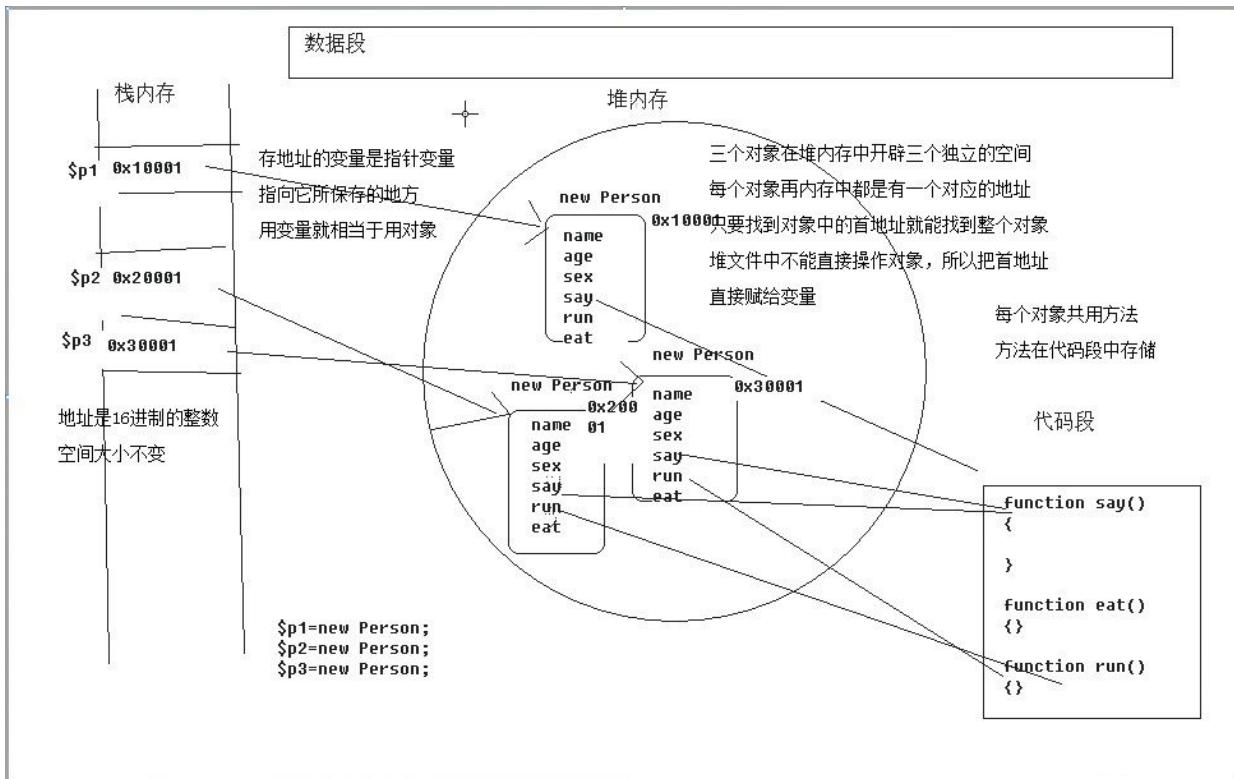
```
$a=10;
$b=&$a; //为变量 a 起个别名叫 b
$b=20;
echo $a; //20
```

```
unset($b); //销毁变量 b;
```

```
echo $a; //20
```

在栈里面存放变量名有赋值时会在堆里面申请一个空间进行存放值





四、函数

1. 函数的定义
2. 自定义函数
3. 函数的工作原理和结构化编程
4. PHP 变量的作用域
5. 声明及应用各种形式的 PHP 函数

1、函数的定义

函数是什么：

函数（function）是一段完成指定任务的已命名代码块。函数可以遵照给它的一组值或参数完成特定的任务，并且可能返回一个值。在 PHP 中有两种函数：自定义函数与系统函数。

函数的优越性：

- 提高程序设计的复杂性
- 提高软件的可靠性
- 提高软件的开发效率
- 提高软件的可维护性
- 提高程序的重用性

2、自定义函数

自定义函数语法格式：

```
function 函数名称 ([参数 1 [,参数 2 [,⋯]]) {  
    程序内容叙述(也叫函数体);  
    [return 返回值;] //如需函数有返回值时使用  
}
```

函数的使用：

函数名称 ([参数 1 [,参数 2 [,⋯⋯⋯]]) ;

自定义函数的名称：

它是函数在程序代码中的识别名称，函数名可以是以字母或下划线开头后跟零个或多个字母、下划线和数字的任何字符串。函数名不区分大小写。命名函数时不可使用已声明的函数，或 PHP 内建的函数名称。

参数：

所谓的参数就是用来把数值由函数外部传入函数体中，并用来加以运算处理。参数之间用“，”号隔开。当函数不需要任何数值传入时，可以省略参数。（初始化参数）

返回值：（如有需要函数中处理的值就直接自定义返回值，将值返回到函数的调用处。）

当调用函数时需要它返回一些数值，那么就要函数体中用 return 语句实现。格式如下：

```
return 返回值; //返回值也可以是一个表达式  
exit(); //无返回值 void
```

注意：函数定义后不调用不会执行，程序运行会将检测当前脚本语法的正确性后遇到函数就登记在内存中，当使用时就只要直接调用执行脚本即可执行函数的功能，说明函数的调用不分先后顺序同理于实例化类。

注意：遇到函数调用的处就进入函数的代码块中执行代码，执行函数中的代码结束后再出来到调用的函数处，接着往下执行当前脚本的代码。

》如返回值需要在另一个函数中使用或者需要对返回值进行下一步的处理。

》在函数中一但遇到 return，就会终止当前函数的执行，返回到调用处

```
<?php  
//定义函数  
function mysum($x,$y){  
    //echo $x+$y;//直接输出结果  
    return $x+$y;//直接返回结果  
}  
//调用函数  
echo mysum(10,20);  
  
/*直接在函数中输出结果的可定制弱  
将函数的处理的值进行返回时可定制  
强，对返回值可以进一步处理*/
```

如何定义函数：

先实现代码功能再使用关键字 function 定义函数名（参数）{ 实现的功能代码} 最后将可变的变量提取为参数即可

3、判断函数是否存在

在 PHP 中，函数可以在被调用之前定义，也可以在被调用之后定义。

function_exists() 判断函数是否存在。

```

<?php
//判断一个函数是否存在 function_exists();

//定义函数
function fun(){
    echo "ok";
}

//先判断函数是否存在
if(function_exists("fun")){
    fun(); //调用函数
}else{
    echo "函数fun不存在!";
}

```

4、PHP 变量的范围（由于函数的出现导致有变量的作用范围）

变量的范围

当主程序调用函数时，PHP 会暂时停止目前主要程序流程的运行，并传递必要的运算参数给目标函数使用，以执行函数的程序码片段。

在函数执行结束后，函数会回传执行结果所得的数值，并将执行流程转回原本主程序中断的地方，继续执行运作。

变量的能见度

所谓变量的能见度，意指[变量在程序中的可作用范围](#)。当一个变量执行赋值动作后，会随着声明局部的差异，而有不同的作用范围。大致上来说变量会依据声明的局部分为下列两种：[局部变量](#)和[全局变量](#)

1.局部变量（内部变量）简单理解就是在函数内部定义的变量

在函数之中声明的变量就是局部变量，并且该变量只有在函数范围之中才能加以使用。如果其它程序局部需要调用使用该变量值时，必须透过「return」指令，来将其传回至主程序区块以作后续处理。

```

<?php
$a=0;
function print_A(){
    $a = 3; //定义局部变量
    echo "在函数中显示局部变量 a 值: $a <p>";
    return $a;
}
$b = print_A();
echo "在函数外显示局部变量 b 值: $b <br>";
echo $a;
?>

```

2.全局变量

在函数范围之外所声明的变量就是全局变量。由于函数可以视为单独的程序片段，所以局部变量会复盖全局变量的能见度，因此在函数中并无法直接调用使用全局变量。

函数中若要使用全局变量时，必须要利用 **global** 关键字定义目标变量，以告诉函数主体此变量为全局。
理解：在函数内使用函数外的变量时需要在函数内用 **global** 关键字修饰，要在函数内使用的变量是函数外的变量的。

```
<?php
$A= "Hello !!"; //定义全局变量
function print_A() { //定义函数 print_A()
    global $A; //利用 global 关键字修饰变量 A 以说明函数内使用的该变量是外面声明的
    echo $A;
}
print_A();
?>
```

也可以使用预定义的**全局变量数组\$GLOBALS**。这是一个特殊变量在程序运行时自动创建。

格式：echo \$GLOBALS[“A”];

```
<?php
//变量的作用域
//由于函数的出现，导致变量的作用域范围发生改变：
//分：局部变量、全局变量、静态变量
```

//1. 局部变量：就是在函数内定义的变量

```
function fun(){
    $a=10; //在函数内定义的变量是局部变量
    echo "函数内a的值:{$a}<br/>";
}

fun(); //调用
//fun(); //调用
echo "函数外a的值:{$a}<br/>"; //函数内的变量，在函数外是无法使用
echo "<br/>";
```

//2. 全局变量：就是在函数外定义的变量。特点在函数内可见（但是需要使用**global**关键字声明一下）

\$name="zhangsan"; //函数外定义的变量称全局变量

```
function demo(){
    global $name; //使用global关键字声明一下变量，（就是说我要使用外边的name变量）
    echo "函数内name的值:{$name}<br/>";
    $name="lisi"; //修改
}

demo();
echo "函数外name的值:{$name}<br/>";
```

函数内a的值:10

| (!) Notice: Undefined variable: a in D:\wamp\www\1\test.php on line 16 | | | | |
|--|--------|--------|----------|---------------|
| Call Stack | | | | |
| # | Time | Memory | Function | Location |
| 1 | 0.0000 | 143584 | {main}() | ..\test.php:0 |

函数外a的值：

函数内name的值:zhangsan

函数外name的值:lisi

3. 静态变量(在函数内使用 static 关键字定义的变量)

PHP 支持声明函数变量为静态的(static)。一个静态变量在所有对该函数的调用之间共享，并且仅在脚本的执行期间函数第一次被调用时被初始化。要声明函数变量为静态的用关键字 static。通常，静态变量的第一次使用时赋予一个初始值。

```
function test() {  
    // 使用 static 修饰变量时仅执行一次该行代码，再次调用时不会执行(赋值)但变量依然有效  
    static $a = 0; 声明变量$a为静态变量同时赋初值才能有效  
    echo $a;  
    $a++;  
}
```

特点：初始化只做一次，而且多次调用函数依然有效。

总结：由于函数的出现，导致变量的能见度（生命周期）不同。（有内外之分）

共划分为：全局变量、局部变量、静态变量：

1. *全局变量：就是在函数外定义变量

作用域：在函数外和函数内都可以看到的，并可以使用的变量

注意：在函数内使用函数外的全局变量要使用 global 关键字先声明一下。

```
<?php  
    // 变量的作用域：全局变量  
    $a=10;// 定义一个变量a 值为10  
    $b=20;  
    // 定义一个函数  
    function fun(){  
        global $a; // 尝试使用外部的全局变量a  
        echo $a; // 输出变量a 的值10  
        echo $b; // 输出变量b 的值，结果没有（因为没有使用global）  
        $a=100;  
    }  
    fun(); // 调用函数fun  
    echo $a; // 100  
?>
```

2. *局部变量：在函数内定义的变量

```
<?php  
    // 变量的作用域：局部变量  
    // 定义一个函数  
    function fun(){  
        $a=100; // 函数内定义变量  
        echo $a; // 输出变量a 的值100  
    }  
    fun(); // 调用函数fun  
  
    echo $a; // 没有输出，在函数内定义的变量，在函数外是无法使用的  
?>
```

3. 静态变量：在函数内使用 static 关键字定义的变量

```
//变量作用域：静态变量的使用
function fun(){
    //在当前脚本中连续多次调用本函数，i 依然有效，里面的静态声明（初始化）只会调用一次。
    static $i=1;
    $i++;
    echo $i."<br/>";
}
fun(); //1
fun(); //2
fun(); //3
fun(); //4
```

查看函数三要素：

无论是使用系统函数还是自定义的函数，我们都要可通过如下函数的三要素来了解一个函数：

- 》 函数的功能描述
- 》 函数的参数
- 》 函数的返回值

5、声明及应用各种形式的 PHP 函数

- 1 常规参数的函数
- 2 伪类型参数的函数
- 3 引用参数的函数
- 4 默认参数的函数
- 5 可变个数参数的函数
- 6 回调函数

1. 常规参数的函数

常规参数的函数格式说明：

```
string example(string name,int age,double height)
```

所谓的常规参数的函数，就是实参和形参应该个数相等、类型一致。就像 C 或 Java 等强类型语言。

上面函数有三个参数，调用时传递的参数个数和顺序必须一致。

```
string chr(int $ascii) //返回指定的字符
float ceil(float $value) //进一法取整
array array_combine(array $keys,array $values)//合并一个数组
string implode(string $glue,array $pieces)
```

2. 伪类型参数的函数

伪类型参数的函数格式说明：

```
mixed funName(mixed $a, number $b, callback $c)
```

PHP 是弱类型语言，不仅在声明变量时不需要指定类型，当然在声明函数时也不需要指定类型，所以在 PHP 中函数的每个参数，都可以为其传递任意类型的值。

三种伪类型： mixed、number 和 callback。

```
bool empty(mixed $var) //检查一个变量是否为空
```

```
bool usort(array &$array,callback $cmp_function)//使用用户自定义的比较函数对数组中的值进行排序
```

```
number abs(mixed $number) //绝对值
```

3. 引用参数的函数

引用参数的函数格式说明：

```
void funName(array &args)
```

相对于按值传递模式，并不会将父程序中的指定数值或目标变量传递给函数，而是把该数值或变量的内存储存区块相对地址导入函数之中。因此当该数值在函数中有任何变动时，会连带对父程序造成影响。

注意：如果在函数的形参中使用“&”修饰的参数，在调用该函数时必须传入一个变量给这个参数，而是不能传递一个值。

实例：

```
<?php
//函数调用时的参数传递：值传递、引用传递

//函数定义（其中第一参数属于值传递，第二个参数属于引用传递（b是y别名））
function demo($a,&$b){
    echo "在函数内： a={$a}和b={$b}<br/>";
    $a=100;
    $b=200;
}

//调用函数
$x=10;
$y=20;
echo "调用函数前： x={$x}和y={$y}<br/>";

demo($x,$y);

echo "调用函数后： x={$x}和y={$y}<br/>";
```

调用函数前： x=10和y=20

在函数内： a=10和b=20

调用函数后： x=10和y=200

4. 默认参数的函数

默认参数(“[]”中的)的函数格式说明：(PHP 特点)

```
mixed funName(string name [,string value [,int num]])
```

在 php 中，支持函数的默认方式调用。如果在调用函数时没有指定参数的值，在函数中会使用参数的默认值。

注意：默认参数必须列在所有没有默认值参数的后面。（从后面开始指定默认值）

```
function add($a,$b=0,$c=0){
    return $a+$b+$c;
}
echo add(10,20,30)."<br/>";
echo add(10,20)."<br/>";
echo add(10)."<br/>";
```

5. 可变个数参数的函数

可变个数参数的函数格式说明：

```
mixed funName(string arg1 [,string ...] )
```

通常用户定义函数时，设置的参数数量是有限的。如果希望函数可以接受任意数量的参数，需使用以下函数：

```
func_get_args() //返回一个数组，包含所有参数
```

```
func_num_args() //返回参数总数
```

```
func_get_arg() //接收一个数字参数，返回指定参数
```

```
<?php
//可变长度参数传递
function add(){
    //echo func_num_args(); //获取参数的个数
    //echo func_get_arg(4); //获取指定位置上（从0开始）的参数值

    //var_dump(func_get_args()); //获取所有参数值，并解析输出

    $sum=0;
    $m=func_num_args(); //获取参数的个数
    //循环获取所有参数值
    for($i=0;$i<$m;$i++){
        $sum+=func_get_arg($i); //获取每个位置上的参数值，并累加到sum中
    }
    return $sum;//返回累加值;
}
```

总结：

函数中参数的传递：值传递、引用方式传递、参数默认值、可变长度参数列表。

1. 值传递：

在定义函数时的参数是普通参数；

2. 引用方式传递：

使用&符来实现引用传递，就是将参数直接给给函数使用。（起别名）

假如函数内容发生改变，会使当前参数也发生改变。

注意：调用引用参数时，必须使用变量，不可以直接用值。

如：fun(\$x); 不可以使用fun(10);

在定义函数时的参数是使用&标识的参数；

```
function myadd($x,$y,&$z){
    $z=$x+$y;
}
$a=10;
$b=20;
$sum=0;
myadd($a,$b,$sum);//调用函数myadd，将前两个参数的和赋给第三个参数
echo $sum; //30
```

3. 参数默认值：

我们可以在定义函数时，将部分参数提前附上初始值，

当被调用时没有给这参数传值，就会采用默认值，以保障函数正常使用。

要求默认值从参数的右边(后面)开始。

4. 可变长度参数列表。

此函数的参数个数任意个。

```
func_get_args(); //以数组的方式返回所有的参数信息  
func_get_arg($index) //获取指定参数位置 (index: 从 0 开始的整数) 的信息  
func_num_args() //获取参数的个数。
```

```
//可变长度参数列表  
function dosum(){  
    //echo "参数的个数: ".func_num_args()."<br/>";  
    //echo "某个参数的值: ".func_get_arg(4)."<br/>";  
    $sum=0;  
    for($i=0;$i<func_num_args();$i++){  
        echo func_get_arg($i)." ";  
        $sum+=func_get_arg($i);  
    }  
    echo "<br/>";  
    echo "共计: ".$sum;  
}
```

dosum(10,20,50,60,300,100,20);

5. 变量函数：就是调用函数使用的不是直接的函数名，而是一个变量。

如：

要调用 add 函数。应该是：add(10,20);
但是： \$name="add"; \$name(10,20); 等价于 add(10,20);

定义一个函数库文件： functions.php

6. 在另外文件： 导入： include() require()

require() 和 include() 几乎完全一样，除了处理失败的方式不同之外。

include() 产生一个 Warning (适合导入不重要的文件)

require() 则导致一个 Fatal Error。 (导入重要文件)

6. 变量函数： 可变函数---其实函数的名字还是一个变量

回调函数

回调函数格式说明：

mixed funName(callback arg)

所谓回调函数，就是指调用函数时并不是传递一个标准的变量作为参数，而是将另一函数作为参数传递到调用的函数中。

1. 变量函数

2. 使用变量函数声明和应用的回调函数

3. 借助 call_user_func_array() 函数自定义回调函数

4. 类静态函数和对象的方法回调

实例：

```
<?php
//变量函数：可变函数---其实就是函数的名字还是一个变量
function aa(){
    echo "aaaaaaaaaa<br/>";
}

function bb(){
    echo "bbbbbbbbbb<br/>";
}

//$name="bb";

//将$name换成对应的值，在作为函数使用

//回调函数
function fun($m){
    if(function_exists($m)){
        $m(); //可变函数（变量函数）
    }else{
        die("{$_m}函数不存在！");
    }
}

fun("aa"); //传入一个函数名aa进去
fun("bb"); //传入一个函数名bb进去
fun("cc"); //传入一个函数名cc进去（由于函数cc不存在，故报错）
```

使用变量函数声明和应用的回调函数

```
<?php
// 声明回调函数 filter, 在 0-100 的整数中通过自定义条件过滤不要的数字
function filter( $fun ) {
    for($i=0; $i <= 100; $i++) {
        //将参数变量$fun 加上一个圆括号$fun(), 则为调用和变量$fun 值同名的函数
        if( $fun($i) )
            continue;
        echo $i."<br>";
    }
}

//声明一个函数 one, 如果参数是 3 的倍数就返回 true, 否则返回 false
function one($num) {
    return $num%3 == 0;
}

// 声明一个函数 two, 如果参数是一个回文数(翻转后还等于自己的数)返回 true, 否则返回 false
function two($num) {
```

```

        return $num == strrev($num);
    }
filter("one"); //打印出 100 以内非 3 的倍数, 参数"one"是函数 one()的名称字符串, 是一个回调
echo '-----<br>';
filter('two'); //打印出 100 以内非回文数, 参数"two"是函数 two()的名称字符串, 是一个回调

```

7. 借助 call_user_func_array() 函数自定义回调函数

```

<?php
function add($a,$b){
    echo "{$a}+{$b}=".($a+$b);
}

$name="add"; //即将要执行的函数名
$p=array(100,200); //调用函数时所需的参数;

//第一种
//$name($p[0],$p[1]); //通过变量函数调用方式来执行

//第二种: 通过call_user_func_array()
call_user_func_array($name,$p);

```

强化实例:

```

//可变长度参数的求和函数
function add2(){
    $sum=0;
    $m=func_num_args(); //获取参数的个数
    //循环获取所有参数值
    for($i=0;$i<$m;$i++){
        $sum+=func_get_arg($i); //获取每个位置上的参数值, 并累加到sum中
    }
    return $sum;//返回累加值;
}

//1. 直接执行函数
//echo add2(10,20,30,40,50,60);

//2. 变量函数(函数名可变)
//$name="add2";
//$p=array(10,20,30,40);
//echo $name($p[0],$p[1],$p[2],$p[3]);

//3. 通过call_user_func_array()(函数名和参数可变)
$name="add2";
$p=array(10,20,30,40,50,60);
echo call_user_func_array($name,$p);

```

8. 自定义函数库:

1. 定义一些常用函数存放在一个 php 文件中, 如: functions.php
2. 在我们要使用函数的 php 文件中使用:

include 或 require 导入函数库文件后即可使用。

如: include("functions.php");

或 require("functions.php");

9. 使用系统函数:

如何了解一个系统函数并学会使用他

1.功能: 要了解本函数的实现功能(干什么?)

2.参数: 调用这些函数都需要哪些参数(可选和必选)以及类型。

3.返回值: 本函数都有什么返回结果。

//1.file_get_contents() --将整个文件读入一个字符串

//2. file_put_contents -- 将一个字符串写入文件

10、递归函数

递归函数--就是在函数体的代码中出现了调用自己本身函数的语句。注意递归的结束。

关键点: 函数在哪调用, 就返回到哪。

```
<?php
//递归函数
function fun($m){
    echo $m." ";
    if($m>1){ //递时指定条件
        fun($m-1); //递
    }
    echo $m." "; //归
}

fun(3);
/*
3
if(3>1){
    2
    if(2>1){
        1
        1
    }
    2
}
3
*/
```

输出结果: 3 2 1 1 2 3

```

//递归函数
function fun($m){
    echo $m." ";
    if($m>1){ //递时指定条件
        fun($m-1); //递
    }
    echo $m." "; //归
}

fun(3);

```

拆解：（一定要明白函数在那调用，就返回到那，函数执行结束标志是花括号）

| | | |
|---|---|---|
| <pre> function fun3(\$m){ echo \$m." "; if(\$m>1){ fun2(\$m-1); } echo \$m." "; } </pre> | <pre> function fun2(\$m){ echo \$m." "; if(\$m>1){ fun1(\$m-1); } echo \$m." "; } </pre> | <pre> function fun1(\$m){ echo \$m." "; if(\$m>1){ fun1(\$m-1); } echo \$m." "; } </pre> |
|---|---|---|

注意特点：不停做相同的事情，执行相同的代码，可以建议使用递归函数。

| | | |
|--|---|--|
| <pre> <?php function fun(\$m){ //递归结束条件 if(\$m<1){ return; } echo \$m." "; fun(\$m-1); //调用自身函数 echo \$m." "; } </pre> | <p>//此处调用函数fun3,函数处理结果返回到调用处</p> <pre> fun3(3); </pre> | |
| <pre> function fun2(\$m){ echo \$m." "; fun1(\$m-1); echo \$m." "; } </pre> | <pre> function fun1(\$m){ echo \$m." "; fun0(\$m-1); echo \$m." "; } </pre> | <pre> function fun0(\$m){ return; } </pre> |

11、使用自定义函数库

1.代码重用

通过重复使用已有的代码，提高开发效率，降低成本

2.include()和 require() 函数。

require()将一个文件在预处理期间被导入，像把该文件粘贴到使用函数的地方。

include()与 require()几乎等价，区别在于在脚本执行时包含，当处理失败时，include()产生一个警告（程序会继续执行，一般导入不太重要如静态页面，使用在页面拆分方面）而 require()则导致一个致命错误（程序终止，一般导入重要的文件如数据库配置文件，函数库文件）。

3.include_once()和 require_once()函数

两个函数在脚本执行期间包括并运行指定文件。与 include()语句及 require()类似，唯一区别是如果该文件中的代码已经被包括了，则不会再次包括，只会包括一次。这两个函数应该用于在脚本执行期间同一个文件有可能被包括超过一次的情况下，你想确保它只被包括一次以避免函数重定义，变量重新赋值等问题。

五、PHP 中的数组与数据结构

1. 数组的概述
2. 数组的定义
3. 数组的遍历

2013 年度中国国优秀开源项目评选--了解

1. 数组的概述

PHP 中的数组实际上是一个有序图。图是一种把 values 映射到 keys 的类型。此类型在很多方面做了优化，因此可以把它当成真正的数组来使用，或列表（矢量），散列表（是图的一种实现），字典，集合，栈，队列以及更多可能性。因为可以用另一个 PHP 数组作为值，也可以很容易地模拟树。

重要知识：

- 》 所谓的数组下标可以视为资料内容在此数组中的识别名称，通常被称为数组下标。
- 》 当索引值为数值时，也代表此资料内容在数组中的储存位置。
- 》 数组中有几个索引值就被称为几维数组。（数组包含数组）

数组分类：

在 PHP 中有两种数组：索引数组和关联数组。

索引（indexed）数组的索引值是整数，以 0 开始。当通过位置来标识东西时用索引数组。

关联（associative）数组以字符串做为索引值，关联数组更像操作表。索引值为列名，用于访问列的数据。（关联数组可读性强）

2. 数组的定义

- 1> 直接赋值的方式声明数组
- 2> 使用 array()语言结构新建数组
- 3> 多维数组的声明

数组常用的赋值方式：

由于 PHP 是属于弱类型数据，因此源代码中的数组并不需要经过特别的声明操作，直接将一组数值指定给某一数组元素即可。一般情况下数组的赋值有两种方式：

直接赋值方式 如：

\$a[0]= ‘spam@126.com’ ;

为什么 \$foo[bar] 错了？

\$a[1]= ‘abuse@sohu.com’ ;

应该始终在用字符串表示的数组索引上加上引号。例如用 \$foo['bar'] 而不是 \$foo[bar]。但是为什么呢？可能在老的脚本中见过如下语法：

使用 array 函数 如：

\$a=array(“spam@126.com” , “abuse@sohu.com”);

\$foo [bar] = ‘enemy’;

echo \$foo [bar];

// etc

?>

这样是错的，但可以正常运行。那么为什么错了呢？原因是此代码中有一个未定义的常量（bar）而不是字符串（‘bar’ - 注意引号），而 PHP 可能会解析为常量，它能运行，是因为 PHP 自动将裸字符串（没有引号的字符串且不对应于任何已知符号）转换成一个其值为该裸字符串的正常字符串。例如，如果没有常量定义为 bar，PHP 将把它替代为 ‘bar’ 并使用之。

声明数组的方式：（2 种）---其他语言必须声明时必须指定长度

直接赋值的方式声明数组

格式：

\$数组变量名[索引值]=资料内容

其中索引值（下标）可以是一个字符串或一个整数。等价于整数（不以 0 开头）的字符串值被当作整数对待。因此，数组\$array[3]与\$array[‘3’]是引用相同的元素。但是\$array[‘03’]引用的另外不同的元素。

一维数组

数组中索引值(下标)只有一个的数组称为一维数组。在数组中这是最简单的，也是最常用的了。

使用 array()语言结构新建数组

格式：

```
array( [key]=> value , ... )
// key 可以是 integer 或者 string
// value 可以是任何值
```

» key 可以是 integer 或者 string。如果键名是一个 integer 的标准表达方法，则被解释为整数（例如“8”将被解释为 8，而“08”将被解释为“08”）。key 中的浮点数被取整为 integer。PHP 中没有不同的数字下标和关联下标数组，数组的类型只有一种，它可以同时包含整型和字符串型的下标。
» 如果对给出的值没有指定键名，则**取当前最大的整数索引值，而新的键名将是该值加一**。如果**指定的键名已经有了值，则该值会被覆盖**。

3. 多维数组的声明

二维数组的声明

多维数组的声明方式及规则，与一维数组相同，例如：下面二维数组的声明片段：

```
<?php
$A[0][] = 0;
$A[0][] = 1;
$A[“String”][0] = “Zero” ;
$A[“String”][1] = “One” ;
?>
```

这时数组中的资料内容如下：

一维索引值 0		一维索引值 String	
二维索引值 0	二维索引值 1	二维索引值 0	二维索引值 1
0	1	Zero	One

如果以 array 语法声明，则如下程序片段：

```
<?php
$A = array(0=>array{0, 1}, “String” =>array( “Zero” , “One” ));
?>
```

```
<?php
$arr[0][]=10;
$arr[0][]=20;
$arr[1][]=30;
$arr[1][]=40;
echo "<pre>";
print_r($arr);
echo "</pre>";
```

```
Array
(
    [0] => Array
        (
            [0] => 10
            [1] => 20
        )
    [1] => Array
        (
            [0] => 30
            [1] => 40
        )
)
```

4. 数组的遍历（4 种）

1. 使用 for 语句循环遍历数组
2. 使用 foreach 语句遍历数组
3. 联合使用 list()、each() 和 while 循环遍历数组
4. 使用数组的内部指针控制函数遍历数组

使用 for 语句循环遍历数组

```
<?php
//数组的遍历（迭代）

//1. 使用for循环遍历索引式数组（特点只适合遍历下标从0开始连续索引式数组）;
$a=array(10,20,30,40,50,60);
$m = count($a); //使用count函数获取数组的长度
for($i=0;$i<$m;$i++){
    echo $a[$i]." ";
}
echo "<br/>";

//1.1 使用for循环遍历特殊数组(重组合索引数组)
$a=array(10,20,30,40,10=>50,60,"a"=>"aaaa");
$b = array_values($a); //获取数组a中所有值，并以索引式数组返回，赋给b
$m = count($b); //使用count函数获取数组的长度
for($i=0;$i<$m;$i++){
    echo $b[$i]." ";
}

//print_r($a);
//echo "<br/>";
//print_r(array_values($a)); //获取数组a中所有值，并以索引式数组返回
```

*使用 foreach 语句遍历数组

foreach 循环结构：

foreach 仅用于数组，有两种语法。

```
foreach (array_expression as $value)
    ...
foreach (array_expression as $key => $value)      ...

```

第一种格式 遍历给定的 array_expression 数组。每次循环中，当前单元的值被赋给 \$value 并且数组内部的指针向前移一步

第二种格式 做同样的事，只除了当前单元的键值也会在每次循环中被赋给变量 \$key。

注：当 foreach 开始执行时，数组内部的指针会自动指向第一个单元。此外注意 foreach 所操作的是指定数组的一个拷贝，而不是该数组本身。

```
//遍历数组方式2：使用foreach遍历
$a = array(10,20,'a'=>30,10=>40,50,60);
//使用foreach遍历数组a，不带键值（下标）
foreach($a as $value){
    echo $value." ";
}
```

```
//使用foreach遍历数组a，带键值（下标）
foreach($a as $k=>$value){
    echo $k."=>".$value." ";
}
```

联合使用 list()、each()和 while 循环遍历数组

1. list()--直接解析赋过来的数组中的索引下标对应的值，解析是从 0 开始

```
//将数组转为变量
list($x,$y,$z,$z2,$z3)=array(10,20,30,40,50); //将数组中的10赋给变量x，数组中的20赋给y
echo $x."/".$y."/".$z."<br/>";//10:20:30

list($a,$b)=array("a"=>"aa","b"=>"bb"); //list只封装索引式数组值
echo $a."/".$b."<br/>";// 没有值

list($a,$b,$c)=array(2=>"aa",0=>"bb"); //list封装是按索引值顺序值来封装
echo $a."/".$b."/".$c."<br/>";//bb::aa
```

2. each()--获取当前数组指针位置的键和值，并以关联和索引两种方式返回，并且指针向下移动一位。

```
$a=array("name"=>"zhangsan","age"=>20,"sex"=>"man");
$list = each($a);
$list = each($a);
$list = each($a);
reset($a); //将数组指针移至首位
$list = each($a);
print_r($list);
//Array ( [1] => zhangsan [value] => zhangsan [0] => name [key] => name )
```

遍历数组方式：联合 while each reset list 遍历数组（半自动方式）

```
$a=array("name"=>"zhangsan","age"=>20,"sex"=>"man");

while($list = each($a)){
    echo $list["key"].":".$list["value"]."<br/>";
}
echo "<hr/>";

reset($a); //将数组指针移至首位(保证数据的完整性)
while(list($k,$v) = each($a)){
    echo $k."/".$v."<br/>";
}
```

//使用 each 获取数组 a 当前位置上的键和值分别赋给 k 和 v 变量，并且数组指针向下移动一位。

使用下面的函数来遍历数组(纯手工遍历数组)

reset()-- 将数组的内部指针指向第一个单元 end() -- 将数组的内部指针指向最后一个单元
next() -- 将数组中的内部指针向前移动一位 prev() -- 将数组的内部指针倒回一位
current() -- 返回数组中的当前单元 key() -- 从关联数组中取得键名

```
<?php
$a = array(10,20,30,40,50,60);
reset($a);//将数组指针移至首位
do{
    echo key($a)."=>".current($a)."<br/>"; //获取键和值
}while(next($a));
```

总结：

一、 基本概念

1. 数组的概念：

*2. 数组的分类：根据数组的下标(索引值)类型不同分为：

索引式数组：数组下标都是整数的， 默认数组的索引下标是从 0 开始

关联式数组：数组下标是以字串表示的 (在其他强类型语言中有的称这个叫集合)

3.php 中数组的定义：

*(1). 直接赋值： \$数组名[下标] = "值";

```
echo "<p>=====索引式数组=====</p>";
$a[] = 10;
$a[] = 20;
$a[] = 30;
$a[10] = "cc";
$a[] = "bb";
```

```
echo "<p>=====关联式数组=====</p>";
$p["name"] = "张三";
$p["sex"] = "男";
$p["age"] = 20;
$p["name"] = "李四"; //修改
```

*(2). 使用 array 函数来赋值。

```
$a = array(10,20,30,40); //使用 array 函数定义一个数组 (未指定下标)
$b = array("aa","bb","cc"); //使用 array 函数定义一个数组 (未指定下标)
$c = array(1=>10,2=>20,6=>80); //使用 array 函数, 定义一个指定下标的数组
```

```
$stu = array("name"=>"张三", "age"=>20); //使用 array 函数, 定义一个指定关联下标数组
```

```
$stu = array("name"=>"张三", "age"=>20, 3=>"qq", "pp"); //这是 pp 的下标是 4
```

```
$p = array("name"=>"张三", "age"=>20, "aa", "bb"); //这时数组 aa 和 bb 的下标为 0,1
```

(3)、 使用其他函数返回一个数组。

4. 多维数组：当一个数组中的元素单元还是一个数组时，称为多维数组。 (几层就是几维)

如： echo \$a[10]; //一维数组

```
echo $a[1][2]; //二维数组
```

```
$a = array(10,20,40,50); //一维数组
```

```
$b = array( array(10,20), array(30,40)...); //二维数组
```

如二维数组的赋值:

直接赋值: \$a[][]=10;

二、数组的遍历(迭代)输出 (4 种)

1. 使用循环结构: for/while/do...while

使用到一个函数: count()-- 获取数组的长度 (元素的个数)

示例:

```
$a=array(10,20,30,40,50,60);
$ll=count($a);
for($i=0;$i<$ll;$i++){
    echo $a[$i]." ";
}
```

注意: 使用for 循环只适合输出索引式数组, 并且数组下标是**从 0 开始连续**的整数。
但是可以使用 array_values () 函数来弥补数组下标不足的地方。

示例:

```
echo "<br/>";
//输出非规则的数组
$a=array(2=>10,3=>40,6=>80,4=>30,"name"=>"张三");
//获取数组中的所有值, 并以索引式数组返回
$list = array_values($a);
//使用for 循环遍历
for($i=0;$i<count($list);$i++){
    echo $list[$i]." ";
}
```

*2. 使用 foreach 遍历数组

格式: foreach(被遍历的数组 as [键=>]值){

```
....  
}
```

示例: //关联式数组的输出

```
$stu = array("name"=>"zhangsan","age"=>20,"sex"=>"man");
foreach($stu as $k=>$v){
    echo "{$k}=>{$v}<br/>";
}
```

3. 联合 while each reset list 遍历数组 (半自动方式)

each()-- 获取当前数组指针位置的键和值, 并以关联和索引两种方式返回, 并且指针向下移动一位。
reset()-- 将数组指针移至首位。

list()-- 直接解析赋过来的数组中的索引下标对应的值, 解析是从 0 开始

```
list($v1,$v2)=array("key"="name",0=>"name","value"="zhangsan",1=>"zhangsan");
```

```
echo $v1.".$v2; //name:zhangsan
```

示例:

```
//使用while、each、reset、list 解析遍历的结果
$stu = array("name"=>"zhangsan", "age"=>20, "sex"=>"man");
reset($stu);
while(list($k,$v)=each($stu)){
    echo "{$k}>{$v}<br/>";
}
```

4. 使用下面的函数来遍历数组(纯手工遍历数组)

reset()-- 将数组的内部指针指向第一个单元
end() -- 将数组的内部指针指向最后一个单元
next() -- 将数组中的内部指针向前移动一位
prev() -- 将数组的内部指针倒回一位
current() -- 返回数组中的当前单元
key() -- 从关联数组中取得键名

do....while 循环

示例:

```
$a = array(10,20,30,40,50,60);
reset($a);//将数组指针移至首位
do{
    echo key($a)."=>".current($a)."<br/>"; //获取键和值
}while(next($a));
```

5. 预定义数组（超全局数组）

\$GLOBALS 包含以下所有信息

***\$_SERVER** 服务器和执行环境信息

***\$_GET** 通过 URL 参数传递给当前脚本的变量的数组。

***\$_POST** 通过 HTTP POST 方法传递给当前脚本的变量的数组。

***\$_FILES** 保存文件上传信息（在文件处理章节中细讲）

（cookie 和 session 在会话跟踪章节中细讲）

***\$_COOKIE** 通过 HTTP Cookies 方式传递给当前脚本的变量的数组（用于储存论坛、文库、博客等登陆信息）

***\$_SESSION** 当前脚本可用 SESSION 变量的数组。（用于网站购物车等的信息存储）

***\$_REQUEST** 包含 get、post 和 cookie

\$_ENV 存储的是系统环境变量信息

***\$_SERVER["HTTP_REFERER"]**--上一页面的 url 地址

\$_SERVER["SERVER_NAME"]--服务器的主机名

***\$_SERVER["SERVER_ADDR"]**--服务器端的 IP 地址

```

$_SERVER["SERVER_PORT"]--服务器端的端口
*$_SERVER["REMOTE_ADDR"]--客户端的 IP
$_SERVER["DOCUMENT_ROOT"]--服务器的 web 目录路径
*$_SERVER["REQUEST_URI"];//--URL 地址
echo $_GET["name"];
echo $_REQUEST["name"]; //获取信息比上面 get 的会慢一些

```

注意：

form 表单的 get 提交方式：url 地址可见，相对不安全，长度受限 2KB，可以做为标签连接使用。
 form 表单的 post 提交方式：url 地址不可见，相对安全，长度不受限。

6、* 数组的相关处理函数

- 1 数组的键/值操作函数 2 统计数组元素的个数与唯一性 3 使用回调函数处理数组的函数
- 4 数组的排序函数 5 拆分、合并、分解与结合数组 6 数组与数据结构
- 7 其他有用的数组处理函数

1. 数组的键和值的操作函数

*array_values — 返回数组中所有的值 array_keys — 返回数组中所有的键名
 array_flip — 交换数组中的键和值 *in_array — 检查数组中是否存在某个值
 array_reverse — 返回一个单元顺序相反的数组 *is_array() --判断是否是数组

```

$a = array("a"=>"aaa", "b"=>"bbb", "c"=>"ccc");
echo "原数组值: ";
print_r($a);
echo "<br/>";

echo "通过函数array_keys获取数组中所有的键:";
print_r(array_keys($a));
echo "<br/>";

```

```

echo "通过函数array_values获取数组中所有的值:";
print_r(array_values($a));
echo "<br/>";

```

```

//自定义一个array_values函数
function my_array_values($list){
    $res=array();
    foreach($list as $v){
        $res[]=$v;
    }
    return $res;
}

```

```

//自定义一个array_keys函数
function my_array_keys($list){
    $res=array();
    foreach($list as $k=>$v){
        $res[]=$k;
    }
    return $res;
}

```

```
<?php
//array_flip - 交换数组中的键和值
$a = array("a"=>"aaa","b"=>"bbb","c"=>"ccc","d"=>"aaa");
echo "原数组值: ";
print_r($a);
echo "<br/>";

echo "键值交换: ";
print_r(array_flip($a));
echo "<br/>";

//自定义array_flip函数
function my_array_flip($list){
    $res=array();
    foreach($list as $k=>$v){
        $res[$v]=$k;
    }
    return $res;
}
```

```
<?php
/*in_array - 检查数组中是否存在某个值
$a=array(10,20,30,40,50,60);
//判断20是否在数组a中,true表示检查类型
//if(in_array("20",$a,true)){
if(in_array("20",$a)){
    echo "yes";
}else{
    echo "no";
}

//自定义函数in_array()
function my_in_array($m,$list,$b=false){
    foreach($list as $v){
        if($b){
            if($m===$v){
                return true;
            }
        }else{
            if($m==$v){
                return true;
            }
        }
    }
    return false;
}
```

```
function myarr($ar){  
    $m = array();  
    end($ar);  
    do{  
        $m[key($ar)]=current($ar);  
    }while(prev($ar));  
    return $m;  
}
```

```
function myarr($m){  
    if(gettype($m)=="array"){  
        echo '是数组';  
    }else{  
        echo '不是数组';  
    }  
    //考虑使用switch ... case ...  
}
```

2. 数组的统计相关函数

*count -- 计算数组中的单元数目或对象中的属性个数

array_count_values -- 统计数组中所有的值出现的次数（考虑自定义类比冒泡排序）

array_unique -- 移除数组中重复的值（考虑自定义类比冒泡排序）

```
//递归统计数组单元个数(模拟count())  
$arr = array('aa'=>10,'5'=>50,60,70,array(1,2,3));  
function myarr($ar,$b=false){  
    $num=0;  
    foreach($ar as $v){  
        if(is_array($v) && $b==true){  
            $num+=myarr($v);  
        }else{  
            $num++;  
        }  
    }  
    return $num;  
}
```

3. 带回调函数的

array_filter -- 用回调函数过滤数组中的单元（常用）

array_walk -- 对数组中的每个成员应用用户函数

array_map -- 将回调函数作用到给定数组的单元上

array_filter -- 用回调函数过滤数组中的单元

语法: ray array_filter (array input [, callback callback])

1. 依次将 input 数组中的每个值传递到 callback 函数。如果 callback 函数返回 TRUE，则 input 数组的当前值会被包含在返回的结果数组中。数组的键名保留不变。

2. 如果没有提供 callback 函数，array_filter() 将删除 input 中所有等值为 FALSE 的条目。

```
//array_filter -- 用回调函数过滤数组中的单元
$a = array(70,80,58,59,90,79,98,40,58);

//将数组中的每个值遍历出来每个值递给回调函数处理
$res = array_filter($a,"fun");

print_r($res);

//过滤的回调函数，目的是指定过滤规则
function fun($m){
    //echo "ff:$m.<br/>";//检测回调函数参数
    if($m>=60){
        return true;
    }else{
        return false;
    }
}
```

```
//定义过滤规则回调函数
function docheck($m){
    if($m['classid']=="lamp90"){
        return true;
    }else{
        return false;
    }
}

function my_filter($n,$func){
    $arr = array();
    foreach($n as $k => $v){
        if($func($v)){
            $arr[$k]=$v;
        }
    }
    return $arr;
}

print_r(my_filter($list,"docheck"));
```

array_walk--对数组中的每个成员应用用户函数

语法: `bool array_walk (array &array, callback funcname [, mixed userdata])`

1. 如果成功则返回 TRUE, 失败则返回 FALSE。
2. 将用户自定义函数 `funcname` 应用到 `array` 数组中的每个单元。典型情况下 `funcname` 接受两个参数。`array` 参数的值作为第一个, 键名作为第二个。如果提供了可选参数 `userdata`, 将被作为第三个参数传递给 `callback funcname`。
3. `array_walk()`不会受到 `array` 内部数组指针的影响。`array_walk()` 会遍历整个数组而不管指针的位置。

```
//不及格的分数加2分
$list = array(70,80,60,54,58,40,36,90,75);
array_walk($list, "fun");
function fun(&$n){
    if($n<60){
        $n+=2;
    }
}

function myarr(&$arr){
    foreach($arr as $k => $v){
        if($v<60){
            $arr[$k]=$v+2;
        }
    }
    return $arr;
}

function myarr($arr,$func){
    foreach($arr as $k => $v){
        $arr[$k]=func($v);
    }
    return $arr;
}

function func(&$m){
    if($m<60){
        $m+=2;
    }
    return $m;
}
```

array_map-- 将回调函数作用到给定数组的单元上

语法: `array array_map (callback callback, array arr1 [, array ...])`

`array_map()` 返回一个数组, 该数组包含了 `arr1` 中的所有单元经过 `callback` 作用过之后的单元。`callback` 接受的参数数目应该和传递给 `array_map()` 函数的数组数目一致。

```
<?php
//array_map -- 将回调函数作用到给定数组的单元上
$list = array(70,80,58,59,90,79,98,40,58);
$new = array_map("fun",$list);
print_r($new);

//回调函数
function fun($m){
    if($m<60){
        $m+=2;
    }
    return $m;
}
```

4. 数组的排序

*`sort`-- 对数组排序(升序)

```

rsort -- 对数组逆向排序（降序）
asort -- 对数组进行排序并保持索引关系（关联数组排序）
arsort -- 对数组进行逆向排序并保持索引关系
*usort -- 使用用户自定义的比较函数对数组中的值进行排序
uasort -- 使用用户自定义的比较函数对数组中的值进行排序并保持索引关联
ksort -- 对数组按照键名排序
krsort -- 对数组按照键名逆向排序
uksort -- 使用用户自定义的比较函数对数组中的键名进行排序
*natsort -- 用“自然排序”算法对数组排序
natcasesort -- 用“自然排序”算法对数组进行不区分大小写字母的排序
array_multisort -- 对多个数组或多维数组进行排序（了解）--按照第一维度排序后面对应排序

krssort($a); //对数组的下标进行降序(k表示对下标进行排序)

asort($a); //对数组a进行升序排序(a表示保持索引值不变)

arsort($a); //对数组进行降序(a表示保持索引值不变)

sort($a); //对数组a进行升序排序

rssort($a); //对数组进行降序

usort($list,"fun");

//自定义排序规则回调函数
function fun($a,$b){
    if($a["age"]<$b["age"]){
        return 1; //交换顺序
    }else{
        return -1;
    }
}

$aa=array("id8","id100","id28");
sort($aa); //普通排序（升序）（字符串排序原则会将各自提取对应位数进行比较）
natsort($aa); //自然排序（升序）（按照字符串中的数值排序）

Array ( [0] => id8 [1] => id100 [2] => id28 )
Array ( [0] => id100 [1] => id28 [2] => id8 )
Array ( [2] => id8 [1] => id28 [0] => id100 )

```

5. 拆分、合并、分解与结合数组

array_slice -- 从数组中取出一段

```
$aa=array(10,5=>20,30,40,50,60,70);
```

```
$bb = array_slice($aa,3); //从3号（默认0开始）位置开始取
print_r($bb); //40,50,60,70
```

```
$b = array_slice($a,-3); //取后三个
print_r($b); //50,60,70
$b = array_slice($a,2,3,true); //从索引位置2开始取3个(true:保持原有下标)
print_r($b); // [6] => 30 [7] => 40 [8] => 50
```

array_splice -- 把数组中的一部分去掉并用其它值取代

```
//将数组的索引位置2开始换掉3个，将数组c放进去
$a=array(10,5=>20,30,40,50,60,70);
$c=array("a","b","c","d");
$b=array_splice($a,2,3,$c); //直接在数组a上操作替换，并将替换掉的部分返回给b
print_r($a);//[0] => 10 [1] => 20 [2] => a [3] => b [4] => c [5] => d [6] => 60 [7] => 70
print_r($b);//[0] => 30 [1] => 40 [2] => 50
```

array_combine -- 创建一个数组，用一个数组的值作为其键名，另一个数组的值作为其值

*array_merge -- 合并一个或多个数组

```
<?php
//数组的合并
//array_combine -- 创建一个数组，用一个数组的值作为其键名，另一个数组的值作为其值
/*array_merge -- 合并一个或多个数组

$a=array("a","b","c");
$b=array(100,200,300);

$c = array_combine($a,$b); //将a作为键，将b作为值合并成一个新数组（要求ab数组长度必须一致）
print_r($c);//Array ([a] => 100 [b] => 200 [c] => 300 )

echo "<br/>";
$c = array_merge($a,$b); //将数组的单元进行合并
print_r($c);//Array ([0]=>a [1]=>b [2]=>c [3]=>100 [4]=>200 [5]=>300 )
```

array_intersect -- 计算数组的交集

语法 `array array_intersect (array array1, array array2 [, array ...])`

`array_intersect()` 返回一个数组，该数组包含了所有在 `array1` 中也同时出现在所有其它参数数组中的值。注意键名保留不变。

array_diff -- 计算数组的差集

语法: `array array_diff (array array1, array array2 [, array ...])`

`array_diff()` 返回一个数组，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。

=====

6. 数组与数据结构(栈是指向堆的名称，栈是等宽的，堆是内存中容器空间，队列先进先出)

array_pop -- 将数组最后一个单元弹出（出栈）

语法: `mixed array_pop (array &array)`

`array_pop()` 弹出并返回 `array` 数组的最后一个单元，并将数组 `array` 的长度减一。如果 `array` 为空（或者不是数组）将返回 `NULL`。

array_push -- 将一个或多个单元压入数组的末尾（入栈）

语法: `int array_push (array &array, mixed var [, mixed ...])`

`array_push()` 将 `array` 当成一个栈，并将传入的变量压入 `array` 的末尾。`array` 的长度将根据入栈变量的数目增加。

array_shift -- 将数组开头的单元移出数组

语法: mixed array_shift (array &array)

array_shift() 将 array 的第一个单元移出并作为结果返回，将 array 的长度减一并将所有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。如果 array 为空（或者不是数组），则返回 NULL。

array_unshift -- 在数组开头插入一个或多个单元

语法: int array_unshift (array &array, mixed var [, mixed ...])

array_unshift() 将传入的单元插入到 array 数组的开头。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。

7. 其他有用的数组处理函数

array_rand -- 从数组中随机取出一个或多个单元

shuffle -- 将数组打乱

file_put_contents(); //文件的写操作

file_get_contents(); //文件的读操作

六、字符串处理

字符串的处理介绍

字符串的处理方式

字符串类型的特点

常用的字符串输出函数

常用的字符串格式化函数

字符串比较函数

1. 字符串的处理方式

在 C 语言中字符串是作为字节数组处理的。在 Java 语言中字符串是作为对象处理的。而 php 则把字符串作为基本数据类型来处理。通常对字符串的处理涉及字符串的格式化。字符串的分割和连接、字符串的比较、以及字符串的查找、匹配和替换。

2. 常用的字符串输出函数

常用的输出字符串函数:

echo()	-- 输出字符串 语言结构,不是函数	echo 1,2; //ok
print()	-- 输出一个字符串	print 1,2;//error

以上两者区别: echo 脚本继承的, printC 语言继承的, echo 支持同时输出多个字符, print 不支持

die() -- 输出一条消息，并退出当前脚本 (exit 别名)

printf() -- 输出格式化字符串

sprintf() -- 把格式化的字符串写入一个变量中

如: echo 'aaa', 'bbbb', 'ccc'; //输出多个值

\$link=@mysql_connect(" ", " ", " ") or die("失败")

printf 与 sprintf 都是格式化字符串：

字符串转换格式：

%%	返回百分比符号
%b	二进制数
%c	依照 ASCII 值的字符
%d	带符号十进制数
%e	可续计数法（如 1.5e3）
%u	无符号十进制数
%f 或%F	浮点数
%o	八进制数
%s	字符串
%x 或%X	十六进制数

<?php

```
printf("123的二进制是: %b <br/>",123);
printf("123的八进制是: 0%o ; 十六进制是: 0x%x<br/>",123,123);
printf("123.456789的小数保留: %0.3f <br/>",123.456789);
printf("65的对应的字符是: %c",65);
```

123的二进制是: 1111011

123的八进制是: 0173 ; 十六进制是: 0x7b

123.456789的小数保留: 123.457

65的对应的字符是: A

3. 常用的字符串格式化函数

去除空格和字符串填充补函数

字符串大小写的转换

和 HTML 标签相关联的字符串格式化

其他字符串格式化函数

》去除空格和字符串填充补函数

函数:ltrim()

语法: string ltrim(string str[, string charlist]);

返回值: 字符串

本函数用来删去字符串中的前导空格 (whitespace)。

函数:rtrim() (还有个别名: chop ())

语法: string rtrim(string str[, string charlist]);

返回值: 字符串

本函数用来删去字符串中的后缀空格 (whitespace)。

*函数:trim()

截去字符串首尾的空格。

语法: string trim(string str[, string charlist]);

返回值: 字符串

本函数返回字符串 string 首尾的空白字符去除后的字串。

```

<?php
    //字符串的处理函数: ltrim、rtrim、trim
    $s = " aa ";
    echo "#". $s . "#<br/>";
    echo "#". trim($s) . "#<br/>";
    echo "#". ltrim($s) . "#<br/>";
    echo "#". rtrim($s) . "#<br/>";
    |

    //使用":"将上面a数组连成一个字符串
    $a=array(10,20,30,40,50,60);
    $str="";
    foreach($a as $v){
        $str.=$v.":";
    }
    echo rtrim($str,":"); //去除右侧多余":"
    /*常用在一次传多值
    # aa #
    #aa#
    #aa #
    # aa#
    10:20:30:40:50:60
    */

```

函数: str_pad() 按需求对字符串进行填充。

语法: string str_pad (string input, int pad_length [, string pad_string [, int pad_type]])

STR_PAD_LEFT 字符串左添补

STR_PAD_RIGHT 字符串右添补

STR_PAD_BOTH 字符串两端添补

```

<?php
//str_pad() 使用另一个字符串填充字符串为指定长度
//将aa字符串用0填充为4个长度按照两侧填
//echo str_pad("aa",4,"0",STR_PAD_BOTH);

//利用字符串填充来实现学号的制定
for($i=1;$i<100;$i++){
    echo "2013".str_pad($i,4,"0",STR_PAD_LEFT)."<br/>";
}

```

》字符串大小写的转换

*函数: strtolower()

语法: string strtolower(string str);

本函数将字符串 str 全部变小写字符串。

*函数: strtoupper()

语法: string strtoupper(string str);

本函数将字符串 str 全部变大写字符串。

函数: ucfirst()

将字符串第一个字符改大写。

语法: string ucfirst(string str);

本函数返回字符串 str 第一个字的字首字母改成大写。

函数: ucwords()

将字符串每个字第一个字母改大写。

语法: string ucwords(string str);

本函数返回字符串 str 每个字的字首字母全都改成大写。

```
<?php
//字符的处理
/*
函数: strtolower( )
    语法: string strtolower(string str);
    本函数将字符串 str 全部变小写字符串。
函数: strtoupper( )
    语法: string strtoupper(string str);
    本函数将字符串 str 全部变大写字符串。
*/
$str="Hello PHP!";
echo $str."<br/>";
echo strtolower($str)."<br/>"; //小写
echo strtoupper($str)."<br/>"; //大写

$str = "string's first character lowercase";
echo $str."<br/>";
echo ucfirst($str)."<br/>"; //将第一单词的首字符大写
echo ucwords($str)."<br/>"; //将每个单词的首字母大写

//函数 int ord(string $string) – 返回字符的 ASCII 码值
//函数 string chr(int $ascii) – 返回指定的字符

echo "65ASCII码对应的字符是: ".chr(65)."<br/>";
echo "A字母对应的ASCII码值是: ".ord("A")."<br/>";
```

》 和 HTML 标签相关联的字符串格式化

函数: nl2br()

语法: string nl2br (string string)

将字符串中”\n”转成 HTML 换行符 “
”

*函数: htmlspecialchars()

语法 : string htmlspecialchars (string string [, int quote_style [, string charset]])

把指定特殊符号转换成实体，如<&>

'&' : '&' '\"' : '"' '>' : '>' "" : ''' '<' : '<'

函数: htmlentities()

语法 : string htmlentities (string string [, int quote_style [, string charset]])

可以将所有的非 ASCII 码转换成对应实体代码。

函数: string strip_tags()

语 法 : string strip_tags(string str [, string allowable_tags])
删除 HTML 的标签函数

》 其他字符串格式化函数

函数: strrev()

颠倒字符串。将字符串前后颠倒。

语法: string strrev(string string);

函数: strlen()

取得字符串长度。

语法: int strlen(string str);

本函数返回指定的字符串长度。

函数: number_format()

语 法 : string number_format (float number [, int decimals [, string dec_point, string thousands_sep]])

格式货币、数字、时间等。

函数: md5() 加密函数

格式: string md5 (string str [, bool raw_output])

```
<?php
$str="abcdef";
echo strrev($str).'  
';
//自定义反转函数
function my_strrev($string){
    $strLen = strlen($string);
    $newStr = '';
    for($i = ($strLen-1) ; $i >= 0 ; $i--){
        $newStr .= $string[$i];
    }
    return $newStr;
}
echo my_strrev($str);
```

```
<?php
$str = "1234567891.3476";
echo number_format($str,2).'  
';
echo number_format($str);
echo '

---

';
function my_str($number){
    $string = (string)$number;
    $strLen = strlen($string);
    $k = $strLen%3;
    $newString = '';
    for($i=0 ; $i < $strLen ; $i++){
        if($i % 3 == $k && $i!=0){
            echo ',';
        }
        echo $string[$i];
    }
}
echo my_str(1234567891);
```

4. 字符串比较函数

按字节顺序进行字符串比较

按自然顺序进行字符串比较

PHP 中有多种方法可以对字符串进行比较，除了可以直接使用条件运算符 (<, >, ==) 加以比较外。

按字节进行字符串的比较

strcmp(); (整个比较是否相同)

strncmp(); (指定比较长度的字符串比较)

strcasecmp(); (不区分大小写的全串比较)

按自然排序法时行字符串的比较

strnatcmp();

字符串的模糊比较

similar_text(); (返回相似字符串的个数)

```
<?php
//字符串比较:
/*
    strcmp -- 整个比较是否相同
    strncmp --指定长度的比较
   strcasecmp-- 不区分大小写比较
    |
    strnatcmp -- 按自然顺序比较
    similar_text -- 模糊比较
*/
$s1 = "abe";
$s2 = "abd";
|
//比较字符串是否相同: 0相同, -1表示后面的大,1表示前面的大
var_dump(strcmp($s1,$s2));
var_dump(strncmp($s1,$s2,2)); //比较前2个字符是否相同

$s1="wertyuiopvcxrttylopa,uhaw tbwryt werwertwertwer";
$s2="wertyuiopvcxrttylopo,uhaw tbwryt werwertwertwer";

//相似度
echo ((similar_text($s1,$s2)/strlen($s1))*100)."%";
```

5. 字符串的分割与拼装

explode () -- 使用一个字符串分割另一个字符串

语法: array explode (string separator, string string [, int limit])

此函数返回由字符串组成的数组，每个元素都是 string 的一个子串，它们被字符串 separator 作为边界点分割出来。如果设置了 limit 参数，则返回的数组包含最多 limit 个元素，而最后那个元素将包含 string 的剩余部分。

implode () -- 用一组较小的字符串创建一个大字符串。 str_split — 将字符串转换为数组

```
<?php
    $str = "Hello Friend" ;
    |
    $arr1 = str_split ( $str );
    $arr2 = str_split ( $str , 3 );
    echo '<pre>';
    print_r ( $arr1 );
    print_r ( $arr2 );
)
Array
(
[0] => H
[1] => e
[2] => l
[3] => l
[4] => o
[5] =>
[6] => F
[7] => r
[8] => i
[9] => e
[10] => n
[11] => d
)
Array
(
[0] => He
[1] => lo
[2] => Fri
[3] => end
)
```

格式: string implode (string glue, array pieces)

第一个参数 glue 是放在第二个参数 pieces 的元素之间的字符串。可以像下面这样重建简单的逗号分隔的字符串。

```
<?php
/* explode -- 字符串拆分函数
/* implode -- 字符串组合函数
$str="10,20,30,40,50,60";echo $str."<br/>";
$arr = explode(",",$str); //将字符串str以","号方式拆分成数组
print_r($arr);echo "<br/>";

$s = implode(":",$arr); //将数组a以“：“号分隔符组装成字符串
echo $s;echo "<hr/>";

//将下面字符串拆分，并以表格输出
$str="张三##男##20@@李四##女##21@@王五##男##22";

//1.先一个"@@"符拆分
$list = explode("@@",$str);
echo "<table width='400' border='1'>";
//2. 遍历
foreach($list as $v){
    echo "<tr>";
    //3.对每条学生信息，使用“##”拆分
    $stu = explode("##",$v);
    foreach($stu as $value){
        echo "<td>{$value}</td>";
    }
    echo "</tr>";
}
echo "</table>";
```

10, 20, 30, 40, 50, 60

Array ([0] => 10 [1] => 20 [2] => 30 [3] => 40 [4] => 50 [5] => 60)

10:20:30:40:50:60

| | | |
|----|---|----|
| 张三 | 男 | 20 |
| 李四 | 女 | 21 |
| 王五 | 男 | 22 |

6. 字符串的截取

函数: substr() -- 取部份字符串。

语法: string substr(string string, int start, int [length]);

返回值: 字符串

本函数将字符串 string 的第 start 位起的字符串取出 length 个字符，若省略参数 length，则

取到字符串末尾。若 start 为负数，则从字符串尾端往前开始提取。如果 length 为整数，表示返回 length 个字符，若为负数，则表示取到倒数第 length 个字符。

```
<?php
//字符串截取函数:
/*substr -- 截取字符串 a.txt a.rm a.rmvb a.b.txt
   string 返回值 substr(
      string string 被截取字符串,
      int start 起始位置
      [, int length 长度] )
mb_substr() -- 截取指定编码的字符串*/
$s = "zhangsan.txt";

echo substr($s,5)."<br/>"; //san.txt 从索引位置5开始截取到最后
echo substr($s,2,3)."<br/>"; //ang 从索引位置2开始截取3个长度
echo substr($s,-3)."<br/>"; //txt 从倒数第3个位置开始，截取到最后
echo substr($s,-5,-3)."<br/>"; //n. 从倒数第5个位置开始，截取到最后，去掉最后3个
echo substr($s,-5,2)."<br/>"; //n. 从倒数第5个位置开始，截取2个
echo substr($s,3,2)."<br/>"; //ng 从索引位置3开始，截取2个
echo substr($s,3,-3)."<br/>"; //ngsan. 从索引位置3开始，截取到最后，去掉最后3个

echo "<hr/>";

$str="a字符串截取函数";
echo mb_substr($str,0,4,"UTF-8"); //a字符串  mb_substr — 获取字符串的一部分
```

7. 字符串的查找

函数: strstr() 别名: strchr() --- 返回字符串中某字符串开始处至结束的字符串。

语法: string strstr(string haystack, string needle);

返回值: 字符串

本函数将 needle 最先出现在 haystack 处起至 haystack 结束的字符串返回。若找不到 needle 则返回 false。

函数: strrchr() --- 取得某字符最后出现的位置。

语法: string strrchr(string haystack, string needle);

本函数用来寻找字符串 haystack 中的字符 needle 最后出现位置，并将此位置起至字符串 haystack 结束之间的字符串返回。若没有找到 needle 则返回 false。

```
<?php
/*strstr -- 字符串查找并截取
   strchr -- 是上面的strstr的别名找并截取
   strrchr -- 从后面做字符串查找。找并截取*/
$str="qwertyuiopabcsadfaabcsdf";

echo strstr($str,"abc")."<br/>";//abcuiopabcsadfaabcsdf //从前面找
echo strrchr($str,"abc")."<br/>";//abcsdf //从后面找
```

函数:strpos() -- 寻找字符串中某字符最先出现的位置。默认从 0 开始。

语法: int strpos(string haystack, string needle, int [offset]);

本函数用来寻找字符串 haystack 中的字符 needle 最先出现的位置。若找不到指定的字符，则返回 false 值。参数 offset 可省略，用来表示从 offset 开始找。

函数:strrpos() -- 寻找字符串中某字符最后出现的位置。

语法: int strrpos(string haystack, char needle);

返回值: 整数

本函数用来寻找字符串 haystack 中的字符 needle 最后出现的位置。若找不到指定的字符，则返回 false 值。

```
<?php
/* strpos --查找一个字符出现索引位置（从前开始查找。第三个参数为查找起始位置）
   strrpos --查找一个字符出现索引位置（从后面开始查找。） */
|
$str="zhangsan";
echo strpos($str,"a");//2 //从前面找a字母出现的位置(默认从头0位置开始找)
echo strpos($str,"a",3);//6 //从前面找a字母出现的位置(从3位置开始找)
echo strrpos($str,"a");//6 //从后面找a字母出现的位置

echo "<hr/>";

//实例：截取文件名的后缀名

$filename="a.b.html";

//先从文件名后找点出现的位置开始找，然后再截取
echo substr($filename,strrpos($filename,".")+1);//html
```

8. 字符串的替换

str_replace()---将第三个参数里面出现在第一个参数的字符换成第二个参数字符

字符串替换，三种替换方式

str_replace(string \$search, string \$replace, string \$str);

str_replace(array \$search, string \$replace, string \$str);

str_replace(array \$search, array \$replace, string \$str);

```
<?php
//换成字串 str_replace(被换字串, 换成的字串, 原字串,[数量]);字符串替换
$str="hello php!";

echo str_replace("php","world",$str);//hello world!
//将$str字串中出现的php，都换成world

$s = "10,20,30;40,50;60";
echo str_replace(array(",",";"),":",$s);//10:20:30:40:50:60
//将s字串中的","和";"都换成":""

echo str_replace(array(",",";"),array(":","!"),$s);//10:20:30!40:50!60
//将s字串中","换成":" ;"换成"!"。注意：要求两个参数数组长度必须一致。
```

```
<?php
/*
    file_get_contents("文件名") // 获取指定文件的内容
    file_put_contents("文件名","内容");//将内容写入(覆盖写)到指定文件中
    |
    //将内容写入(追加写)到指定文件中
    file_put_contents("文件名","内容",FILE_APPEND)
*/

//file_put_contents("a.txt","hello world!"); //清空写
//file_put_contents("a.txt","hello world!",FILE_APPEND); //追加写

$str = file_get_contents("a.txt");

echo $str;
```

=====

PHP -- 字串的处理总结

=====

一、字串的定义

-
1. 单引号: '' 不支持变量的解析, 转义符: \\'
 2. 双引号: "" 支持变量的解析, 转义符: \\n \\r \\t \\\" \\\$ || ...
 3. 定界符: <<< 注意结束符的使用。

```
$str = <<<mystr
.....
mystr;
```

二、常用字串的输出函数

-
1. echo() 支持多个变量同时输出 如: echo \$s1,\$s2;
 2. print() 不支持多个变量。
 3. die() 别名 exit() 终止当期脚本执行, 可以顺便输出内容
 4. print_r() 与 var_dump () 函数类似, 都是格式化变量并直接输出, 可以解析数组, 对象等类型
常用于开发时的临时输出使用 (测试)
 5. printf() 格式化字符串并输出
 6. sprintf() 格式化字符串并返回

命名: 驼峰命名法

函数名: strDel()

userNameAdd()

类名: *UserAction* ()

三、常用字串中的处理函数（格式化）

ltrim() 去除左侧多余字符（默认删空格）

rtrim() 去除右侧多余字符（默认删空格）

* *trim()* 去除两侧多余字符（默认删空格）

str_pad() 使用另一个字符串填充字符串为指定长度

* 函数: *strtolower()*

语法: *string strtolower(string str);*

本函数将字符串 *str* 全部变小写字符串。

* 函数: *strtoupper()*

语法: *string strtoupper(string str);*

本函数将字符串 *str* 全部变大写字符串。

函数: *ucfirst()*

将字符串第一个字符改大写。

语法: *string ucfirst(string str);*

本函数返回字符串 *str* 第一个字的字首字母改成大写。

函数: *ucwords()*

将字符串每个字第一个字母改大写。

语法: *string ucwords(string str);*

本函数返回字符串 *str* 每个字的字首字母全都改成大写。

int ord(string \$string) — 返回字符的 ASCII 码值

string chr(int \$ascii) — 返回指定的字符

网页输出格式化函数

nl2br-- 将字串中\n换成
标签，实现换行输出。

* *htmlspecialchars--* 格式换字串中的html 标签

htmlentities--

* *strip_tags --* 删除html 标签函数

strrev -- 将字串颠倒返回

* *strlen --* 求字串长度: 字母:一个算一个长度,汉字: utf-8 编码是每个汉字3 个长度, 其他是2 长度

* *mb_strlen(str,"utf-8");* 获取中文字的长度

number_format -- 格式化数字的

* *md5 --* 单向加密的（不可逆的（不能解密的））。

strcmp -- 整个比较是否相同

strncmp -- 指定长度的比较

strcasecmp-- 不区分大小写比较

strnatcmp -- 按自然顺序比较

similar_text -- 模糊比较

* *explode* -- 字符串拆分函数

* *implode* -- 字符串组合函数

* *substr* -- 截取字符串 *a.txt a.rm a.rmvb a.b.txt*

string 返回值 *substr*(

string string 被截字符串,

int start 起始位置

 [, *int length* 长度])

* *mb_substr()* -- 截取指定编码的字符串

* *strstr* -- 字串查找并截取

strchr -- 是上面的 *strstr* 的别名找并截取

strrchr -- 从后面做字串查找。找并截取

* *strpos* -- 寻早一个字符出现位置 (从前开始。第三个参数为查找起始位置)

strrpos -- 寻早一个字符出现位置 (从后面开始找。)

* 换成字符串 *str_replace(被换字符串, 换成的字符串, 原字符串,[数量]);* 字符串替换

file_get_contents("文件名") // 获取指定文件的内容

file_put_contents("文件名","内容")//将内容写入(覆盖写)到指定文件中

file_put_contents("文件名","内容",FILE_APPEND)

//将内容写入(追加写)到指定文件中

实例：

学生信息管理

|--index.php 浏览学生信息 ok

|

|--add.php 添加学生信息表单页 ok

|

|--doAdd.php 执行学生信息添加操作 ok

|

|--doDel.php 执行学生信息删除操作

|

```
|--menu.php 公共菜单导航栏页 ok
```

```
|
```

```
|--stu.txt 学生信息储存文件 ok
```

```
Add.php
```

```
<html>
```

```
  <head>
```

```
    <title>学生信息管理</title>
```

```
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
  </head>
```

```
  <body>
```

```
    <center>
```

```
      <?php include("menu.php"); //导入导航栏页 ?>
```

```
      <h3>添加学生信息</h3>
```

```
      <form action="doAdd.php" method="post">
```

```
        <table width="240" border="0">
```

```
          <tr>
```

```
            <td>姓名:</td>
```

```
            <td><input type="text" name="name"/></td>
```

```
          </tr>
```

```
          <tr>
```

```
            <td>年龄:</td>
```

```
            <td><input type="text" name="age"/></td>
```

```
          </tr>
```

```
          <tr>
```

```
            <td>性别:</td>
```

```
            <td><input type="radio" name="sex" value="男"/>男
```

```
              <input type="radio" name="sex" value="女"/>女</td>
```

```
          </tr>
```

```
          <tr>
```

```
            <td>班级:</td>
```

```
            <td><input type="text" name="classid"/></td>
```

```
          </tr>
```

```
          <tr>
```

```
            <td colspan="2" align="center">
```

```
              <input type="submit" value="添加"/>&nbsp;&nbsp;
```

```
              <input type="reset" value="重置"/>
```

```
            </td>
```

```
          </tr>
```

```
        </table>
```

```
      </form>
```

```
    </center>
```

```
  </body>
```

```
</html>
Doadd.php
<html>
    <head>
        <title>学生信息管理</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    </head>
    <body>
        <center>
            <?php include("menu.php"); //导入导航栏页 ?>

            <h3>添加学生信息</h3>
            <?php
                //执行学生信息添加

                //1. 获取要添加的学生信息
                $name = $_POST['name'];
                $sex = $_POST['sex'];
                $age = $_POST['age'];
                $classid = $_POST['classid'];
                //2. 拼装信息
                $stu = $name."##".$age."##".$sex."##".$classid."@@";

                //3. 将 stu 信息追加到 stu.txt 文件中
                file_put_contents("stu.txt",$stu,FILE_APPEND);

                //
                echo "添加成功！ ";
            ?>
        </center>
    </body>
</html>
```

```
Index.php
<html>
    <head>
        <title>学生信息管理</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    </head>
    <body>
        <center>
            <?php include("menu.php"); //导入导航栏页 ?>

            <h3>浏览学生信息</h3>
```

```

<table width="500" border="1">
    <tr>
        <th>姓名</th>
        <th>年龄</th>
        <th>性别</th>
        <th>班级</th>
        <th>操作</th>
    </tr>
    <?php
        //读取学生信息，并解析输出
        //1.读取学生信息文件，并去除多余的"@"
        $content = trim(file_get_contents("stu.txt"), "@");

        //2. 以"@@@"符拆分成一条条学生信息
        $list = explode("@@@", $content);

        //3. 遍历 list 信息
        foreach($list as $k=>$v){
            echo "<tr>";
            $stu = explode("##", $v); //将每条学生信息字符串，拆分成一个个数据单元
            echo "<td>{$stu[0]}</td>";
            echo "<td>{$stu[1]}</td>";
            echo "<td>{$stu[2]}</td>";
            echo "<td>{$stu[3]}</td>";
            echo "<td><a href='doDel.php?k={$k}'>删除</a></td>";
            echo "</tr>";
        }
    ?>
</table>
</center>
</body>
</html>

```

```

Dodel.php
<html>
    <head>
        <title>学生信息管理</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    </head>
    <body>
        <center>
            <?php
                include("menu.php"); //导入导航栏页

```

```

//执行学生信息删除

//1.获取要删除的索引号 k
$k = $_GET['k'];

//2.读取学生信息文件,并去除多余的"@"
$content = trim(file_get_contents("stu.txt"), "@");

//3. 以"@@@"符拆分成一条条学生信息
$list = explode("@@@", $content);

//4.删除指定位置指定值
unset($list[$k]);

//5.将数据组装好写回去
$content = implode("@@@", $list). "@@";
file_put_contents("stu.txt", $content);

echo "操作成功！ ";

?>
</center>
</body>
</html>

```

```

Menu.php
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<h2>学生信息管理</h2>
<a href="index.php">浏览信息</a> |
<a href="add.php">添加信息</a>
<hr/>

```

七. 正则表达式（模糊搜索）

1. 正则表达式简介
2. 正则表达式的语法规则
3. 与 Perl 兼容的正则表达式函数

1. 正则表达式简介：

正则表达式是用于描述字符排列和匹配模式的一种语法规则。它主要用于字符串的模式分割、匹配、查找及替换操作。到目前为止，我们前面所用过的精确（文本）匹配也是一种正则表达式。

在 PHP 中，正则表达式一般是由正规字符和一些特殊字符（类似于通配符）联合构成的一个文本模式的程序性描述。

PHP 中，正则表达式有三个作用：

匹配，也常常用于从字符串中析取信息。

用新文本代替匹配文本。

将一个字符串拆分为一组更小的信息块。

一个正则表达式中至少包含一个原子。

1》 在 PHP 中有两套正则表达式函数库，两者功能相似，只是执行效率略有差异：

一套是由 PCRE (Perl Compatible Regular Expression) 库提供的。使用 “preg_” 为前缀命名的函数；

一套由 POSIX (Portable Operating System Interface of Unix) 扩展提供的。使用以 “ereg_” 为前缀命名的函数；

PCRE 来源于 Perl 语言，而 Perl 是对字符串操作功能最强大的语言之一，PHP 的最初版本就是由 Perl 开发的产品。

PCRE 语法支持更多特性，比 POSIX 语法更强大。

举例：想一想这两个正则表达式做什么用？

/^-?\d+\$/^-?0[xX][da-fA-F]+\$/ 正数

/^[-0-9a-zA-Z_-]+@[0-9a-zA-Z_-]+\.(0-9a-zA-Z_-)+\{0,3\}\$/ Email

2》 与 Perl 语言兼容的正则表达式处理函数

函数名	功能描述
preg_match()	进行正则表达式匹配
preg_match_all()	进行全局正则表达式匹配
preg_replace()	执行正则表达式的搜索和替换
preg_split()	用正则表达式分割字符串
preg_grep()	返回与模式匹配的数组单元
preg_replace_callback()	用回调函数执行正则表达式的搜索和替换

2. 正则表达式的语法规则

1 定界符

2 原子

3 元字符

4 模式修正符

Perl 兼容正则表达式

正则表达式作为一个匹配的模版，是由原子（普通字符，例如字符 a 到 z）、特殊字符（元字符，例如*、+ 和 ? 等）、以及模式修正符三部分组成的名字模式。

一个最简单正则表达式至少包含一个原子。

将下面的正则表达式拆分如下：

'/<a.*?(?:\\t|\\r|\\n)?href=[\"?](.+?)["?](?:(?:\\t|\\r|\\n)+.*?)?>(.+?)</a.*?>/sim'

1. 定界符：两个斜线 “/”。
2. 原子用到了< a href=“ “ />等普通字符和\ t \r \n 等转义字符
3. 元字符使用了 [] () | . ? * + 等具有特殊含义的字符
4. 用到了模式修正符是在定界符最后一个斜线之后的三个字符: sim

定界符

在程序语言中，使用与 Perl 兼容的正则表达式，通常都需要将模式表达式放入定界符之间，如“/”。作为定界符常使用反斜线 “/”，如“/apple/”。用户只要把需要匹配的模式内容放入定界符之间即可。作为定界的字符也不仅仅局限于“/”。除了字母、数字和斜线 “\” 以外的任何字符都可以作为定界符，像“#”、“|”、“!”等都可以的。

/<\\w+>/ -- 使用反斜线作为定界符合法
|(\d{3})-\d+|Sm -- 使用竖线 “|” 作为定界符合法
!^(?i)php[34]! -- 使用竖线 “!” 作为定界符合法
{^s+(\s+)?\$} -- 使用竖线 “}” 作为定界符合法
/href= ‘(.*)’ -- 非法定界符，缺少结束定界符
1-\d3-\d3-\d4| -- 非法定界符，缺少其实定界符

原子（一般定界符以内）

原子是正则表达式的最基本的组成单元，而且在每个模式中最少要少包含一个原子。原子是由所有那些未显示指定为元字符的打印和非打印字符组成，具体分为 5 类。

1. 普通字符作为原子：如 a~z、A~Z、0~9 等
2. 一些特殊字符和转义后元字符作为原子：
所有标点符号，但语句特殊意义的符号需要转义后才可作为原子，如：\” \‘ * \+ \? \. 等
3. 一些非打印字符作为原子：如：\f \n \r \t \v \cx
4. 使用“通用字符类型”作为原子：如：\d \D \w \W \s \S。
5. 自定义原子表([])作为原子：如：’/[apj]sp/’ ’/[^apj]sp/’

正则表达式中常用的非打印字符

原子字符	含义描述
\cx	匹配由 x 指明的控制字符。如\cM 匹配一个 Control-M 或回车符。x 的值必须为 A~Z 或 a~z 之一。
\f	匹配一个换页符。等价于 \x0c 或\cL
\n	匹配一个换行符。等价于 \x0a 或\cJ
\r	匹配一个回车符。等价于 \x0d 或\cM
\t	匹配一个制表符。等价于 \x09 或\cI
\v	匹配一个垂直制表符。等价于 \x0b 或\cK

正则表达式中常用的“通用字符类型”

原子字符	含义描述
\d	匹配任意一个十进制数字，等价于[0~9]
\D	匹配任意一个除十进制数字以外的字符，等价于[^0~9]
\s	匹配任意一个空白符，等价于[\f\n\r\t\v]
\S	匹配除空白符以外任何字符，等价于[^ \f\n\r\t\v]
\w	匹配任意一个数字、字母或下画线，等价于[0~9a-zA-Z_]
\W	匹配一个除数字、字母或下画线以外的任意一个字符，等价于[^0~9a-zA-Z_]

原子

原子是正则表达式的最基本的组成单元，而且在每个模式中最少要少包含一个原子。原子是由所有那些未显示指定为元字符的打印和非打印字符组成，具体分为 5 类。

1. 普通字符作为原子：如 a~z、A~Z、0~9 等
2. 一些特殊字符和转义后元字符作为原子：
所有标点符号，但语句特殊意义的符号需要转义后才可作为原子，如：\” \’ * \+ \? \. 等
3. 一些非打印字符作为原子：如：\f\n \r\t\v \cx
4. 使用“通用字符类型”作为原子：如：\d \D \w \W \s \S。
5. 自定义原子表([])作为原子：如：' /[apj]sp/' ' /[^apj]sp/'

单词边界限制

在使用各种编辑软件的查找功能时，可以通过选择“按单词查找”获得更准确的结果。正则表达式中也提供类似的功能。

例如：在字符串“This island is a beautiful land”中

元字符“\b”对单词的边界进行匹配；

\bis\b/ 匹配单词“is”，不匹配“This”和“island”。

\bis/ 匹配单词“is”和“island”中的“is”，不匹配“This”

元字符“\B”对除单词边界以外的部分进行匹配。

\Bis\B/ 将明确的指示不与单词的左、右边界匹配，只匹配单词的内部。所以在这个例子中没有结果。

\Bis/ 匹配单词“This”中的“is”

*元字符（特殊意义的字符）

元字符	含义描述
*	匹配 0 次、1 次或多次其前的原子
+	匹配 1 次或多次其前的原子

?	匹配 0 次或 1 次其前的原子
.	匹配除了换行符外的任意一个字符
	匹配两个或多个分支选择
{n}	表示其前面的原子恰好出现 n 次
{n, }	表示其前面的原子出现不小于 n 次
{n, m}	表示其前面的原子至少出现 n 次，最多出现 m 次
^或 \A	匹配输入字符串的开始位置（或在多行模式下行的开头，即紧随一个换行符之后）
\$或 \Z	匹配输入字符串的结束位置（或在多行模式下行的结尾，即紧随一个换行符之前）
\b	匹配单词的边界
\B	匹配除单词边界以外的部分
[]	匹配方括号中指定的任意一个原子
[^]	匹配除方括号中的原子以外的任意一个字符
()	匹配其整体为一个原子，即模式单元。可以理解为由多个单个原子组成的大原子

原子表 一方括号表达式

[] 一个中括号表示一个符号的匹配，内可以是列表

子表”[]”中存放一组原子，彼此地位平等，且仅匹配其中的一个原子。如果想匹配一个 ”a” 或 ”e” 使用 [ae]。

例如: Pr[ae]y 匹配 ”Pray” 或者 ”Prey”。

原子表 ”[^]” 或者称为排除原子表，匹配除表内原子外的任意一个字符。

例如: /p[^u]/匹配 “part” 中的 “pa”，但无法匹配 “computer” 中的 “pu” 因为 “u” 在匹配中被排除。依次对每个数据进行匹配

原子表 “[-]” 用于连接一组按 ASCII 码顺序排列的原子，简化书写。

例如: /x[0123456789]/可以写成 x[0-9],用来匹配一个由 “x” 字母与一个数字组成的字符串。

例如:

/[a-zA-Z]/匹配所有大小写字母

/^/[a-z][0-9]\$/匹配比如 “z2” 、 “t6” 、 “g7”

/0[xX][0-9a-fA-F]/匹配一个简单的十六进制数字，如 “0x9”。

/[^0-9a-zA-Z_]/匹配除英文字母、数字和下划线以外任何一个字符，其等价于 \W。

/0?[xX][0-9a-fA-F]+/匹配十六进制数字，可以匹配 “0x9B3C” 或者 “X800” 等。

/<[A-Za-z][A-Za-z0-9]*>/可以匹配 “<P>” 、 “<hl>” 或 “<Body>” 等 HTML 标签，并且不严格的控制大小写。

```

<?php
//正则表达式中的元字符:

//[] 一个中括号表示一个符号的匹配,里面可以是列表
/*例如: [abc]匹配a或b或c的任意一个字符
   [a-z]表示从a到z的一个字符
   [0-9]表示0到9的任意一位数值
   [0-9][0-9] 表示两位数值。
   [a-zA-Z][A-Za-z]表示两位的大小字母
   [aoeiu]表示一个元音字母
*/
if(preg_match("/[^0-9]/","123456789")){
    echo "存在非数字字符";
}else{
    echo "纯数字";
}

```

preg_match 第三的参数表示匹配出来的结果

(1) 重复匹配

元字符“{}”准确地指定原子重复的次数，指定所匹配的原子出现的次数。

“{m}” 表示其前原子恰好出现 m 次。

“{m, n}” 表示其前原子至少出现 m 次，至多出现 n 次。

“{m, }” 表示其前原子出现不少于 m 次。

/zo{1,3}m/ 只能匹配字符串“zom”、“zoom”、或“zooom”。

/zo{3}m/ 只能匹配字符串“zooom”。

/zo{3, }m/ 可以匹配以“z”开头，“m”结束，中间至少为 3 个“o”的字符串。

/bo{0,1}u/ 可以匹配字符串“bought a butter” 中的“bou”和“bu”，等价于 bo?u。

```

$str = "zhangsan345346ffff";
if(preg_match("/[0-9]{3,9}/",$str,$res)){
    echo 'ok';
}else{
    echo 'no';
}
print_r($res);

```

okArray ([0] => 345346) (匹配满足即可)

(2) 重复匹配

正则表达式中有一些用于重复匹配某些原子的元字符：“?”、“*”、“+”。他们主要的区别是重复匹配的次数不同。

元字符“?”：表示 0 次或 1 次匹配紧接在其前的原子。

例如：/colou?r/匹配“colour”或“color”。

元字符“*”：表示 0 次、1 次或多次匹配紧接在其前的原子。

例如：/zo*/可以匹配 z、zoo

元字符“+”：表示 1 次或多次匹配紧接在其前的原子。

例如：/go+gle/匹配“google”、“google”或“gooogle”等中间含有多个 o 的字符串。

字符串边界限制

在某些情况下，需要对匹配范围进行限定，以获得更准确的匹配结果。“^”和“\$”分别指定字符串的开始和结束。

例如，在字符串“Tom and Jerry chased each other in the house until tom’s uncl come in”中元字符“^”或“\A”置于字符串的开始确保模式匹配出现在字符串首端；

/^Tom/

元字符“\$”或“\Z”置于字符串的结束，确保模式匹配出现字符串尾端。

/in\$/

如果不加边界限制元字符，将获得更多的匹配结果。

/^Tom\$/精确匹配 /Tom/模糊匹配

=====

^ 有两层的特殊意义：

在中括号里面表示取反

在正则表达式里面表示字符开头

=====

```
//2. 精确式匹配
if(preg_match("/^[0-9]{2}/","789zhangsan",$res)){ //以2位数字开头
    echo "yes";
} else{
    echo "no";
}

print_r($res);
if(preg_match("/^0-9{2}$/","78",$res)){ //绝对的2位数字
    echo "yes";
} else{
    echo "no";
}
print_r($res);
```

单词边界限制

在使用各种编辑软件的查找功能时，可以通过选择“按单词查找”获得更准确的结果。正则表达式中也提供类似的功能。

例如：在字符串“This island is a beautiful land”中

元字符“\b”对单词的边界进行匹配；

\bis\b/ 匹配单词“is”，不匹配“This”和“island”。

\bis/ 匹配单词“is”和“island”中的“is”，不匹配“This”

元字符“\B”对除单词边界以外的部分进行匹配。

\Bis\B/ 将明确的指示不与单词的左、右边界匹配，只匹配单词的内部。所以在这个例子中没有结果。

\Bis/ 匹配单词“This”中的“is”

任何一个字符

元字符“.”匹配除换行符外任何一个字符。

相当于: [^\n](Unix 系统)或[^r\n](windows 系统)。

例如: /pr.y/可以匹配的字符串“prey”、“pray”或“pr%y”等。

通常可以使用“.*”组合来匹配除换行符外的任何字符。在一些书籍中也称其为“全匹配符”或“单含匹配符”。

例如:

/^a.*z\$/ 表示可以匹配字母“a”开头,字母“z”结束的任意不包括换行符的字符串。

/.+/ 也可以完成类似的匹配功能所不同的是其至少匹配一个字符。

/^a.+z\$/ 匹配“a%z”不匹配字符串“az”。

模式单元

元字符“()”将其中的正则表达式变为原子(或称模式单元)使用。与数学表达式中的括号类似,“()”可以做一个单元被单独使用。

例如:

/(\Dog)+/匹配的“Dog”、“DogDog”、“DogDogDog”,因为紧接着“+”前的原子是元字符“()”括起来的字符串“Dog”。

/You (very)+ old/匹配“You very old”、“You very very old”

/Hello (world|earth)/匹配“Hello world”、“Hello earth”

一个模式单元中的表达式将被优先匹配或运算。

实例: 小括号作用

```
//preg_grep -- 返回与模式匹配的数组单元
$a = array(
    'zhangTao@lampbrother.net',
    'tao.zhang.tc@163.com',
    'zhangsan.sohu.com',
    'qwe@rty.u');
//匹配出上面数组中有效的Email地址
$list = preg_grep("/^[\a-zA-Z_\.\\-]+@[\\a-zA-Z\\-]+([\\a-zA-Z]+)[1,2]$"/i", $a);
print_r($list);
//Array ( [0] => zhangTao@lampbrother.net [1] => tao.zhang.tc@163.com [3] => qwe@rty.u )
```

重新使用的模式单元

系统自动将模式单元“()”中的匹配依次存储起来,在需要时可以用“\1”、“\2”、“\3”的形式进行引用。当正则表达式包含有相同的模式单元时,这种方法非常便于对其进行管理。注意使用时需要写成“\\1”、“\\2”

例如:

/^\d{2}([\W])\d{2}\\\1\d{4}\$/匹配“12-31-2006”、“09/27/1996”、“86 01 4321”等字符串。

但上述正则表达式不匹配“12/34-5678”的格式。这是因为模式“[\W]”的结果“/”已经被存储。下个位置“\1”引用时,其匹配模式也是字符“/”。

当不需要存储匹配结果时使用非存储模式单元“(?:)”

例如/(?:a|b|c)(D|E|F)\\1g/ 将匹配“aEEg”。在一些正则表达式中,使用非存储模式单元是必要的。否则,需要改变其后引用的顺序。上例还可以写成/(a|b|c) (C|E|F)\\2g/。

```
<?php
//重用单元模式
$date="08/26/2003"; //2003/08/26
|
//echo preg_replace('/([0-9]{2})/\\1/([0-9]{2})/\\1/([0-9]{4})/\\1/','\\1\\2\\3', $date);
|
echo preg_replace('/([0-9]{2})/\\1/([0-9]{2})/\\1/([0-9]{4})/\\1/','\\1\\2\\3', $date);
```

实例：重用模式单元

```
//将下面数组中有效日期全部匹配出来 (小括号有子存储功能)
$a = array("2013-08-22","2012/12-12","2014-09-20","2011-02/20",'2013/06/25');
$list = preg_grep("/\d{4}[\-\ \ ]\d{2}[\-\ \ ]\d{2}/",$a); //会出现分隔符不统一
//Array ([0] => 2013-08-22 [1] => 2012/12-12 [2] => 2014-09-20 [3] => 2011-02/20 [4] => 2013/06/25 )

$list = preg_grep("/\d{4}([\-\ \ ])\d{2}\1\d{2}/",$a); //\1 获取第一个子存储内容使保持一致
print_r($list); //Array ([0] => 2013-08-22 [2] => 2014-09-20 [4] => 2013/06/25 )
//此处正则中的\1表示重复前面第一个括号的内容 (要求一样)。 \1中\需要转义
```

模式选择符

元字符“|”又称模式选择符。在正则表达式中匹配两个或更多的选择之一。

例如：

在字符串“*There are many apples and pears.*”中，/apple|pear/在第一次运行时匹配“apple”；再次运行时匹配“pear”。也可以继续增加选项，如：/apple|pear|banana|lemon/

实例：证明以下两个知识点

取消贪婪模式 .+?

匹配可有可无属性 .*?

```
<?php
```

```
//$str='<ul><li></li></ul>';
//1.匹配所有图片链接
//if(preg_match('/

//2. 获取所有图片链接即将正则模式用小括号括起来,()表示存储子单元的作用
//if(preg_match("/<img src=(.+)\>/", $str, $res)){// [0] =>  [1] => ./aa.jpg

//$str= '<ul><li></li><hr size="2"/></ul>';// 最大匹配(贪婪模式)
//if(preg_match("/<img src=.+\>/", $str, $res)){// [0] => </li><hr size="2"/>

//3. (.+?)取消贪婪模式
//if(preg_match('//', $str, $res)){ // [0] =>  [1] => ./aa.jpg

//4. .*? 可有可无 [0] =>  [1] => ./aa.jpg
//$str= '<ul><li></li><hr size="2"/></ul>';
if(preg_match('/<img src=(.+?).*\>/', $str, $res)){
    echo 'ok';
} else{
    echo 'no';
}

print_r($res);
```

实例：获取指定网站的超链接（理解子存储，preg_match_all函数的使用）

```
<?php
```

//声明一个可以匹配 URL 的正则表达式

\$pattern = '(https?|ftps?):\/\/(www|bbs)\.(^\.\w+).(com|net|org)(\/[\w-\.\w?\%\&]=)*?/i';

//声明一个包含多个 URL 链接地址的多行文字

\$subject = "网址为 http://bbs.lampbrother.net/index.php 的位置是 LAMP 兄弟连，

网址为 http://www.baidu.com/index.php 的位置是百度，

网址为 http://www.google.com/index.php 的位置是谷歌。";

\$i = 1; //定义一个计数器，用来统计搜索到的结果数

```

//搜索全部的结果
if(preg_match_all($pattern, $subject, $matches, PREG_SET_ORDER)) {
    foreach($matches as $urls) { //循环遍历二维数组$matches
        echo "搜索到第".$i."个 URL 为: ".$urls[0]."<br>";
        echo "第".$i."个 URL 中的协议为: ".$urls[1]."<br>";
        echo "第".$i."个 URL 中的主机为: ".$urls[2]."<br>";
        echo "第".$i."个 URL 中的域名为: ".$urls[3]."<br>";
        echo "第".$i."个 URL 中的顶级域为: ".$urls[4]."<br>";
        echo "第".$i."个 URL 中的文件为: ".$urls[5]."<br>";
        $i++; //计数器累加
    }
} else {
    echo "搜索失败! ";
}

<?php
//获取http://www.baidu.com/网上的所有超级链接信息

//读取网站内容
//$content = file_get_contents("http://www.baidu.com/");
//echo $content;
$content = file_get_contents("baidu.html");
//匹配解析数据
preg_match_all("/<a href=\"?(.*?)\"?>(.*?)</a>/", $content, $res);

//遍历结果输出
echo "<table width='700' border='1'>";
echo "<tr><th>链接地址</th><th>URL</th><th>名称</th></tr>";
foreach($res[0] as $k=>$v){
    echo "<tr>";
    echo "<td>{$v}</td>";
    echo "<td>{$res[1][$k]}</td>";
    echo "<td>{$res[2][$k]}</td>";
    echo "</tr>";
}
echo "</table>";

<?php
/**
 * 用于获取 URL 中的文件名部分
 * @param string $url      任何一个 URL 格式的字符串
 * @return string          URL 中的文件名称部分
 */
function getFileName($url) {
    //获取 URL 字符串中最后一个 "/" 出现的位置，再加 1 则为文件名开始的位置
    $location = strpos($url, "/") + 1;
}

```

```

//获取在 URL 中从$location 位置取到结尾的子字符串
$fileName = substr($url, $location);
//返回获取到的文件名称
return $fileName;
}

//获取网页文件名 index.php
echo getFileName("http://bbs.lampbrother.net/index.php");
//获取网页中图片名 logo.gif
echo getFileName("http://bbs.lampbrother.com/images/Sharp/logo.gif");
//获取本地中的文件名 php.ini
echo getFileName("file:///C:/WINDOWS/php.ini");

<?php
$str="[2013-08-28]";
//将上面的日期年月日都匹配出来
preg_match_all("/\d+/", $str, $a);    第一种方式
print_r($a);
//Array ( [0] => Array ( [0] => 2013 [1] => 08 [2] => 28 ) )

echo "<hr/>";

//匹配出来的整体单元若需要存储子单元就使用括号() 第二种方式
preg_match("/(\d{4})-(\d{2})-(?:\d{2})/", $str, $a); //?:表示拒绝子存储
print_r($a);
//Array ( [0] => 2013-08-28 [1] => 2013 [2] => 08 )

echo "<hr/>";
//将下面字符串中的HOST和localhost解析出来 认识小括号作用
$s = 'define("HOST","localhost");define("USER","root");define("PASS","123456");';
preg_match_all("/define\((\".*?\")\,\\".*?\")\/", $s, $a);
print_r($a);

Array ( [0] => Array ( [0] => 2013 [1] => 08 [2] => 28 ) )
Array ( [0] => 2013-08-28 [1] => 2013 [2] => 08 )


---


Array
(
    [0] => Array
    (
        [0] => define("HOST", "localhost")
        [1] => define("USER", "root")
        [2] => define("PASS", "123456")
    )

    [1] => Array
    (
        [0] => HOST
        [1] => USER
        [2] => PASS
    )

    [2] => Array
    (
        [0] => localhost
        [1] => root
        [2] => 123456
    )
)

```

```

<?php
$pattern = "/(\d{2})\|(\d{2})\|(\d{4})/"; //日期格式的正则表达式

$text="今年国庆节放假日期为 10/01/2012 到 10/07/2012 共 7 天。";//带有两个日期格式的字符串

echo preg_replace($pattern, "\3-\1-\2", $text); //将日期替换为以“-”分隔的格式

echo preg_replace($pattern, "\$3-\$1-\$2", $text); //将 “\1” 改为 “\$1” 的形式

```

```

<?php
//可以匹配所有 HTML 标记的开始和结束的正则表达式
$pattern = "/(<?)(w+)([>]*>)/e";

```

```

//声明一个带有多个 HTML 标记的文本
$text = "这个文本中有<b>粗体</b>和<u>带有下画线</u>以及<i>斜体</i>还有<font color='red' size='7'>带有颜色和字体大小</font>的标记";

```

//将所有 HTML 的小写标记替换为大写

```
echo preg_replace($pattern, "\1".strtoupper("\2")."\3", $text);
```

子字符串的匹配与查找函数

1. 函数 **preg_match()** --执行一个正则表达式匹配

```
int preg_match(string $pattern, string $subject,[array &$matches])
搜索 subject 与 pattern 给定的正则表达式的一个匹配.
```

2. 函数 **preg_match_all()** --执行全局正则表达式匹配（必须是三个参数）

```
int preg_match_all(string $pattern ,string $subject ,array &$matches [, int $flags])
搜索 subject 中所有匹配 pattern 给定正则表达式 的匹配结果并且将它们以 flag 指定顺序输出到 matches 中. 参数 flags 是指定 matches 的数组格式。
```

3. 函数 **preg_grep()** --返回匹配模式的数组条目 返回符合模式的数组单元（是 preg_filter 函数的删除版，因为只匹配没有替换）

```
array preg_grep ( string $pattern , array $input [, int $flags = 0 ] )
返回给定数组 input 中与模式 pattern 匹配的元素组成的数组。
```

4. 其它子串处理函数： **strstr()**、**strpos()**、**strrpos()**、**substr()**

子串替换函数 查找并替换函数

1. **preg_replace** — 执行一个正则表达式的搜索和替换

```
mixed preg_replace ( mixed $pattern , mixed $replacement,mixed $subject [,int $limit = -1])
搜索 subject 中匹配 pattern 的部分，以 replacement 进行替换.
```

2. **str_replace** — 子字符串替换

```
mixed str_replace ( mixed $search , mixed $replace , mixed $subject [, int &$count ] )
```

该函数返回一个字符串或者数组。该字符串或数组是将 subject 中全部的 search 都被 replace 替换之后的结果。通过\$count 参数指定来获取替换的次数。

1.

```
$pattern='/[0-9]/';
$subject='Reg235ula46rExpres78s';
$replacement = 'OK';

$int = preg_replace($pattern,$replacement,$subject);
$int1 = preg_filter($pattern,$replacement,$subject);
```

2.

```
$pattern= array('/[234]/' , '/[56]/' , '/[78]/');
$subject='Reg235ula46rExpres78s';
$replacement = array('AA' , 'BB' , 'CC');

$int = preg_replace($pattern,$replacement,$subject);
$int1 = preg_filter($pattern,$replacement,$subject);
```

3、匹配不到忽略

```
$pattern= array('/[234]/' , '/[56]/' , '/[78]/');
$subject=array('Reg23','5ula46r','Expr','es78s');
$replacement = array('AA' , 'BB' , 'CC');

//目标是数组形式，返回数组（两个函数返回结果有区别之处）
$int = preg_replace($pattern,$replacement,$subject);
$int1 = preg_filter($pattern,$replacement,$subject);
```

```

string 'RegOKOKu1OKOkrExpresOKOKs' (length=28)    string 'RegAAAAABBu1AABBExpresCCCCs' (length=28)
string 'RegOKOKu1OKOkrExpresOKOKs' (length=28)    string 'RegAAAAABBu1AABBExpresCCCCs' (length=28)

Array
(
    [0] => RegAAAA
    [1] => BBu1AABB
    [2] => Expr
    [3] => esCCCCs
)

Array
(
    [0] => RegAAAA
    [1] => BBu1AABB
    [2] => esCCCCs
)

<?php
//preg_replace -- 执行正则表达式的搜索和替换

//将下面字符串中的分割符都换成“,”;
$s = "21;3:4#23!45;34:54:58;66|5#87";

//echo str_replace(array(";",":","#","!"),",",$s); //使用字符串替换(不灵活)
echo preg_replace("/\D/",",",$s); //使用正则替换

echo "<hr/>";

//将下面字符串中的b标签换成i标签
$str = "<b>aaa</b><b style='color:red'>ccc</b><b>bbb</b><b>ddd</b>";
                                         \\1重复使用前面的单元(.*)?
//echo preg_replace("/<b(.*)>(.*)</b>/","<i\\1>\\2</i>",$str);
//小括号的作用子存储作用,$n等效\n重复使用单元模式,双引号中$有特殊意义需要转义
echo preg_replace("/<b(.*)>(.*)</b>/","<i\\$1>\\$2</i>",$str);
echo preg_replace("/<b(.*)>(.*)</b>/",'<i\\$1>$2</i>',$str,-1,$m); // -1换所有,$m计数用
echo "<br/>成功替换{$m}个";

//其中\\1或$1表示重复前面正则中的第一小括号内容, 以至类推\\2或$2表示第二小括号

```

字符串的分割和连接

`preg_split` — 通过一个正则表达式分隔字符串

```
array preg_split ( string $pattern , string $subject [, int $limit = -1 [, int $flags = 0 ]] )
```

通过一个正则表达式\$pattern 分隔给定字符串\$subject。其中\$limit 是最大替换个数。

flags 可以是任何下面标记的组合

PREG_SPLIT_NO_EMPTY: 返回分隔后的非空部分

PREG_SPLIT_DELIM_CAPTURE: 用于分隔的模式中的括号表达式将被捕获并返回.

PREG_SPLIT_OFFSET_CAPTURE: 返回附加字符串偏移量

`explode` -- 使用一个字符串分割另一个字符串

```
array explode ( string $separator , string $string [, int $limit ] )
```

此函数返回由字符串组成的数组，每个元素都是 string 的一个子串，它们被字符串 separator 作为边界点分割出来。 其中\$limit 是指定对大分割个数。

`implode` -- 使用一个子串组装一个数组。

```
string implode ( string $glue , array $pieces )
```

将 pieces 数组中的每个值，使用 glue 作为分隔符组装成一个子串并返回。

```
<?php
// *preg_split -- 用正则表达式分割字符串
```

//将下面字符串中的分成数据数组;

```
$s = "21;3:4#23!45;34:54:58;66|5#87";
```

```
//$a = preg_split("/\D/",$s,8); //最大拆分出8个
```

```
//$a = preg_split("/\D/",$s,-1); //-1表示默认值, 能拆分成多少是多少
```

```
$a = preg_split("/\D/",$s); //默认能拆分成多少是多少
```

```
print_r($a);
```

```
$pattern= '/[247]/';
$subject='Reg235ula46rExpres78s';
$int1 = preg_split($pattern,$subject);
```

```
$str = 'piece1,piece2,piece3,piece4,piece5';
$int = explode(',',$str);
show($int1);
echo '<hr />';
show($int);
```

```
Array
(
    [0] => Reg
    [1] => 35ula
    [2] => 6rExpres
    [3] => 78s
)
```

```
Array
(
    [0] => piece1
    [1] => piece2
    [2] => piece3
    [3] => piece4
    [4] => piece5
)
```

模式修正符

修正符	含义描述
i	在和模式进行匹配时不区分大小写
m	将字符串视为多行。默认的正则开始“^”和结束“\$”将目标字符串作为单一的一“行”字符。加上m后，那么开始和结束将会指字符串的每一行。
s	如果设定了此修正符，模式中的圆点元字符“.”匹配所有的字符，包括换行符。即：将字符串视为单行，换行符作为普通字符看待
x	模式中的空白忽略不计，除非它已经被转义
e	只用在preg_replace()函数中，在替换字符串中对逆向引用做正常的替换，将其作为PHP代码求值，并用其结果来替换所搜索的字符串。
U	本修正符反转了匹配数量的值使其不是默认的重复，而变成在后面跟上“?”才变得重复。这和Perl不兼容。也可以通过在模式之中设定(U)

```

<?php
//修正符的使用
var_dump(preg_match("/^[a-z]+$/i","qwerAGFGqwer")); //其中i修正符表示不区分大小写

$s=<li>AAA
</li>;
preg_match("/<li>.*?</li>/s",$s,$a); //s修正符是让.点可以支持换行符的匹配
print_r($a);

echo '<hr/>';

preg_match("/<li>(.*)</li>/s",$s,$a);
print_r($a);

echo "<hr/>";

$s=<li>aaa</li><li>bbb</li>";
preg_match("/<li>.*</li>/",$s,$a);
print_r($a);

echo '<hr/>';

preg_match("/<li>.*</li>/U",$s,$a); //U修正符拒绝所有贪婪匹配(不常用, 使用.*?)
print_r($a);

```

<pre>int 1 Array ([0] => • AAA) </pre>	<pre>Array ([0] => • AAA [1] => AAA)) </pre>	<pre>Array ([0] => • aaa • bbb) </pre>
---	---	---

重要实例：修改配置文件

```
<?php
//数据库连接配置信息

define("HOST","127.0.0.1"); //数据库主机名
define("USER","admin"); //账号
define("PASS","666"); //密码
define("DBNAME","lamp71"); //数据库名

edit.php
<form action="doEdit.php" method="post">
    <table width="240" border="0">
        <?php
        //定义字段名对应说明信息
        $typelist = array("HOST"=>"主机名","USER"=>"账号","PASS"=>"密码","DBNAME"=>"数据库名");
        //读取配置文件信息，并输出到当前表单中

        //1. 读取配置文件信息
        $content = file_get_contents("config.php");

        //2. 使用正则进行数据解析
        preg_match_all("/define\((\".*?\")\,(\".*?\")\)\/", $content, $a);
        //var_dump($a);

        //3. 遍历解析的结果，并输出
        foreach($a[1] as $k=>$v){
            echo "<tr>\n";
            echo "\t<td>{$typelist[$v]}</td>\n";
            echo "\t<td><input type=\"text\" name=\"$v\" value=\"$a[2][$k]\" /></td>\n";
            echo "</tr>\n";
        }
    ?>
    <tr>
        <td colspan="2" align="center">
            <input type="submit" value="修改"/>&ampnbsp&ampnbsp;
            <input type="reset" value="重置"/>
        </td>
    </tr>
</table>
</form>

doedit.php
<?php
//执行配置文件信息的修改

//1. 读取配置文件信息
```

```
$content = file_get_contents("configure.php");

//2. 遍历 POST 中的信息，并做正则替换
foreach($_POST as $k=>$v){
    $content = preg_replace("/define\(\"{$k}\",\".*?\")/", "define(\"{$k}\",\"{$v}\")", $content);
}

//3. 再写回源文件
file_put_contents("config.php", $content);

echo "ok";
```

===== php--正则表达式 =====

一、正则表达式的介绍：

正则表达式是用于描述字符排列和匹配模式的一种语法规则。

它主要用于字符串的模式分割、匹配、查找及替换操作。

1. 用途：匹配、查找、替换、分割
2. PHP 提供了两套正则表达式函数库
 - *1. Perl 兼容正则表达式函数（推荐使用）
 - 2. POSIX 扩展正则表达式函数

二、语法：

1. 表达式的格式：“/表达式/[修正符]”

解释：其中“/”表示正则表达式的定界符，但是也可以是其他符号：如“#”，“！”“

注意：定界符不可以是字母、数字和斜线\。

像“#”、“|”、“!”等都可以的

如：/.../ #...# |....|

其中修正符是可选的，表示对表达式做额外的修饰。

三、正则表达式的组成部分：

1. 原子是组成正则表达式的基本单位，在分析正则表达式时，应作为一个整体。

原子包括以下内容：

> 单个字符、数字，如 a-z, A-Z, 0-9。

- > 模式单元，如 (ABC) 可以理解为由多个原子组成的大原子。
- > 原子表，如 [ABC]。
- > 重新使用的模式单元，如: \1
- > 普通转义字符，如: \d, \D, \w
- > 转义元字符，如: *, \.
- *> 元字符

*2. 元字符（具有特殊意义字符）：

[] 表示单个字符的原子表

例如: [aoeiu] 表示任意一个元音字母

[0-9] 表示任意一位数字

[a-z][0-9] 表示小写字母和一位数字构成的两位字符

[a-zA-Z0-9] 表示任意一位大小字母或数字

[^] 表示除中括号内原子之外的任何字符 是[]的取反

例如: [^0-9] 表示任意一位非数字字符

[^a-z] 表示任意一位非小写字母

{m} 表示对前面原子的数量控制，表示是 m 次

例如: [0-9]{4} 表示 4 为数字

[1][3-8][0-9]{9} 手机号码

{m,} 表示对前面原子的数量控制，表示是至少 m 次

例如: [0-9]{2,} 表示两位及以上的数字

{m,n} 表示对前面原子的数量控制，表示是 m 到 n 次

例如: [a-z]{6,8} 表示 6 到 8 位的小写字母

* 表示对前面原子的数量控制，表示是任意次，等价于{0,}

+ 表示对前面原子的数量控制，表示至少 1 次，等价于{1,}

? 表示对前面原子的数量控制，表示 0 次或 1 次（可有可无） 等价于{0,1}

例如：正整数: [1-9][0-9]*

整数: [-]?[0-9]+

email:

) 表示一个整体原子，【还有一个子存储单元的作用】。

也可以使用?: 来拒绝子存储。 (?::*)?

例如: (red) 字串 red

(red|blue) 字串 red 或 blue

(abc){2} 表示两个 abc

| 表示或的意思

(red|blue) 字串 red 或 blue

^ 用在正则单元块的开头处，表示必须以指定的开头

\$ 用在正则单元块的结尾处，表示必须以指定的结尾

. 表示任意一个除换行符之外的字符
常用组合: .*? 或 .+? 表示最小匹配所有字符 (拒绝贪婪匹配)

3. 普通转义字符:

*\d	匹配一个数字; 等价于[0-9]
*\D	匹配除数字以外任何一个字符; 等价于[^0-9]
*\w	匹配一个英文字母、数字或下划线; 等价于[0-9a-zA-Z_]
*\W	匹配除英文字母、数字和下划线以外任何一个字符; 等价于[^0-9a-zA-Z_]
*\s	匹配一个空白字符; 等价于[\f\n\r\t\v]
*\S	匹配除空白字符以外任何一个字符; 等价于[^{\f\n\r\t\v}]
\f	匹配一个换页符等价于 \x0c 或 \cL
*\n	匹配一个换行符; 等价于 \x0a 或 \cJ
*\r	匹配一个回车符等价于\x0d 或 \cM
*\t	匹配一个制表符; 等价于 \x09\或\cl
\v	匹配一个垂直制表符; 等价于\x0b 或\ck
\oNN	匹配一个八进制数字
\xNN	匹配一个十六进制数字
\cC	匹配一个控制字符

/^-?\d+\$|^-?0[xX][\da-fA-F]+\$/ 表示十进制和十六进制的一个数字

^-?\d+\$ ^-?0[xX][\da-fA-F]+\$

//表示一个邮箱地址
/^[\w.-]+@[^\w.-]+\.\w{2,3}\$/,

4. 模式修整符

i 表示不区分大小写;

"/[a-zA-Z]"/" <==> "[a-z]/i"

s 表示匹配视为单行 (就是可以让点.支持换行)

U 表示拒绝贪婪匹配

四、 正则表达式的函数:

preg_grep -- 返回与模式匹配的数组单元
* preg_match_all -- 进行全局正则表达式匹配 , 返回共计匹配的个数。

和下面的一样，不同的是匹配到最后（全局匹配）

* preg_match -- 进行正则表达式匹配，只匹配一次，返回 1，否则 0，

格式： preg_match(" 正则表达式 ", " 被匹配的字符串 ", 存放结果的变量名, PREG_OFFSET_CAPTURE, 起始偏移量)

其中： PREG_OFFSET_CAPTURE 表示获取匹配索引位置

起始偏移量：从指定位置开始匹配

preg_quote -- 转义正则表达式字符

preg_replace_callback -- 用回调函数执行正则表达式的搜索和替换

*preg_replace -- 执行正则表达式的搜索和替换

*preg_split -- 用正则表达式分割字符串

八、PHP 的错误处理

1 错误报告级别

2 调整错误报告级别

3 使用 trigger_error() 函数来替代 die()

4 自定义错误处理

5 写错误日志

错误报告级别

PHP 程序的错误发生一般归属于下列三个领域：

语法错误：

是在程序执行之前的语法校验中发现的错误，如代码中缺少分号，花括号，小括号。

运行时错误：

就是程序在执行过程中发生的错误，如调用一个不存在的函数，或缺少参数等等

逻辑错误：

这种错误最麻烦，既不阻止脚本执行，也不输出错误消息。如死循环，条件判断

异常：

一个异常（后面面向对象时会讲）则是在一个程序执行过程中出现的一个例外，或是一个事件，它中断了正常指令的运行，跳转到其他程序模块继续执行。

PHP 的错误报错级别

级别常量	错误值	错误报告描述
E_ERROR	1	致命的运行时错误（阻止脚本执行）
E_WARNING	2	运行时警告（非致命性错误）
E_PARSE	4	从语法中解析错误
E_NOTICE	8	运行时注意消息（可能是或可能不是一个问题）
E_CORE_ERROR	16	PHP 启动时初始化过程中的致命错误

E_CORE_WARNING	32	PHP 启动时初始化过程中的警告(非致命性错)
E_COMPILE_ERROR	64	编译时致命性错
E_COMPILE_WARNING	128	编译时警告(非致命性错)
E_USER_ERROR	256	用户自定义的致命错误
E_USER_WARNING	512	用户自定义的警告(非致命性错误)
E_USER_NOTICE	1024	用户自定义的提醒(经常是 bug)
E_STRICT	2048	编码标准化警告

调整错误报告级别

display_errors: 是否开启 PHP 输出错误报告的功能

值为: On (默认输出错误报告) 、 Off (屏蔽所有错误信息)

在 PHP 脚本中可调用 `ini_set()` 函数, 动态设置 `php.ini` 配置文件.

如: `ini_set("display_errors","On"); //显示所有错误信息`

error_reporting: 设置不同的错误报告级别。

`error_reporting = E_ALL & ~E_NOTICE`

-- 可以抛出任何非注意的错误, 默认值

`error_reporting = E_ERROR | E_PARSE | E_CORE_ERROR`

-- 只考虑致命的运行时错误、新解析错误和核心错误。

`error_reporting = E_ALL & ~(E_USER_ERROR | E_USER_WARNING | E_USER_NOTICE)`

-- 报告除用户导致的错误之外的所有错误。

在 PHP 脚本可以通过 `error_reporting()` 函数动态设置错误报告级别。如: `error_reporting(E_ALL);`

确定 PHP 错误报告行为的配置指令

配置指令	默认值	描述
<code>display_startup_errors</code>	<code>Off</code>	是否显示 PHP 引擎在初始化时遇到的错误
<code>log_errors</code>	<code>Off</code>	确定日志语句记录位置
<code>error_log</code>	<code>Null</code>	设置错误可以发送到 syslog 中
<code>log_errors_max_len</code>	<code>1024</code>	每个日志项的最大长度, 以字节为单位, 设置 0 表示指定最大长度。
<code>ignore_repeated_errors</code>	<code>Off</code>	是否忽略同一个文件、同一行发生的重复错误消息
<code>ignore_repeated_source</code>	<code>Off</code>	忽略不同文件中和同一文件中不同行发生的重复错误。
<code>track_errors</code>	<code>Off</code>	启动该指令会使 PHP 在 <code>\$php_errormsg</code> 中存储最近发生的错误信息。

使用 trigger_error() 函数来替代 die()

首先函数 die() 等同于 exit()，两者如果执行都会终止 PHP 程序，而且可以在退出程序之前输出一些错误报告。trigger_error() 则可以生成一个用户警告来代替，使程序更具有灵活性。

例如，trigger_error（“没有找到文件”，E_USER_ERROR）。使用 trigger_error() 函数来替代 die()，你的代码在处理错误上会更具优势，对于客户程序员来说更容易处理错误。

```
function demo($a,$b){  
    if($b==0){  
        //die("除数不可以为零");  
        trigger_error("除数不可以为零！",E_USER_ERROR);  
    }  
    echo $a/$b;  
}  
demo(10,0);
```

自定义错误处理

自定义错误报告的处理方式，可以完全绕过标准的 PHP 错误处理函数，这样就可以按自己定义的格式打印错误报告，或改变错误报告打印的位置，以下几种情况可以考虑自定义错误处理。

可以记下错误的信息，及时发现一些生产环境出现的问题

可以屏蔽错误。

可以控制错误的输出。

可以作为调试工具。

使用 set_error_handler() 函数来设置用户自定义错误处理

自定义错误处理

```
<?php  
error_reporting(0); //屏蔽程序中的错误  
/**  
 * 定义 Error_Handler 函数，作为 set_error_handler() 函数的第一个参数"回调"  
 * @param int      $error_level      错误级别  
 * @param string   $error_message    错误信息  
 * @param string   $file            错误所在文件  
 * @param int      $line            错误所在行数  
 */  
function error_handler($error_level, $error_message, $file, $line) {  
    $EXIT = FALSE;  
    switch( $error_level ) {  
        //提醒级别  
        case E_NOTICE:  
        case E_USER_NOTICE:  
            $error_type = 'Notice'; break;  
  
        //警告级别  
        case E_WARNING:  
        case E_USER_WARNING:
```

```

$error_type = '警告(Warning)'; break;

//错误级别
case E_ERROR:
case E_USER_ERROR:
    $error_type = 'Fatal Error';
    $EXIT = TRUE; break;

//其他未知错误
default:
    $error_type = 'Unknown';
    $EXIT = TRUE; break;
}

//直接打印错误信息， 也可以写文件， 写数据库， 反正错误信息都在这， 任你发落
printf ("<font color='#FF0000'><b>%s</b></font>: %s in <b>%s</b> on line <b>%d</b><br>\n",
$error_type, $error_message, $file, $line);

//如果错误影响到程序的正常执行， 跳转到友好的错误提示页面
if(TRUE == $EXIT) {
    echo '<script>location = "err.html"; </script>';
}
}

//这个才是关键点， 把错误的处理交给 error_handler()
set_error_handler('error_handler');

//测试代码
//使用未定义的变量要报 notice 的
echo $novar;

//除以 0 要报警告的
echo 3/0;

//自定义一个错误
trigger_error('Trigger a fatal error', E_USER_ERROR);

```

*写错误日志

两种方式记录错误日志：

- 》 使用指定的文件记录错误报告日志

- 》 错误日志记录到操作系统的日志里

推荐使用方式 第一种方式

```
<?php
//1. 日志记录: (采用文件方式记录日志)
/*
1、配置: 在 php.ini 配置文件中配置如下信息
    error_reporting = E_ALL //将向 PHP 发送每个错误
    display_errors=Off //不显示错误报告
    *log_errors=On //决定日志语句记录的位置。
    log_errors_max_log=1024 // 每个日志项的最大长度
    *error_log=D:/myerror.log //指定错误写进的文件 (可以以日期为文件名)

2、使用函数: 在 php 文件中使用 error_log() 来记录日志, 就可以将
    信息写入到 myerror.log 文件中
    如: error_log("登录失败了!"); //人为的记录错误信息
    注意: 当前 php 程序报错时, 信息也会自动写入到 myerror.log
*/
//测试代码
//abc();

//error_log("出错了!"); //使用函数自己输出错误信息

trigger_error('Trigger a fatal error', E_USER_ERROR);
```

第二种方式

```
<?php
//2. 日志记录: (采用系统文件方式记录日志)
/*
1、先配置 PHP.ini 文件中
    error_reporting = E_ALL //将向 PHP 发送每个错误
    * display_errors=Off //不显示错误报告
    * log_errors=On //决定日志语句记录的位置。
    log_errors_max_log=1024 // 每个日志项的最大长度
    * error_log=syslog //指定到系统日志中。

2、使用四个函数来记录日志:
    define_syslog_variables(); //为系统日志初始化配置
    openlog(); //打开一个日志链接
    syslog(); //发送一条日志记录
    closelog(); //关闭日志链接
*/

define_syslog_variables(); //为系统日志初始化配置
openlog("php5",LOG_PID,LOG_USER); //打开一个日志链接
//发送一条日志记录
syslog(LOG_WARNING, "警告报告向 syslog 中发送的演示, 警告时间: ".date("Y/m/d H:i:s"));
closelog(); //关闭日志链接
```

九、PHP 的日期和时间

- 1 UNIX 时间戳
- 2 在 PHP 中获取日期和时间
- 3 修改 PHP 的默认时区
- 4 使用微妙计算 PHP 脚本执行时间

UNIX 时间戳

自从 Unix 纪元（格林威治时间 1970 年 1 月 1 日 00:00:00）到当前时间的秒数。

相关函数

time(): 函数返回一个当前系统的时间戳

mktime(): 取得一个日期的 Unix 时间戳

格式: int mktime(时[,分[,秒[,月[,日[,年[,is_dst 区]]]]]);

注意: is_dst 参数表示是否为夏时制, PHP5.10 后此参数已废除。

strtotime(): 将任何英文文本的日期时间描述解析为 Unix 时间戳

格式: int strtotime (string \$time [, int \$now])

在 PHP 中获取日期和时间

getdate-- 取得日期 / 时间信息

格式: array getdate ([int timestamp])

返回一个根据 timestamp 得出的包含有日期信息的结合数组。如果没有给出时间戳则认为是当前本地时间。

```
<?php
//时间戳
echo time(); //获取当前系统的时间戳

echo "<br/>";

//给时分秒月日年, 来获取一个时间戳 //格式要求比较严格
echo "2014-02-21 11:12:30时间戳是: ".mktime(11,12,30,2,21,2014);
echo "<br/>";

$str="2014-02-21 11:12:30";//常用
echo $str."时间戳是: ".strtotime($str); //将一个字串式的日期, 转成时间戳

echo "<br/>";
echo "今天的时间戳: ".strtotime("0 day")."<br/>";
echo "明天的时间戳: ".strtotime("+1 day")."<br/>";
echo "昨天的时间戳: ".strtotime("-1 day")."<br/>";
echo "3天前的时间戳: ".strtotime("-3 day")."<br/>";

echo "<br/>";
echo "<h3>getdate()函数的使用</h3>";
$dd = getdate(time()+8*3600); //找回失去8小时 (时区问题)
echo '<pre>';
print_r($dd);
```

```
1424491707  
2014-02-21 11:12:30时间戳是: 1392981150
```

```
2014-02-21 11:12:30时间戳是: 1392981150  
今天的时间戳: 1424491707  
明天的时间戳: 1424578107  
昨天的时间戳: 1424405307  
3天前的时间戳: 1424232507
```

getdate() 函数的使用

```
Array  
(  
    [seconds] => 27  
    [minutes] => 8  
    [hours] => 12  
    [mday] => 21  
    [wday] => 6  
    [mon] => 2  
    [year] => 2015  
    [yday] => 51  
    [weekday] => Saturday  
    [month] => February  
    [0] => 1424520507  
)
```

日期和时间的格式化输出

date -- 格式化一个本地时间 / 日期

格式: string date (string format [, int timestamp])

```
<?php  
//date函数的使用
```

```
//设置和获取时区  
echo "获取默认时区: ".date_default_timezone_get()."<br/>"; //UTC  
date_default_timezone_set("PRC"); //设置中国时区  
  
echo date("Y-m-d H:i:s")."<br/>";  
echo date("Y年m月d日 H:i:s 星期w")."<br/>";  
  
echo "<br/>";  
  
$str = "2012-12-24"; //获取指定时间是星期几?  
echo date("w",strtotime($str)); //先将给的时间转成时间戳，然后求星期几
```

获取默认时区: UTC
2015-02-21 12:26:49
2015年02月21日 12:26:49 星期6

1

修改 PHP 的默认时区

1、修 php.ini 配置文件:

date.timezone = Etc/GMT+8

2、date_default_timezone_set(): -- 设定用于一个脚本中所有日期时间函数的默认时区。

如: date_default_timezone_set("PRC"); //中国时区。

date_default_timezone_get(): -- 获取当前时区

使用微秒计算 PHP 脚本执行时间

microtime -- 返回当前 Unix 时间戳和微秒数

格式: mixed microtime ([bool get_as_float])

```
<?php
/*microtime -- 返回当前 Unix 时间戳和微秒数

echo microtime(); //默认返回字符串格式: 0.64929600 1424493049
echo "<br/>";
echo microtime(true); //返回浮点数: 1424493049.6493

echo "<hr/>";
//计算100万次数值比较的耗时。
$t1 = microtime(true);
for($i=0;$i<1000000;$i++){
    if("abc"=="abd"){
    }
}
$t2 = microtime(true);

echo "脚本耗时:".($t2-$t1); //0.26101517677307
```

PHP 错误日志、时间日期函数

一、 *php 中错误分类:

1. 语法错误:
2. 运行时错误:
3. 逻辑错误:

二、 *在 php.ini 配置文件中，常用的错误和日志的配置。

error_reporting: 错误等级
display_errors: 在浏览器中是否显示错误信息
log_errors=On; 是否启动日志记录

ini_set() //php.ini 配置信息临时设置函数
ini_set("display_errors","On"); //通过函数设置，实现当前脚本临时关闭错误输出。
error_reporting(E_ALL & ~E_NOTICE); //临时设置错误输出级别。

三、 PHP 的日志记录方式：

-
- *1. 采用文件记录，
 - 2. 依靠系统的服务信息帮助记录

**1.采用文件记录日志

1、配置：在 php.ini 配置文件中配置如下信息
error_reporting = E_ALL //将向 PHP 发送每个错误
display_errors=Off //不显示错误报告
* log_errors=On //决定日志语句记录的位置。
log_errors_max_log=1024 // 每个日志项的最大长度
* error_log=G:/myerror.log //指定错误写进的文件
2、使用函数：在 php 文件中使用 error_log() 来记录日志,就可以将信息写入到 myerror.log 文件中
如：error_log("登录失败了！");//人为的记录错误信息
注意：当前 php 程序报错时，信息也会自动写入到 myerror.log

2. 依靠系统的服务信息帮助记录日志

1、先配置 PHP.ini 文件中
error_reporting = E_ALL //将向 PHP 发送每个错误
* display_errors=Off //不显示错误报告
* log_errors=On //决定日志语句记录的位置。
log_errors_max_log=1024 // 每个日志项的最大长度
* error_log=syslog //指定到系统日志中。

2、使用四个函数来记录日志：

```
define_syslog_variables(); //为系统日志初始化配置  
openlog(); //打开一个日志链接  
syslog(); //发送一条日志记录  
closelog(); //关闭日志链接
```

四、 日期 / 时间函数

checkdate -- 验证一个格里高里日期
date_default_timezone_get -- 取得一个脚本中所有日期时间函数所使用的默认时区

*date_default_timezone_set -- 设定用于一个脚本中所有日期时间函数的默认时区
date_sunrise -- 返回给定的日期与地点的日出时间
date_sunset -- 返回给定的日期与地点的日落时间
**date -- 格式化一个本地时间 / 日期
getdate -- 取得日期 / 时间信息
gettimeofday -- 取得当前时间
gmdate -- 格式化一个 GMT/UTC 日期 / 时间
gmmktime -- 取得 GMT 日期的 UNIX 时间戳
gmstrftime -- 根据区域设置格式化 GMT/UTC 时间 / 日期
idate -- 将本地时间日期格式化为整数
localtime -- 取得本地时间
*microtime -- 返回当前 Unix 时间戳和微秒数
*mktime -- 取得一个日期的 Unix 时间戳
strftime -- 根据区域设置格式化本地时间 / 日期
strptime -- 解析由 strftime() 生成的日期 / 时间
**strtotime -- 将任何英文文本的日期时间描述解析为 Unix 时间戳
**time -- 返回当前的 Unix 时间戳

万年历实例：

```
<html>
<head>
    <title>万年历</title>
    <meta charset='utf-8'>
</head>
<body>
    <center>
        <?php
            $year=$_GET["y"]?$_GET['y']:date("Y"); //获取当前年
            $month=$_GET["m"]?$_GET['m']:date("m"); //获取当前月

            $day=date("t",mktime(0,0,0,$month,1,$year)); //当前月天数
            $w   =date("w",mktime(0,0,0,$month,1,$year)); //当前月 1 号星期几？

            echo "<h3>{$year}年 {$month} 月</h3>";
        ?>

        <table width="600" border="1">
            <tr>
                <th>星期日</th><th>星期一</th>
                <th>星期二</th><th>星期三</th>
                <th>星期四</th><th>星期五</th>
                <th>星期六</th>
            </tr>
```

```

<?php
    $dd=1;//日
    while($dd<=$day){
        echo "<tr>";
        //每次都循环一周
        for($i=0;$i<7;$i++){
            if(($i<$w && $dd==1)|| $dd>$day){
                echo "<td>&nbsp;</td>";
            }else{
                echo "<td>{$dd}</td>";
                $dd++;
            }
        }
        echo "</tr>";
    }

/*
if($dd==$day && ($i>=$w || $dd!=1)){
    echo "<td>{$dd}</td>";
    $dd++;
}else{
    echo "<td>&nbsp;</td>";
}
*/
echo "</table><br/><br/>";
$py=$ny=$year;
$pm=$month-1; //上一月
$nm=$month+1; //下一月
if($pm==0){
    $pm=12;
    $py--;
}
if($nm==13){
    $nm=1;
    $ny++;
}

echo "<a href='date.php?y={$py}&m={$pm}'>上一月</a>&nbsp;";
echo "<a href='date.php?y={$ny}&m={$nm}'>下一月</a>";
?>
</center>

```

```
</body>
</html>
```

十、PHP 中 GD 库的使用

- 1 GD 简介
- 2 画布管理
- 3 设置颜色
- 4 生成图像
- 5 绘制图像
- 6 在图像中绘制文字

GD 简介

PHP 不仅限于只产生 HTML 的输出，还可以创建及操作多种不同格式的图像文件。PHP 提供了一些内置的图像信息函数，也可以使用 GD 函数库创建新图像或处理已有的图像。目前 GD2 库支持 GIF、JPEG、PNG 和 WBMP 等格式。此外还支持一些 FreeType、Type1 等字体库。

JPEG 是一种压缩标准的名字，通常是用来存储照片或者存储具有丰富色彩和色彩层次的图像。这种格式使用了有损压缩。

PNG 是可移植的网络图像，对图像采用了无损压缩标准。

GIF 原义是“图像互换格式”，是一种基于 LZW 算法的连续色调的无损压缩格式。

GD 库图像绘制的步骤

在 PHP 中创建一个图像应该完成如下所示的 4 个步骤：

1. 创建一个背景图像（也叫画布），以后的操作都基于此背景图像。
2. 在背景上绘制图像轮廓或输入文本。
3. 输出最终图形
4. 释放资源



画布管理

imagecreate -- 新建一个基于调色板的图像

```
resource imagecreate ( int x_size, int y_size )
```

本函数用来建立空新画布，参数为图片大小，单位为像素 (pixel)。支持 256 色。

imagecreatetruecolor -- 新建一个真彩色图像

```
resource imagecreatetruecolor ( int x_size, int y_size )
```

新建一个真彩色图像画布，需要 GD 2.0.1 或更高版本，不能用于 GIF 文件格式。

imagedestroy -- 销毁一图像

```
bool imagedestroy ( resource image )
```

imagedestroy() 释放与 image 关联的内存。

设置颜色

```
imagecolorallocate -- 为一幅图像分配颜色  
$im = imagecreatetruecolor(100, 100); //创建画布的大小为 100x100  
$red = imagecolorallocate($im,255,0,0); //由十进制整数设置一个颜色  
$white = imagecolorallocate($im, 0xFF, 0xFF, 0xFF); // 十六进制方式
```

生成图片

```
imagegif -- 以 GIF 格式将图像输出到浏览器或文件  
    语法: bool imagegif (resource image [,string filename] )  
imagejpeg -- 以 JPEG 格式将图像输出到浏览器或文件  
    语法: bool imagejpeg (resource image [,string filename [, int quality]] )  
imagepng -- 以 PNG 格式将图像输出到浏览器或文件  
    语法: bool imagepng (resource image [,string filename] )  
imagewbmp -- 以 WBMP 格式将图像输出到浏览器或文件  
    语法: bool imagewbmp (resource image [, string filename [, int foreground]] )
```

绘制图像

```
imagefill -- 区域填充  
imagesetpixel -- 画一个单一像素  
imageline -- 画一条线段  
imagerectangle -- 画一个矩形  
imagefilledrectangle -- 画一矩形并填充  
imagepolygon -- 画一个多边形  
imagefilledpolygon -- 画一多边形并填充  
imageellipse -- 画一个椭圆  
imagefilledellipse -- 画一椭圆并填充  
imagearc -- 画椭圆弧  
imagefilledarc -- 画一椭圆弧且填充
```

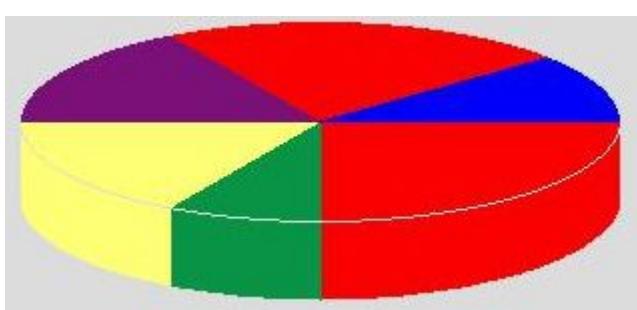
在图像中绘制文字

```
imagechar -- 水平地画一个字符  
imagecharup -- 垂直地画一个字符  
imageftttext -- 用 TrueType 字体向图像写入文本
```

```
//GD的绘画  
//1. 创建画布，准备颜色（画笔）  
//$im = imagecreate(200,200); //创建一个200*200的画布(支持256种颜色)  
$im = imagecreatetruecolor(200,200); //创建一个200*200的画布(真彩色,颜色多)  
$c = imagecolorallocate($im,240,145,181); //分配一个颜色  
  
//2. 开始绘画  
imagefill($im,0,0,$c); //填充背景颜色  
  
//3. 输出图片  
header("Content-Type: image/jpeg");  
imagejpeg($im);  
  
//4. 销毁资源  
imagedestroy($im);
```

饼状图

```
<?php  
//GD 的绘画(画圆弧)  
//1. 创建画布, 准备颜色 (画笔)  
$im = imagecreate(200,200); //创建一个 200*200 的画布  
$im = imagecreatetruecolor(400,400); //创建一个 200*200 的画布  
$c = imagecolorallocate($im,220,220,220); //分配一个颜色  
$c2 = imagecolorallocate($im,9,147,71); //分配一个颜色  
$c3 = imagecolorallocate($im,255,255,117); //分配一个颜色  
$c4 = imagecolorallocate($im,124,16,121); //分配一个颜色  
$blue = imagecolorallocate($im,0,0,250); //分配一个颜色  
$red = imagecolorallocate($im,250,0,0); //分配一个颜色  
  
//2. 开始绘画  
imagefill($im,0,0,$c); //填充背景颜色  
  
//画圆弧  
for($i=0;$i<40;$i++){  
    imagefilledarc($im,200,200-$i,300,100,0,90,$red,IMG_ARC_PIE);  
    imagefilledarc($im,200,200-$i,300,100,90,120,$c2,IMG_ARC_PIE);  
    imagefilledarc($im,200,200-$i,300,100,120,180,$c3,IMG_ARC_PIE);  
    imagefilledarc($im,200,200-$i,300,100,180,240,$c4,IMG_ARC_PIE);  
    imagefilledarc($im,200,200-$i,300,100,240,320,$red,IMG_ARC_PIE);  
    imagefilledarc($im,200,200-$i,300,100,320,360,$blue,IMG_ARC_PIE);  
}  
imagearc($im,200,161,300,100,0,360,$c);  
  
//3. 输出图片  
header("Content-Type: image/jpeg");  
imagejpeg($im);  
  
//4. 销毁资源  
imagedestroy($im);
```



验证码的绘制和使用

1. 验证码（CAPTCHA）是“Completely Automated Public Turing test to tell Computers and Humans

Apart”（全自动区分计算机和人类的图灵测试）的缩写，是一种区分用户是计算机和人的公共全自动程序。

2. 使用验证码的目的：可以防止：恶意破解密码、刷票、论坛灌水，有效防止某个黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试。

3. 验证码是现在很多网站通行的方式（比如招商银行的网上个人银行，百度社区）。

验证码：

```
<?php  
//GD 的绘画(验证码的绘制)  
  
$length=4; //验证码位数长度  
$code = getCode($length,1); //获取随机的验证码值  
$width = 20*$length;  
$height= 30;  
  
//1. 创建画布，准备颜色（画笔）  
$im = imagecreatetruecolor($width,$height); //创建一个画布  
//定义背景颜色  
$bg[] = imagecolorallocate($im,236,237,220); //分配一个颜色  
$bg[] = imagecolorallocate($im,244,220,219); //分配一个颜色  
$bg[] = imagecolorallocate($im,163,225,151); //分配一个颜色  
$bg[] = imagecolorallocate($im,217,220,213); //分配一个颜色  
//定义绘制颜色  
$c[] = imagecolorallocate($im,0,0,250); //分配一个颜色  
$c[] = imagecolorallocate($im,250,0,0); //分配一个颜色  
$c[] = imagecolorallocate($im,32,139,16); //分配一个颜色  
$c[] = imagecolorallocate($im,39,104,116); //分配一个颜色  
$c[] = imagecolorallocate($im,116,35,120); //分配一个颜色  
  
//2. 开始绘画  
imagefill($im,0,0,$bg[rand(0,3)]); //填充背景颜色  
  
//使用字体文件绘制字串  
for($i=0;$i<$length;$i++){  
    imagettftext($im,15,rand(-45,45),5+$i*20,25,$c[rand(0,4)],"./MSYHBD.TTF",$code[$i]);  
}  
  
//添加干扰点和线  
for($i=0;$i<100;$i++){  
    $color = imagecolorallocate($im,rand(0,255),rand(0,255),rand(0,255)); //分配一个随机颜色  
    imagesetpixel($im,rand(0,$width),rand(0,$height),$color);  
}  
for($i=0;$i<5;$i++){
```

```

$color = imagecolorallocate($im,rand(0,255),rand(0,255),rand(0,255)); //分配一个随机颜色
imageline($im,rand(0,$width),rand(0,$height),rand(0,$width),rand(0,$height),$color);
}

//3. 输出图片
header("Content-Type: image/jpeg");
imagejpeg($im);

//4. 销毁资源
imagedestroy($im);

//获取验证码字符串函数
function getCode($m=4,$type=1){
    $str = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
    //获取要随机字符串长度
    switch($type){
        case 1: $length=9; break;//纯数字,
        case 2: $length=33; break;//数字+小写字母,
        default: $length=strlen($str)-1; //数字+大小写字母 (所有)
    }

    //循环随机获取字符
    $s="";
    for($i=0;$i<$m;$i++){
        $k = rand(0,$length); //随机一个有效字符下标
        $s.=$str[$k]; //获取对应的字符
    }
    //返回结果
    return $s;
}

<img src='code.php' onclick="this.src='code.php?id='+Math.random()" />

```

[十一、PHP 图片处理](#)

图片背景管理

图片缩放和裁剪

添加图片水印

图片旋转和翻转

[1. 创建画布，准备颜色](#)

从指定的图片文件或 URL 地址来新建一个图像。成功则返回一个图像标识符，失败时返回一个空字符串，并且输出一条错误信息。由于格式不同，则需要分别使用对应图片背景处理函数。

[resource imagecreatefrompng \(string filename \)](#)

从 PNG 文件或 URL 新建一图像

[resource imagecreatefromjpeg \(string filename \)](#)

从 JPEG 文件或 URL 新建一图像

```
resource imagecreatefromgif ( string filename )
```

从 GIF 文件或 URL 新建一图像

```
resource imagecreatefromwbmp ( string filename )
```

从 WBMP 文件或 URL 新建一图像

2. 其他图像处理函数:

```
int imagesx ( resource image )
```

取得图像宽度

```
int imagesy ( resource image )
```

取得图像高度

```
array getimagesize ( string $filename [, array &$imageinfo ] )
```

取得图像大小、类型等信息

```
<?php
```

```
//GD的绘画(基于已知图片的绘画--图片水印)
```

```
//1. 创建画布，准备颜色（画笔）
```

```
$im = imagecreatefromjpeg("./8.jpg");
```

```
$logoim = imagecreatefromjpeg("./begin.jpg");
```

```
//获取图片源的宽度和高度
```

```
$w = imagesx($im);
```

```
$h = imagesy($im);
```

```
$lw = imagesx($logoim);
```

```
$lh = imagesy($logoim);
```

从源图片上的0,0位置获取高度为\$lw,高度\$lh的图片区域

```
//2. 开始绘画 源图片
```

```
imagecopyresampled($im,$logoim,$w-$lw-10,$h-$lh-10,0,0,$lw,$lh,$lw,$lh);
```

```
//3. 输出图片 目标图片
```

```
header("Content-Type: image/jpeg");
```

```
imagejpeg($im);
```

```
imagejpeg($im, "bb.jpg");
```

从目标图片上的坐标\$w-\$lw-10,\$h-\$lh-10
位置放置\$lw,\$lh大小的源图片

```
//4. 销毁资源
```

```
imagedestroy($im);
```

```
imagecopyresampled($im,$logoim,$w-$lw-10,$h-$lh-10,0,0,$lw/2,$lh/2,$lw,$lh);
```

```
imagecopyresampled($im,$logoim,$w-$lw-10,$h-$lh-10,0,0,$lw,$lh,$lw/2,$lh/2);
```



```

<?php
//GD的绘画(基于已知图片的绘画--添加文字水印)

//1. 创建画布, 准备颜色(画笔)
$im = imagecreatefromjpeg("./8.jpg");
//获取图片源的宽度和高度
$w = imagesx($im);
$h = imagesy($im);
$blue = imagecolorallocate($im, 0, 0, 250); //分配一个颜色
// $red = imagecolorallocate($im, 250, 0, 0); //分配一个颜色
|
//2. 开始绘画
imagettfttext($im, 25, 0, $w-200, $h-20, $blue, "./STCAIYUN.TTF", "LAMP兄弟连! ");

//3. 输出图片
header("Content-Type: image/jpeg");
imagejpeg($im);
imagejpeg($im, "aa.jpg");

//4. 销毁资源
imagedestroy($im);

```

3. 图片缩放和裁剪

`bool imagecopyresampled (resource $dst_image , resource $src_image , int $dst_x , int $dst_y , int
$src_x , int $src_y , int $dst_w , int $dst_h , int $src_w , int $src_h)`

重采样拷贝部分图像并调整大小, 是将一幅图像中的一块正方形区域拷贝到另一个图像中, 平滑地插入像素值, 因此, 尤其是, 减小了图像的大小而仍然保持了极大的清晰度。成功时返回 TRUE, 或者在失败时返回 FALSE。其中 `dst_image` 和 `src_image` 分别是目标图像和源图像的标识符。

4. 添加图片水印

`bool imagecopy (resource $dst_im , resource $src_im , int $dst_x , int $dst_y , int $src_x , int $src_y ,
int $src_w , int $src_h)`

拷贝图像的一部分(也就是图片合成)。

将 `src_im` 图像中坐标从 `src_x, src_y` 开始, 宽度为 `src_w`, 高度为 `src_h` 的一部分拷贝到 `dst_im` 图像中坐标为 `dst_x` 和 `dst_y` 的位置上。

5. 图片旋转和翻转

`resource imagerotate (resource $src_im , float $angle , int $bgd_color [, int $ignore_transparent])`

用给定角度旋转图像

将 `src_im` 图像用给定的 `angle` 角度旋转。`bgd_color` 指定了旋转后没有覆盖到的部分的颜色。

旋转的中心是图像中心, 旋转后的图像会按比例缩小以适合目标图像的大小——边缘不会被剪去。

如果 `ignore_transparent` 被设为非零值, 则透明色会被忽略(否则会被保留)。

```

<?php
//GD的绘画(基于已知图片的绘画--缩小一半)
//1. 创建画布, 准备颜色 (画笔)
$srcim = imagecreatefromjpeg("./8.jpg");
//获取图片源的宽度和高度
$w = imagesx($srcim);
$h = imagesy($srcim);

//创建新图
$dstim = imagecreatetruecolor($w/2,$h/2);

//2. 开始绘画
imagecopyresampled($dstim,$srcim,0,0,0,0,$w/2,$h/2,$w,$h);
//3. 输出图片
header("Content-Type: image/jpeg");
imagejpeg($dstim);

//4. 销毁资源
imagedestroy($dstim);
imagedestroy($srcim);

<?php
//GD的绘画(基于已知图片的裁剪)只支持jpg格式
//起始点216*38;宽高: 220*200

//1. 创建画布, 准备颜色 (画笔)
$srcim = imagecreatefromjpeg("./8.jpg");//原图
$dstim = imagecreatetruecolor(220,200); //创建一个目标图片(充当容器)

//2. 开始绘画
imagecopyresampled($dstim,$srcim,0,0,216,38,220,200,220,200);
//3. 输出图片      在源图片的坐标为216,38位置裁剪宽220,高200
header("Content-Type: image/jpeg");
imagejpeg($dstim);

//4. 销毁资源
imagedestroy($dstim);
imagedestroy($srcim);

<?php
header('Content-type:text/html;charset=utf-8');
//GD的绘画(基于已知图片的裁剪),特点支持jpg/gif/png格式的处理
//起始点216*38;宽高: 220*200
//已知变量
$picname="./8.jpg";
$x=216; //起始点x
$y=38; //起始点y
$w=220; //宽度
$h=200; //高度

```

```

//获取图片属性信息
$info = getimagesize($picname);
//var_dump($info);exit;
//1. 创建源图, 和目标图片源
switch($info[2]){
    //根据图片的格式选择对应创建
    case 1: //gif
        $srcim = imagecreatefromgif($picname);
        break;
    case 2: //jpeg
        $srcim = imagecreatefromjpeg($picname);
        break;
    case 3: //png
        $srcim = imagecreatefrompng($picname);
        break;
    default:
        die("无效的图片格式! ");
}
$dstim = imagecreatetruecolor(220,200); //创建一个目标图片

//2. 开始绘画 (裁剪)
imagecopyresampled($dstim,$srcim,0,0,216,38,220,200,220,200);

//3. 输出图片
header("Content-Type: ".$info['mime']);
switch($info[2]){
    //根据图片的格式选择对应输出
    case 1: //gif
        imagegif($dstim);
        break;
    case 2: //jpeg
        imagejpeg($dstim);
        break;
    case 3: //png
        imagepng($dstim);
        break;
}
//4. 销毁资源
imagedestroy($dstim);
imagedestroy($srcim);

```

定义成函数:

```

<?php
//GD 的绘画(基于已知图片的裁剪),特点支持 jpg/gif/png 格式的处理
//起始点 216*38;宽高: 220*200

```

```

//已知变量
$picname=".8.png";
$x=216; //起始点 x
$y=38; //起始点 y
$w=220; //宽度
$h=200; //高度

//测试:
cutImage($picname,$x,$y,$w,$h);

/**
 * 对已知图片裁剪函数
 * @param string $picname 被裁剪图片名及路径信息
 * @param int $x 裁剪起始位置 x 坐标
 * @param int $y 裁剪起始位置 y 坐标
 * @param int $w 裁剪的宽度
 * @param int $h 裁剪的高度
 * @return void 没有返回值
*/
function cutImage($picname,$x,$y,$w,$h){
    //获取图片属性信息
    $info = getimagesize($picname);
    //1. 创建源图, 和目标图片源
    switch($info[2]){ //根据图片的格式选择对应创建
        case 1://gif
            $srcim = imagecreatefromgif($picname);
            break;
        case 2://jpeg
            $srcim = imagecreatefromjpeg($picname);
            break;
        case 3://png
            $srcim = imagecreatefrompng($picname);
            break;
        default:
            die("无效的图片格式! ");
    }
    $dstim = imagecreatetruecolor(220,200); //创建一个目标图片

    //2. 开始绘画 (裁剪)
    imagecopyresampled($dstim,$srcim,0,0,216,38,220,200,220,200);

    //3. 输出图片
    header("Content-Type: ".$info['mime']);
}

```

```
switch($info[2]) { //根据图片的格式选择对应输出
    case 1: //gif
        imagegif($dstim);
        break;
    case 2: //jpeg
        imagejpeg($dstim);
        break;
    case 3: //png
        imagepng($dstim);
        break;
}

//4. 销毁资源
imagedestroy($dstim);
imagedestroy($srcim);
}
```

引用参数

在函数定义的时候，形参上面加了个& 符，这就是引用参数。
通常用在要修改参数值本身的时候用。

注意：

- 1、使用引用参数时，&要放在形参上。
- 2、使用引用参数时，参数必须使用变量，不能使用值。

自定义函数

- 1、先将代码写出来了。
- 2、然后用大括号包含起来
- 3、使用关键字 function 函数名() 来声明函数
- 4、将里面可以经常变的变量提出来作为参数。

注意：

- 1、函数的命名要有意义。
- 2、函数名不区分大小写。
- 3、函数命名规则最好遵循变量命名的规则。
- 4、函数名不能重复，不能和自定义的函数名称重复，也不能和系统的函数名重复。

PHP--GD 库

一、支持：需要 php 支持 gd 库

二、绘画步骤：

1. 创建一个画布（画板）、画笔、色彩。
2. *开始绘画
3. 输出图像（复制型）
4. 销毁图像资源（释放内存）

示例：

```
<?php
    //1.创建一个画布，颜色
    $im = imagecreate(200,200);
    $red = imagecolorallocate($im,255,0,0); //创建一个颜色：红色
    $blue = imagecolorallocate($im,0,0,255); //创建一个颜色：蓝色
    $c1 = imagecolorallocate($im,200,200,200);

    //2.开始绘画
    imagefill($im,0,0,$c1); //填充背景
    //....

    //3.输出图像
    header("Content-Type: image/jpeg");//设置响应头信息为一个 jpeg 的图片
    imagejpeg($im);//输出一个 jpeg 图片

    //4.销毁
    imagedestroy($im); //
```

三、图片的具体绘制：

3.1 创建一个画布：

```
imagecreate(宽,高)--创建一个基于 256 色的画布
imagecreatetruecolor( int x_size, int y_size ) -- 新建一个真彩色图像
//还有基于某个图片的画布
imagecreatefromgif( string filename )
imagecreatefrompng( string filename )
imagecreatefromjpeg( string filename )
```

3.2 绘画：

```
//分配定义颜色
$red = imagecolorallocate($im,255,0,0); //分配一个颜色

//填充背景
bool imagefill(resource image,int x,int y, int color ); //填充背景

//画点
bool imagesetpixel(resource image,int x,int y,int color ); //画一个像素点

//画一个线段的函数
bool imageline ( resource image, int x1, int y1, int x2, int y2, int color )
```

```

//画矩形框(不填充)
bool imagerectangle ( resource image, int x1, int y1, int x2, int y2, int color )
//画矩形框（填充）
bool imagefilledrectangle ( resource image, int x1, int y1, int x2, int y2, int color )

//绘制多边形
bool imagepolygon ( resource image, array points, int num_points, int color )
bool imagefilledpolygon ( resource image, array points, int num_points, int color )

//绘制椭圆（正圆）
imageellipse ( resource image, int cx, int cy, int w, int h, int color )
imagefilledellipse ( resource image, int cx, int cy, int w, int h, int color )

//绘制圆弧（可填充）
imagearc ( resource image, int cx, int cy, int w, int h, int s, int e, int color, int style )
imagefilledarc ( resource image, int cx, int cy, int w, int h, int s, int e, int color, int style )

//绘制字串
bool imagestring ( resource image, int font, int x, int y, string s, int col )
bool imagestringup ( resource image, int font, int x, int y, string s, int col )

//绘制文本：
*array imagettftext ( resource image, float size, float angle, int x, int y, int color, string fontfile,
string text )
当上面的字体出现乱码时，使用下面的函数转换编码
string iconv ( string in_charset, string out_charset, string str )

$name="张三";
$str = iconv("ISO8859-1","UTF-8",$name);
$str = iconv("GBK","UTF-8",$name);
$str = iconv("GB2312","UTF-8",$name);

//图片的裁剪、合成、缩放
**bool imagecopyresampled ( resource dst_image, resource src_image, int dst_x, int dst_y, int
src_x, int src_y, int dst_w, int dst_h, int src_w, int src_h )

* imagesx — 取得图像宽度
* imagesy — 取得图像高度
* array getimagesize ( string $filename [, array &$imageinfo ] ) 取得图像大小

```

3.3.输出图

```
header("Content-Type: image/jpeg");//设置响应头信息为一个 jpeg 的图片
```

```

imagejpeg($im); //输出一个 jpeg 图片

header("Content-Type: image/png"); //设置响应头信息为一个 png 的图片
imagepng($im); //输出一个 png 图片

//输出到指定文件中（另存为）
imagepng($im, "***.png");

```

3.4. 销毁

```
imagedestroy($im); //
```

十二、文件系统

- 1 文件类型
- 2 文件的属性

文件类型

在程序运行时，程序本身和数据一般都存在内存中，当程序运行结束后，存放在内存中的数据被释放。

如果需要长期保存程序运行所需的原始数据，或程序运行产生的结果，就必须以文件形式存储到外部存储介质上。

文件一般指存储在外部介质上具有名字（文件名）的一组相关数据集合。用文件可长期保存数据，并实现数据共享。

PHP 是以 UNIX 的文件系统为模型的。因此在 Windows 系统中我们只能获得” file”、” dir” 或者 “unknown” 三种文件类型。而在 UNIX 系统中，我们可以获得 block、char、dir、fifo、file、link 和 unknown 七种类型。

可以使用函数 filetype() 获取文件的具体类型。

语法：string filetype (string filename)

UNIX 系统中 7 种文件类型说明

文件类型	描述
block	块设备文件，如某个磁盘分区，软驱，光驱 CD-ROM 等
char	字符设备是指在 I/O 传输过程中以字符为单位进行传输的设备，如键盘、打印机等
dir	目录类型，目录也是文件的一种
fifo	命名管道，常用于将信息从一个进程传递到另一个进程
file	普通文件类型，如文本文件或可执行文件等。
link	符号链接，是指向文件指针的指针。类似 Windows

is_dir() -- 判断给定文件名是否是一个目录

语法结构: bool `is_dir` (名称)

返回类型: 文件名存在并且是一个目录则返回 true, 否则返回 false。

`is_executable()` -- 判断给定文件名是否可执行

语法结构: bool `is_executable` (名称)

返回类型: 如果文件存在且可执行则返回 true , 否则返回 false 。

`is_file()` -- 判断给定文件名是否为一个正常的文件

语法结构: bool `is_file`(名称)

返回类型: 如果文件存在且为正常的文件则返回 true 。

`is_link()` -- 判断给定文件名是否为一个符号连接

语法结构: bool `is_link`(名称)

返回类型: 如果文件存在并且是一个符号连接则返回 true。

`is_readable()` -- 判断给定文件名是否可读

语法结构: bool `is_readable` (文件名称)

返回类型: 如果文件存在并且可读则返回 true 。

`is_writable()` -- 判断给定的文件名是否可写

语法结构: bool `is_writable`(文件名称)

返回类型: 如果文件存在并且可写则返回 true 。

文件的属性

函数名	作用	参数	返回值
<code>file_exists()</code>	检查文件或目录是否存在	文件名	存在: true, 不存在: false
<code>filesize()</code>	取得文件大小	文件名	返回大小字节数, 出错: false
<code>is_readable()</code>	判断文件是否可读	文件名	文件可读返回 true
<code>is_writable()</code>	判断文件是否可写	文件名	文件可写返回 true
<code>is_executable()</code>	判断文件是否可执行	文件名	文件可执行返回 true
<code>filectime()</code>	获取文件的创建时间	文件名	返回 UNIX 时间戳格式
<code>filemtime()</code>	获取文件的修改时间	文件名	返回 UNIX 时间戳格式
<code>fileatime()</code>	获取文件的访问时间	文件名	返回 UNIX

十二、目录的基本操作

1. 解析目录路径 2 遍历目录 3 统计目录大小 4 建立与删除目录 5 复制目录

PHP 文件路径相关函数

`basename` -- 返回路径中的文件名部分

语法: string `basename` (string path [, string suffix])

给出一个包含有指向一个文件的全路径的字符串, 本函数返回基本的文件名。如果文件名是以 suffix 结束的, 那这一部分也会被去掉。

dirname -- 返回路径中的目录部分

语法: **string dirname (string path)**

给出一个包含有指向一个文件的全路径的字符串，本函数返回去掉文件名后的目录名。

pathinfo -- 返回文件路径的信息

语法: **array pathinfo (string path [, int options])**

pathinfo() 返回一个联合数组包含有 path 的信息。包括以下的数组单元: dirname, basename 和 extension。

realpath -- 返回规范化的绝对路径名

语法: **string realpath (string path)**

realpath() 扩展所有的符号连接并且处理输入的 path 中的 '.', '..' 以及多余的 '/' 并返回规范化后的绝对路径名。返回的路径中没有符号连接, '.' 或 '..' 成分。

<?php

//路径的相关函数: basename dirname realpath pathinfo

```
$url="http://www.baidu.com/aa/bb/c.php";  
  
echo "原值: ".$url."<br/>";  
echo "原值文件名: ".basename($url)."<br/>";  
echo "原值文件名(前缀): ".basename($url,".php")."<br/>";  
echo "原值路径名: ".dirname($url)."<br/>";  
echo "<hr/>";  
echo "pathinfo的使用<br/>";  
echo "原值: ".$url."<br/>";  
//var_dump(pathinfo($url));  
  
echo "原值文件名: ".pathinfo($url,PATHINFO_BASENAME)."<br/>";  
echo "原值路径名: ".pathinfo($url,PATHINFO_DIRNAME)."<br/>";  
echo "*原值路径的后缀名: ".pathinfo($url,PATHINFO_EXTENSION)."<br/>";  
  
echo "<hr/>";  
echo "文件a.txt绝对位置: ".realpath("test.php");
```

原值: http://www.baidu.com/aa/bb/c.php

原值文件名: c.php

原值文件名(前缀): c

原值路径名: http://www.baidu.com/aa/bb

pathinfo的使用

原值: http://www.baidu.com/aa/bb/c.php

原值文件名: c.php

原值路径名: http://www.baidu.com/aa/bb

*原值路径的后缀名: php

文件test.php绝对位置: D:\wamp\www\test\test.php

解析目录路径

使用 PHP 脚本可以方便对目录进行操作，如创建目录、遍历目录、复值目录与删除目录等操作。

常用的文件目录路径格式：

```
$unixPath="/var/www/html/index.php";
//在 UNIX 系统中的绝对路径，必须使用"/"分隔
$winPath="C:\\Appserv\\www\\index.php";
//在 Windows 系统的绝对路径，默认使用"\"分隔
$winPath2="C:/Appserv/www/index.php";
//在 Windows 系统中也可使用 "/" 分隔。
```

注意使用绝对路径与相对路径。

遍历目录

opendir -- 打开目录句柄

语法：resource opendir (string path [, resource context])

打开一个目录句柄，可用于之后的 closedir(), readdir() 和 rewinddir() 调用中。

readdir -- 从目录句柄中读取条目

语法：string readdir (resource dir_handle)

返回当前目录指针位置的文件名，没有返回 false，并将指针向下移动一位。文件名以在文件系统中的排序返回。

closedir -- 关闭目录句柄

语法：void closedir (resource dir_handle)

关闭由 dir_handle 指定的目录流。流必须之前被 opendir() 所打开。

rewinddir -- 倒回目录句柄

语法：void rewinddir (resource dir_handle)

将 dir_handle 指定的目录流重置到目录的开头。

查看目录：

```
C:\Users\Administrator>d:

D:\wamp\www\test\bb>dir
驱动器 D 中的卷是 本地磁盘<开发>
卷的序列号是 0878-C53D

D:\wamp\www\test\bb 的目录

2015/02/23  13:57    <DIR>          .
2015/02/23  13:57    <DIR>          ..
2009/07/14  12:52            777,835 8.jpg
2013/09/02  09:18            158,711 8.PNG
2013/09/02  10:30            8 a.txt
2015/02/23  13:57    <DIR>          aa
2013/08/30  12:14            12,608 aa.jpg
2013/09/02  10:30            34 b.txt
2013/08/30  12:26            11,108 bb.jpg
2015/02/21  13:16            1,601 date.php
2006/10/13  19:42            14,685,876 MSYHBD.TTF
                           8 个文件      15,647,781 字节
                           3 个目录   34,451,861,504 可用字节
```

理解：while 和 for 的区别

```
<?php
    date_default_timezone_set( 'PRC' );
    //显示目录和文件
    function myfile($dirname){
        //打开目录
        $dir=opendir($dirname);
        readdir($dir); //读第一次 读的是.
        readdir($dir); //读第二次 读的是..
        while($filename=readdir($dir)){
            $file=$dirname.'/'.$filename;
            if(is_dir($file)){
                //是目录再调用用，遍历读取目录文件是从上往下读取的
                echo "<font color='red'>是目录{$file}</font><br>";
            }else{
                //不是目录 就是文件 让遍历
                echo "<font color='green'>是文件{$file}--|--".filesize($file)."--|--".filetype($file).
                    '--|--'.is_writable($file)."--|--".date("Y-m-d H:i:s",filectime($file))."</font><br>";
            }
        }
        //关闭资源
        closedir($dir);
    }

    myfile('./bb');
|
```

```
是文件./bb/8.jpg--|--777835--|--file--|--1--|--2015-02-22 14:05:57
是文件./bb/8.PNG--|--158711--|--file--|--1--|--2015-02-23 13:57:03
是文件./bb/a.txt--|--8--|--file--|--1--|--2015-02-23 13:57:03
是文件./bb/aa/a.txt--|--8--|--file--|--1--|--2015-02-23 13:57:03
是文件./bb/aa/aa.jpg--|--12608--|--file--|--1--|--2015-02-23 13:57:03
是文件./bb/aa/bb.jpg--|--11108--|--file--|--1--|--2015-02-23 13:57:03
是文件./bb/aa/cc/bbbb.txt--|--8--|--file--|--1--|--2015-02-23 15:01:55
是文件./bb/aa/cc/xxxx.jpg--|--12608--|--file--|--1--|--2015-02-23 15:02:06
是目录./bb/aa/cc
是目录./bb/aa
是文件./bb/aa.jpg--|--12608--|--file--|--1--|--2015-02-23 13:57:03
是文件./bb/b.txt--|--34--|--file--|--1--|--2015-02-23 13:57:03
是文件./bb/bb.jpg--|--11108--|--file--|--1--|--2015-02-23 13:57:03
是文件./bb/date.php--|--1601--|--file--|--1--|--2015-02-21 12:40:58
是文件./bb/MSYHBD.TTF--|--14685876--|--file--|--1--|--2015-02-21 18:19:34
```

统计目录大小：

第一版：兼容 Linux 系统

```
<?php
//统计指定目录大小的函数
function dirsiz($path){
    $sum=0; //定义一个用于存放目录大小的变量 sum
```

```

//打开一个目录
$dd = opendir($path);
//遍历目录中的每个文件
while($f = readdir($dd)){
    //过滤掉特殊目录.和..
    if($f=="." || $f==".."){
        continue;
    }
    $file = $path."/".$f; //加上路径
    //判断是否是文件
    if(is_file($file)){
        $sum+=filesize($file); //获取大小并累加
    }
    //判断是否是目录
    if(is_dir($file)){
        $sum+=dirsize($file); //递归调用
    }
}
//关闭目录
closedir($dd);
return $sum; //返回结果
}

//测试:
$path="./bb"; //被统计的目录名

echo dirsize($path);

```

第二版：使用引用参数

```

<?php
echo '<meta charset="utf-8" />';
//统计目录和文件个数
function total($dirname,&$dirnum,&$filenum){
    //打开目录
    $dir=opendir($dirname);
    readdir($dir);//读第一次 读的是.
    readdir($dir);//读第二次 读的是..
    while($filename=readdir($dir)){
        //echo $filename; 读取文件，读到没有为止
        $file=$dirname.'/'.$filename;
        if(is_dir($file)){
            total($file,$dirnum,$filenum);
            $dirnum++;
        }
    }
}

```

```

}else{
    //不是目录 就是文件 让文件自增
    $filenum++;
}
}

return $dirnum;
//关闭资源
closedir($dir);
}

//定义目录总数的变量
$dirnum=0;
//定义文件总数的变量
$filenum=0;
total('./bb',$dirnum,$filenum);
echo "目录总数:".$dirnum."<br/>";
echo '文件总数:'.$filenum."<br/>";

/*总结:如果我们在一个函数的参数上加入引用, 这表示
当函数对该内部变量的值进行修改时, 同时也能够反应到
函数的外部, 你只要在对应的参数前面加上"&"就 OK.*/

```

第三版：统计并转换大小

```

<?php
//文件大小转换函数
function tosize($size){
    if($size>pow(1024,3)){
        $size=round($size/pow(1024,3),2);
        $dw='GB';
    }elseif($size>pow(1024,2)){
        $size=round($size/pow(1024,2),2);
        $dw='MB';
    }elseif($size>1024){
        $size=round($size/1024,2);
        $dw='KB';
    }else{
        $dw='byte';
    }
    return $size.$dw;
}

//统计文件大小的函数
function dirsize($dirname){
    //定义目录大小
    $dirsize=0;

```

```

//打开目录
$dir=opendir($dirname);
//读取目录
while($filename=readdir($dir)){
    //过滤点和点点
    if($filename!='.' && $filename!= '..'){
        //拼接路径
        $file=$dirname.'/'.$filename;
        if(is_dir($file)){//判断是否是目录
            //将所有返回的目录大小累加到一起
            $dirsize+=dirsize($file);//递归调用统计目录大小
        }else{
            $dirsize+=filesize($file);
        }
    }
}
closedir($dir);
return $dirsize;//返回目录大小
}

//调用统计目录大小函数
echo tosize(dirsize('./bb'));

```

统计空间大小

```

echo "当前磁盘可用大小: ".(disk_free_space("./"))/1024/1024/1024)."G<br/>";
echo "当前磁盘共计大小: ".(disk_total_space("./"))/1024/1024/1024)."G<br/>";

```

文件的基本操作

| | | | |
|----------|---------|-------------|--------|
| 文件的打开与关闭 | 写入文件 | 读取文件内容 | 访问远程文件 |
| 移动文件指针 | 文件的锁定机制 | 文件的一些基本操作函数 | |

目录和文件的基本操作函数

```

<?php
//目录的相关函数
/*
mkdir("qq");//创建目录
sleep(5); //此处睡眠5秒
rmdir("qq"); //删除一个空目录
*/
/*
mkdir("./cc/ee");//无法递归创建目录
rmdir("cc");
*/

```

```

//删除指定的文件(条件不能颠倒)

$filename = "a.txt";
if(file_exists($filename) && is_file($filename)){
    unlink($filename);
} else{
    die($filename."文件不存在或不是有效文件! ");
}

//touch('test.txt');//创建文件

//rename('test.txt','aa.txt');//重命名文件

//rename('aa.txt','./qq/aa.txt');//移动文件

//rename('aa.txt','./qq/bb.txt');//移动并重命名文件

```

删除指定目录的函数

```

<?php
//删除指定目录的函数

function deldir($path){
    //打开目录
    $dd = opendir($path);
    //遍历目录中的所有文件
    while($f = readdir($dd)){
        //过滤掉特殊目录.和..
        if($f == "." || $f == ".."){
            continue;
        }
        //为文件加上目录(有就清,没有就不清)
        $file = rtrim($path,"/")."/".$f;
        if(is_file($file)){
            unlink($file);
        }
        if(is_dir($file)){
            deldir($file); //递归调用删除
        }
    }
    //关闭目录
    closedir($dd);
    //删除目录
    rmdir($path);
}
//测试:
$path = "./cc/";
deldir($path);

```

建议：问题细分化，功能替代化，互联网解答

```
<?php
//统计数组长度的函数（递归、统计一维、多维）
|
function countArray($list){
    $sum=0;
    foreach($list as $v){
        $sum++;
        if(is_array($v)){
            //统计v数组的长度
            $sum+=countArray($v); //递归统计

            /*$i=0;
            foreach($v as $vv){
                $i++;
            }
            $sum+=$i;*/
        }
    }
    return $sum;
}

//测试
$a = array(10,20,340,40,50,array(10,20,30),array(30,40));

echo countArray($a);
//echo count($a,true);
```

复制目录（解决问题：先实现第一层复制）

```
<?php
//复制目录

function copydir($srcdir,$dstdir){
    //判断被复制源目录是否无效
    if(!file_exists($srcdir) || !is_dir($srcdir)){
        die("无效的源目录！");
    }
    //判断目标目录是否存在，若不存在则尝试创建
    if(!file_exists($dstdir)){
        @mkdir($dstdir);
    }
    if(!is_dir($dstdir)){
        die("目标目录创建失败！");
    }
}
```

```

//打开源目录(核心代码)
$dd = opendir($srcdir);
//遍历目录中的文件
while($f = readdir($dd)){
    //排除特殊目录.和..
    if($f=="." || $f==".."){
        continue;
    }
    //加上源路径
    $srcfile = $srcdir."/".$f;
    $dstfile = $dstdir."/".$f;
    //复制
    if(is_file($srcfile)){
        copy($srcfile,$dstfile);
    }
    if(is_dir($srcfile)){
        copydir($srcfile,$dstfile); //若是目录采用递归调用，继续复制
    }
}
//关闭目录
closedir($dd);

}

//测试:
copydir("./aa","./cc");

```

文件操作模式

文件的打开与关闭

fopen -- 打开文件或者 URL

fclose -- 关闭一个已打开的文件指针

fwrite -- 写入文件 (可安全用于二进制文件)

fread-- 读文件 文件大小: filesize(\$filename)

*1. fopen(文件名, 打开模式) ---打开一个文件

模式: 读 (r、r+)、 清空写 (w/w+)， 追加写 (a/a+)、 创建模式(x,x+)

*打开模式: *r: 只读模式打开文件, 文件指针执行首位。

r+: 可读可写模式打开文件, 文件指针执行首位,若文件指针不是在最后, 则是覆盖写。

*w: 以写方式打开文件, 文件内容清空, 若文件不存在, 则尝试创建。

w+: 以写和读方式打开文件, 文件内容清空, 若文件不存在, 则尝试创建。(也会覆盖写)

*a: 以追加内容方式打开文件, 指针移至最后, 若文件不存在, 则尝试创建。

a+: 以追加内容和可读方式打开文件, 指针移至最后, 若文件不存在, 则尝试创建。

x: 以创建方式打开文件, 可写。若文件已存在, 则报错。

x+: 以创建方式打开文件, 可写可读。若文件已存在, 则报错。

附加模式:

t: txt 文本模式(字符流)

b: byte 字节模式 (二进制) 默认 (字节流)

如: fopen("a.txt","rb");

文件的读写操作也叫流操作, 其中流分为字符流(t)和字节流(b 二进制)

1. 读 (r、r+) 实例:

```
<?php
//文件操作
//fopen打开一个文件, fclose关闭一个文件, fwrite写、 fread读

$f=fopen("a.txt","r+"); //以r模式(只读) 打开文件a.txt

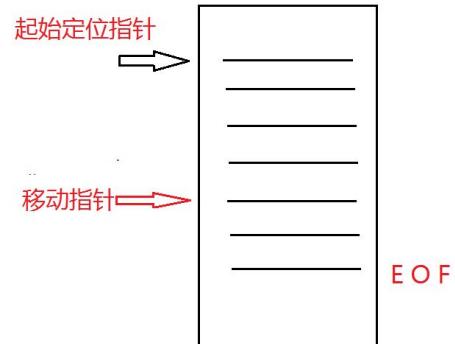
echo fread($f,4)."<br/>"; //读取了4个字符,能有多少读多少
echo fread($f,1)."<br/>";//接着又读取1个字符
fwrite($f,"**");//注意: [不是追加模式打开将会导致覆盖写入
echo fread($f,5)."<br/>"; //读取5个字符
//echo fread($f,1)."<br/>"; //读取1个字符
echo filesize('a.txt');
fclose($f); //关闭

//读写文件内容时存在文件指针的移动, 读到那里就移动到那, 写到那开始写入
//hello**hp work!
```

```
$f = fopen("a.txt", "r");//hello**hp work!

echo "打开后文件指针位置:".ftell($f)."<br/>";
echo "读取3个字符:".fread($f,3)."<br/>";
rewind($f); //将文件指针移至首位
echo "移至首位读取3个字符:".fread($f,3)."<br/>";
echo "读取文件后指针位置:".ftell($f)."<br/>";

fclose($f);
```



```
<?php
//文件操作
//ftell() --返回文件指针的位置
//fseek() --设置文件指针的位置

$f = fopen("a.txt", "r"); //文件内容是zhangsan

echo "读取4个字符:".fread($f,4)."<br/>";
echo "读取文件后指针位置:".ftell($f)."<br/>";

//fseek($f,-2,SEEK_CUR);//从当前位置移动,负数向前移动
//fseek($f,2,SEEK_SET);//从最前位置移动,必须是正数
fseek($f,-6,SEEK_END);//从最后位置移动,必须是负数

//要求读取三个字母的内容是: ang
echo "读取3个字符:".fread($f,3)."<br/>";

fclose($f);
```

2. 清空写 (w/w+)

```
<?php
//以清空写方式打开文件dd.txt,若文件不存在, 则尝试创建
$f = fopen("dd.txt", "w");
fwrite($f, "hello world!");
fwrite($f, "hello world!");
fwrite($f, "hello world!");
/*清空写是指当文件选择打开模式时清空写入内容,并不表示
每次写入内容都清空写.以上形式已写入了3个字符串*/
fclose($f); //hello world!hello world!hello world!
```

```
//以清空写(加可读)方式打开文件dd.txt,若文件不存在, 则尝试创建
```

```
$f = fopen("dd.txt", "w+");
fwrite($f, "hello world!");
fwrite($f, "hello world!");
fwrite($f, "hello world!");
//写入了3个字符串
```

```
//读取 (写入指针已移至最后位置)
```

```
echo "第一次读取:". fread($f, 20). "<br/>";
rewind($f); //将文件指针移至首位
echo "第二次读取(移动指针后):". fread($f, 20). "<br/>";
fwrite($f, "#"); //由于当前文件指针不在最后, 所以会导致覆盖写。
```



```
fclose($f); //hello world!hello wo##d!hello world!
```

3. 追加写 (a/a+) --- 写入日志采用

```
//以追加写方式打开文件ee.txt,若文件不存在, 则尝试创建
```

```
$f = fopen("ee.txt", "a");
fwrite($f, "hello world!\n");

fclose($f);
1 hello world!
2 hello world!
3 hello world!
```

```
//以追加写(加可读)方式打开文件ee.txt,若文件不存在, 则尝试创建
```

```
$f = fopen("ee.txt", "a+");
fwrite($f, "hello world!\n");
//读取
echo "第一次读取:". fread($f, 20). "<br/>";
rewind($f); //将文件指针移至首位
echo "第二次读取(移动指针后):". fread($f, 20). "<br/>";
fwrite($f, "#"); //追加模式打开一直存在由于是追加写, 所以会写在最后。

fclose($f);
```

第一次读取：

第二次读取（移动指针后）：hello world! hello w|

实例：自定义 copy 函数（默认 b 模式）

```
<?php
//自定义复制文件函数

function mycopy($file1,$file2){
    $f1 = fopen($file1,"rb");
    $f2 = fopen($file2,"wb");
    while($s = fread($f1,1024)){
        fwrite($f2,$s);
    }
    fclose($f1);
    fclose($f2);
}

mycopy("aa.jpg","bb.jpg");
```

读取文件内容

| 函数名 | 作用 |
|---------------------|--------------------------------|
| readfile() | 读取文件（可安全用于二进制文件） |
| file_get_contents() | 将文件读入字符串 |
| fgets() | 从打开的文件中读取一行 |
| fgetc() | 从打开的文件中读取一个字符 |
| file() | 把文件读入一个数组中（无需使用 fopen 打开） |
| readfile() | 读取一个文件，并输出到输出缓冲（无需使用 fopen 打开） |

```
<?php
$f = fopen("ee.txt","r");
while($s = fgets($f)){
    echo "<li>".$s."</li>\n";
}
fclose($f);

<?php
//file的使用(一次性将ee.txt文件内容读出，并以每行的方式形成一个数组返回)
$a = file("ee.txt");
echo "<pre>";
var_dump($a);
echo "</pre>";
```

```
array (size=5)
  0 => string 'hello world!
, (length=13)
  1 => string 'hello world!
, (length=13)
  2 => string 'hello world!
, (length=13)
  3 => string 'hello world!
, (length=13)
  4 => string '##' (length=2)
```

```
<?php
readfile("ee.txt"); //读取文件ee.txt内容并直接输出到浏览器
hello world! hello world! hello world! hello world! ##
```

=====

php--文件处理系统

=====

一、概述:

1.1 文件类型:

文件在 windows 系统下分为 3 种不同：文件、目录、未知

在 linux/unix 系统下分为 7 种不同：block、char、dir、fifo、file、link 和 unknown 七种类型

1.2 文件的常用函数:

* filetype() --获取文件类型的函数：

* is_dir() -- 判断给定文件名是否是一个目录

语法结构：bool is_dir (名称)

返回类型：如果文件名存在并且是一个目录则返回 true，否则返回 false。

is_executable() -- 判断给定文件名是否可执行

语法结构：bool is_executable (名称)

返回类型：如果文件存在且可执行则返回 true，否则返回 false。

* is_file() -- 判断给定文件名是否为一个正常的文件

语法结构：bool is_file(名称)

返回类型：如果文件存在且为正常的文件则返回 true。

is_link() -- 判断给定文件名是否为一个符号连接

语法结构: bool is_link(名称)

返回类型: 如果文件存在并且是一个符号连接则返回 true。

is_readable() -- 判断给定文件名是否可读

语法结构: bool is_readable (文件名称)

返回类型: 如果文件存在并且可读则返回 true 。

* is_writable() -- 判断给定的文件名是否可写

语法结构: bool is_writable(文件名称)

返回类型: 如果文件存在并且可写则返回 true 。

*file_exists() 检查文件或目录是否存在

*filesize() 取得文件大小 (不能获取目录大小)

is_readable() 判断文件是否可读

is_writable() 判断文件是否可写

*filectime() 获取文件的创建时间

filemtime() 获取文件的修改时间

fileatime() 获取文件的访问时间

stat() 获取文件大部分属性

二、目录的相关操作

2.1 路经的方式:

windows 系统: D:/a/b/c.php 或 D:\a\b\c.php

linux/unix 系统: /a/b/c.php

为统一建议使用"/"作为目录之间的分割符

2.2 路径的相关函数(4 个):

1. basename("url 地址"[, 去除部分]) -- 获取指定 url 的文件名

2. dirname("url 地址") -- 获取地址的路径值

示例:

```
$url = "http://www.baidu.com/a/b/c.php";
```

```
echo $url."<br/>"; //http://www.baidu.com/a/b/c.php
```

```
echo basename($url)."<br/>";//c.php
```

```
echo basename($url,".php")."<br/>"; //c (去掉".php")
```

```
echo dirname($url)."<br/>"; //http://www.baidu.com/a/b
```

*3. pathinfo (文件路径[,需要的下标]) --获取文件路径的详细信息,返回一个关联数组

结果: 下标: dirname 路径名

basename 文件名

extension 后缀名

filename 文件名 (去掉后缀的)

示例：

```
$url = "http://www.baidu.com/a/b/c.php";
*echo pathinfo($url,PATHINFO_DIRNAME); //http://www.baidu.com/a/b
*echo pathinfo($url,PATHINFO_EXTENSION); //php 后缀名
*echo pathinfo($url,PATHINFO_BASENAME); //c.php 文件名
$a = pathinfo($url);
$a 结果:
array(4) {
    ["dirname"]=>
    string(24) "http://www.baidu.com/a/b"
    ["basename"]=>
    string(5) "c.php"
    ["extension"]=>
    string(3) "php"
    ["filename"]=>
    string(1) "c"
}
```

4. realpath()--获取指定文件的绝对路径

示例： echo realpath("1.php"); //D:\AppServ\www\lamp45\09_file_dir\1.php

2.3 目录的遍历函数：

*opendir("") -- 打开一个目录，返回此目录的资源句柄
*readdir(资源句柄) -- 从目中读取一个目录或文件，并指针向下移动一位。
*closedir(资源句柄)-- 关闭打开的目录
rewinddir (资源句柄) -- 倒回目录指针（移至首位）

2.4 统计空间的大小

```
echo "当前磁盘可用大小: ".(disk_free_space("./")/1024/1024/1024)."G<br/>";
echo "当前磁盘共计大小: ".(disk_total_space("./")/1024/1024/1024)."G<br/>";
```

2.5 目录的操作

*mkdir() -- 创建一个目录
*rmdir() -- 删除一个目录（只支持删除空目录）
*unlink() -- 删除一个文件

2.6 复制文件：

copy() --- 复制文件。不支持目录复制

2.7 创建

touch() ----创建文件

2.8 重命名,移动,移动并重命名

rename()-- 重命名文件

三、文件基本操作

*1. fopen(文件名, 打开模式) ---打开一个文件

 模式: 读 (r、r+)、 清空写 (w/w+)， 追加写 (a/a+)、 创建模式(x,x+)

 *打开模式: *r: 只读模式打开文件, 文件指针执行首位。

 r+: 可读可写模式打开文件, 文件指针执行首位, 若文件指针不是在最后, 则是覆盖写。

 *w: 以写方式打开文件, 文件内容清空, 若文件不存在, 则尝试创建。

 w+: 以写和读方式打开文件, 文件内容清空, 若文件不存在, 则尝试创建。(也会覆盖写)

 *a: 以追加内容方式打开文件, 指针移至最后, 若文件不存在, 则尝试创建。

 a+: 以追加内容和可读方式打开文件, 指针移至最后, 若文件不存在, 则尝试创建。

 x: 以创建方式打开文件, 可写。若文件已存在, 则报错。

 x+: 以创建方式打开文件, 可写可读。若文件已存在, 则报错。

附加模式:

 t: txt 文本模式(字符流)

 b: byte 字节模式 (二进制) 默认 (字节流)

 如: fopen("a.txt","rb");

文件的读写操作也叫流操作, 其中流分为字符流(t)和字节流(b 二进制)

*2. fread(打开的文件资源, 读取的长度) -- 读取文件内容。

*3. fwrite(打开的文件资源, 被写入的字串[, 长度]) -- 向文件写入内容。

*4. fclose(打开的文件资源) -- 关闭文件

5. fgets() -- 从文件资源中读取一行

6. fgetc() -- 读取一个字符

-----不用打开文件可直接操作文件的函数-----

*7. file() 将文件读出 (内容是数组格式), 并返回

*8. readfile() 将文件内容读出, 并输出

9. file_get_contents() 读取文件内容

10. file_put_contents() 将指定内容写入文件

-----文件指针的操作-----

11. ftell () -- 返回文件指针的位置

12. fseek () -- 设置文件指针的位置

示例:

fseek(\$f,-2,SEEK_CUR); //从当前指针位置开始, 向前移动 2 位

```
fseek($f,2,SEEK_SET); //从文件指针开始位置, 向后移动 2 位  
fseek($f,-5,SEEK_END); //从文件指针的最后位置开始, 向前移动 5 位  
13. rewind() --将文件指针移至首位
```

-----文件锁（了解）-----

14. `bool flock (int $handle , int $operation [, int &$wouldblock])`

轻便的咨询文件锁定

`operation` 参数:

要取得共享锁定(读取的程序), 将 `operation` 设为 `LOCK_SH`(PHP 4.0.1 以前的版本设置为 1)。
要取得独占锁定(写入的程序), 将 `operation` 设为 `LOCK_EX`(PHP 4.0.1 以前的版本设置为 2)。
要释放锁定(无论共享或独占), 将 `operation` 设为 `LOCK_UN`(PHP 4.0.1 以前的版本设置为 3)。
若不希望 `flock()` 在锁定时堵塞, 则 `operation` 加上 `LOCK_NB`(PHP 4.0.1 以前的版本设置为 4)。

示例:

```
<?php
```

```
$fp = fopen("/tmp/lock.txt", "w+");  
if(flock($fp, LOCK_EX)) { // 进行排它型锁定  
    fwrite($fp, "Write something here\n");  
    flock($fp, LOCK_UN); // 释放锁定  
} else {  
    echo "Couldn't lock the file !";  
}  
fclose($fp);
```

```
?>
```

*15. `rename()` --修改文件名

16. `ftruncate()` — 将文件截断到给定的长度

十三、文件的上传和下载

1. 文件上传

客户端上传设置

在服务器端通过 PHP 处理上传

1.1 客户端上传设置

在 B/S 程序中文件上传已经成为一个常用功能。其目的是客户可以通过浏览器(Browser)将文件上传到服务器 (Server) 上的指定目录。

1.2 PHP 中文件上传的基础知识:

1.2.1 客户端 form 表单

1.2.2 服务器端对上传文件的操作

form.html 表单

```
<html>
<head><title>文件上传</title></head>
<body>
<form action="upload.php" method="post" enctype="multipart/form-data">
<input type="hidden" name="MAX_FILE_SIZE" value="1000000">
    选择文件: <input type="file" name="myfile">
    <input type="submit" value="上传文件">
</form>
</body>
</html>
```

注意几个特征属性：

POST 方法:

表单最常用的功能,向目标页面传递变量,我们在上传文件的时候,会在表单中设置相应的属性,来完成文件的传递, get 长度限制 2KB-8KB 不同浏览器值不同
enctype="multipart/form-data"

这样服务器就会知道,我们要传递一个文件,这样服务器可以知道上载的文件带有常规的表单信息。

MAX_FILE_SIZE

此字段必须在文件输入字段之前, 控制最大的传递文件的大小(字节)

```
<input type="file" name="userfile">
```

设置浏览器文件输入按钮

在服务器端通过 PHP 处理上传

设置 PHP 配置文件中的指令:用于精细地调节 PHP 的文件上传功能。

\$_FILES 多维数组: 用于存储各种与上传文件有关的信息, 其他数据还是使用\$_POST 获取。

PHP 的文件上传处理函数: 用于上传文件的后续处理。

未经处理的上传文件

```
<?php
var_dump($_FILES); //接收值

//sleep(10); //检测上传文件

//将上传文件复制到指定目录中

//第一版(通俗理解)
copy($_FILES['myfile']['tmp_name'], "./uploads/".$_FILES['myfile']['name']);

//第二版(安全性弱)
if(is_uploaded_file($_FILES['myfile']['tmp_name'])){
    move_uploaded_file($_FILES['myfile']['tmp_name'], "./uploads/".$_FILES['myfile']['name']);
}
```

1.3 \$_FILES 多维数组

超级全局数组\$_FILES

1、\$_FILES["myfile"]["name"]中的值是:

客户端文件系统的文件的名称

2、\$_FILES["myfile"]["**type**"]中的值是:

客户端传递的文件的类型

3、\$_FILES["myfile"]["**size**"]中的值是:

文件的字节的大小

4、\$_FILES["myfile"]["**tmp_name**"]中的值是:

文件被上传后在服务器存储的临时全路径

5、\$_FILES["myfile"]["**error**"]中的值是:

文件上传的错误代码—php 4.2 以后增加的功能

1.4 存储在\$_FILES["myfile"]["**error**"]中的值

值为 0: 表示没有发生任何错误。

值为 1: 表示上传文件的大小超出了约定值。文件大小的最大值是在 PHP 配置文件中指定的，该指令是: upload_max_filesize。

值为 2: 表示上传文件大小超出了 HTML 表单隐藏域属性的 MAX_FILE_SIZE 元素所指定的最大值。

值为 3: 表示文件只被部分上传。

值为 4: 表示没有上传任何文件。

值为 6: 表示找不到临时文件夹。PHP 4.3.10 和 PHP 5.0.3 引进。

值为 7: 表示文件写入失败。PHP 5.1.0 引进。

UPLOAD_ERR_OK : 对应值 0

UPLOAD_ERR_INI_SIZE : 对应值 1

UPLOAD_ERR_FORM_SIZE : 对应值 2

UPLOAD_ERR_PARTIAL : 对应值 3

UPLOAD_ERR_NO_FILE : 对应值 4

UPLOAD_ERR_NO_TMP_DIR : 对应值 6

UPLOAD_ERR_CANT_WRITE : 对应值 7

1.5 PHP 配置文件中与文件上传有关的选项

指令名	默认值	功能描述
file_uploads	ON	是否开启文件上传
upload_max_filesize	2M	限制 PHP 处理上传文件大小的最大值，此值必须小于 post_max_size
post_max_size	8M	限制通过 POST 方法可以接受信息的最大值，也就是整个 POST 请求的提交值。此值必须大于 upload_max_filesize
upload_tmp_dir	NULL	上传文件存放的临时路径，可以是绝对路径。默认 NULL 则使用系统的临时目录。

1.6 PHP 的文件上传处理函数

上传成功的文件会被放置到服务器端临时目录下，文件名是随机生成的临时文件名。

注：该文件在程序执行完后将自动被删除掉。在删除前可以像本地文件一样操作。

文件上传处理函数：

`is_uploaded_file` — 判断文件是否是通过 HTTP POST 上传的

 格式：bool `is_uploaded_file (string $filename)`

`move_uploaded_file` — 将上传的文件移动到新位置

 格式：bool `move_uploaded_file (string $filename , string $destination)`

 注意：如果目标文件已经存在，将会被覆盖。

2. 操作文件上传实例

```
<?php

//1. 设置常用变量信息
$typeList = array("image/jpg","image/jpeg","image/gif","image/png"); //允许的上传文件类型
$ext = array('jpg','png','gif','jpeg');//允许的后缀名
$maxsize = 500000; //设置允许上传文件大小
$upfile = $_FILES['myfile']; //获取上传信息
$path = "./uploads"; //指定上传的保存路径
$op = fopen('./name.txt','a+');//上传文件名记录文件

//2. 判断上传错误信息
if($upfile['error']>0){
    switch($upfile['error']){
        case 1: $info="上传的文件超过了php.ini中限制的值"; break;
        case 2: $info="上传文件的大小超过了HTML表单中 MAX_FILE_SIZE 选项指定的值"; break;
        case 3: $info="文件只有部分被上传"; break;
        case 4: $info="没有文件上传"; break;
        case 6: $info="找不到临时文件夹"; break;
        case 7: $info="临时文件写入失败"; break;
        default: $info="未知错误！"; break;
    }
    die("上传错误:".$info);
}

//3. 判断上传文件类型(可选)
if(!in_array($upfile['type'],$typeList)){
    die("上传文件类型错误: ".$upfile['type']);
}

//判断后缀名(可选)
$nowExt = pathinfo($upfile['name'],PATHINFO_EXTENSION); //获得上传文件的后缀名
if(!in_array($nowExt,$ext)){
    die('不是合法的后缀名');
}

//4. 判断上传文件大小(可选)
if($upfile['size']>$maxsize){
    die("上传文件大小超出！");
}
```

```

//判断上传目录是否存在(可选)
if(!file_exists($path)){
    |   mkdir($path);
}

//5.处理上传的文件名
$ext = pathinfo($upfile['name'],PATHINFO_EXTENSION); //获取上传文件的后缀名

//随机一个文件名并判断是否已存在。(如果存在就再随机一个文件名)
do{
    $filename = time().rand(10000,99999).".". $ext; //足以产生随机名
}while(file_exists($path."/". $filename));

/***
 * $name_arr=explode('.',$upfile['name']); //截取文件后缀
 * $hz=array_pop($name_arr); //接受文件后缀
 * //制作新的文件名
 * $filename=md5(time()).mt_rand(100,999).uniqid().'.'.$hz;
 */

//6.判断并执行文件上传
//判断是否是有效的上传文件
if(is_uploaded_file($upfile['tmp_name'])){
    //执行文件上传处理，并判断是否成功
    if(move_uploaded_file($upfile['tmp_name'],$path."/". $filename)){
        echo "上传文件成功: ".$filename;

        //用来存储 对应的随机名和原文件名
        $tmps = $filename.'|'.$upfile['name']. "\r\n";

        //写入文件中
        fwrite($op,$tmps);

    }else{
        die("上传文件失败! ");
    }
}else{
    die('无效上传文件! ');
}
//关闭文件
fclose($op);

<?php
/***
 * 文件上传函数
 * @param array $upfile 上传文件信息,如: $_FILES['myfile'];
 * @param string $path 上传文件保存路径,
 * @param array $typelist 允许上传文件类型, 默认空数组表示不限
 *           如: array("image/jpg","image/jpeg","image/gif","image/png"); //允许的上传文件类型
 * @param int $maxsize 允许上传文件大小, 0 表示不限制
 * @return array 返回数组, 其中下标 error(布尔值)表示是否成功、info 失败表示错误信息, 成功表示上传的文件名
 */
function uploadfile($upfile,$path,$typelist=array(),$maxsize=0){

```

```
//1. 定义一个返回结果
$res=array("error"=>false,"info"=>"");
//2.判断上传错误信息
if($upfile['error']>0){
    switch($upfile['error']){
        case 1: $info="上传的文件超过了 php.ini 中限制的值"; break;
        case 2: $info="上传文件的大小超过了 HTML 表单中 MAX_FILE_SIZE 选项指定的值";
        break;
        case 3: $info="文件只有部分被上传"; break;
        case 4: $info="没有文件上传"; break;
        case 6: $info="找不到临时文件夹"; break;
        case 7: $info="临时文件写入失败"; break;
        default: $info="未知错误！ "; break;
    }
    $res['info']=$info;
    return $res;
}

//3.判断上传文件类型(可选)
if(count($typelist)>0 && !in_array($upfile['type'],$typelist)){
    $res['info']="上传文件类型错误: ".$upfile['type'];
    return $res;
}

//4.判断上传文件大小(可选)
if($maxsize>0 && $upfile['size']>$maxsize){
    $res['info']="上传文件大小超出！ ";
    return $res;
}

//5.处理上传的文件名
$ext = pathinfo($upfile['name'],PATHINFO_EXTENSION); //获取上传文件的后缀名
do{ //随机一个文件名并判断是否已存在。
    $filename = time().rand(10000,99999).".". $ext;
}while(file_exists($path."/".$filename));

//6.判断并执行文件上传
//判断是否是有效的上传文件
if(is_uploaded_file($upfile['tmp_name'])){
    //执行文件上传处理，并判断是否成功
    if(move_uploaded_file($upfile['tmp_name'],$path."/".$filename)){
        $res['info']=$filename;
        $res['error']=true;
    }else{
}
```

```

        $res['info']="上传文件失败！";
    }
}else{
    $res['info']='无效上传文件！';
}
return $res;
}

//测试
//1.设置常用变量信息
$typeList =array("image/jpg","image/jpeg","image/gif","image/png"); //允许的上传文件类型
$maxsize = 500000; //设置允许上传文件大小
$upfile = $_FILES['myfile']; //获取上传信息
$path = "./uploads"; //指定上传的保存路径

$res = uploadfile($upfile,$path,$typeList,$maxsize);
if($res['error']==false){
    echo "错误: ".$res['info'];
}else{
    echo "成功! ".$res['info'];
}

多文件上传 第一版表单形式

<h2>多文件文件上传1</h2>
<form action="upload.php" method="post" enctype="multipart/form-data">
    文件: <input type="file" name="pic1"/><br/><br/>
    文件: <input type="file" name="pic2"/><br/><br/>
    文件: <input type="file" name="pic3"/><br/><br/>
    <input type="submit" value="提交"/>
</form>

<?php
//处理多文件上传

echo "<pre>";
print_r($_FILES);
echo "</pre>";

//导入函数库
require("functions.php");//文件上传,图片处理

//遍历$_FILES并执行上传
foreach($_FILES as $v){
    $res = uploadfile($v,"./uploads");
    if($res['error']==true){
        echo "上传成功! 文件名:".$res['info']."<br/>";
    }else{
        echo "上传失败! ".$res['info']."<br/>";
    }
}

```

```
<h2>多文件文件上传2</h2>
<form action="5.php" method="post" enctype="multipart/form-data">
    文件: <input type="file" name="pic[]"/><br/><br/>
    文件: <input type="file" name="pic[]"/><br/><br/>
    文件: <input type="file" name="pic[]"/><br/><br/>
    <input type="submit" value="提交"/>
</form>

<?php
//处理多文件上传

echo "<pre>";
print_r($_FILES); //三维数组
echo "</pre>";

//将不规则的上传信息进行转换(将三维转化成二维)
$list = array();
foreach($_FILES['pic']['name'] as $k=>$v){
    $list[$k]['name']=$v; //定位索引下标
    $list[$k]['type']=$_FILES['pic']['type'][$k]; //通过下标获取值再赋值
    $list[$k]['tmp_name']=$_FILES['pic']['tmp_name'][$k];
    $list[$k]['size']=$_FILES['pic']['size'][$k];
    $list[$k]['error']=$_FILES['pic']['error'][$k];
}
//目标数据(二维数组)
echo "<pre>";
print_r($list);
echo "</pre>";

exit;
//导入函数库
require("functions.php");

//遍历$_FILES并执行上传
foreach($list as $v){
    $res = uploadfile($v,"./uploads");
    if($res['error']==true){
        echo "上传成功! 文件名:".$res['info']."<br/>";
    }else{
        echo "上传失败! ".$res['info']."<br/>";
    }
}
```

2. 文件下载

```
<?php
//文件下载练习
$filename="./upload/aa.png";//指定下载文件
$basename=pathinfo($filename);
header("Content-Type: image/png");//指定下载文件类型的
header("Content-Disposition:attachment;filename=".$basename["basename"]);
//指定下载文件的描述信息
header("Content-Length:".$filesize($filename)); //指定文件大小的
readfile($filename);//将内容输出，以便下载。
```

经典实例：

◆ **readme.txt**

在线相册

```
-- index.php 浏览相册信息 ok
|
|-- add.php 发布相册信息表单 ok
|
|-- doAdd.php 执行发布相册信息 ok
|
|-- doDel.php 删除相册信息 ok
|
|-- download.php 下载
|
|-- functions.php 公共函数库，内提供文件上传和图片缩放 ok
|
|-- menu.php 公共导航栏页面 ok
|
|-- picinfo.txt 相册信息存储文件 ok
|
|-- pic/ //图片信息储存目录 ok
```

◆ **index.php**

```
<html>
<head>
    <title>我的相册</title>
    <meta charset="utf-8">
</head>
<body>
    <center>
        <?php include("menu.php"); //导入导航栏 ?>
```

```

<h3>浏览相册信息</h3>
<table width="700" border="1">
<tr>
    <th>图片名</th><th>图片</th>
    <th>类型</th><th>大小</th>
    <th>添加时间</th><th>操作</th>
</tr>
<?php
    //浏览图片信息
    //1. 读取 picinfo.txt 内容
    $contents = trim(file_get_contents("picinfo.txt"), "@");

    //2. 以@@@拆分一条条图片信息
    if(strlen($contents)>10){
        $list = explode("@@@", $contents);
    }
    //3. 遍历，继续以##拆分，并输出信息
    if(is_array($list)){
        foreach($list as $k=>$v){
            $pic = explode("##", $v);
            echo "<tr>";
            echo "<td>{$pic[0]}</td>";
            echo "<td><img src='./pic/s_{$pic[1]}'/></td>";
            echo "<td>{$pic[2]}</td>";
            echo "<td>{$pic[3]}</td>";
            echo "<td>".date("Y-m-d H:i:s", $pic[4])."</td>";
            echo "<td>
                <a href='doDel.php?id={$k}'>删除</a>,
                <a href='download.php?id={$k}' target='_blank'>下载</a>,
                <a href='./pic/{$pic[1]}'>查看</a>
            </td>";
            echo "</tr>";
        }
    }
?>
</table>
</center>
</body>
</html>

```

◆ add.php

```

<html>
    <head>

```

```
<title>我的相册</title>
<meta charset="utf-8">
</head>
<body>
    <center>
        <?php include("menu.php"); //导入导航栏 ?>

        <h3>相册信息发布</h3>
        <form action="doAdd.php" method="post" enctype="multipart/form-data">
            文件: <input type="file" name="pic"/><br/><br/>
            <input type="submit" value="提交"/>
        </form>

    </center>
</body>
</html>
```

❖ doAdd.php

```
<html>
    <head>
        <title>我的相册</title>
        <meta charset="utf-8">
    </head>
    <body>
        <center>
            <?php include("menu.php"); //导入导航栏 ?>

            <h3>相册信息添加</h3>
            <?php
                //执行相册信息的添加
                //定义变量信息
                $types = array("image/jpg","image/jpeg","image/gif","image/png"); //允许上传文件类型
                $path = "./pic"; //储存目录
                $upfile = $_FILES['pic']; //获取上传信息;

                //导入函数库文件
                require("functions.php");

                //执行上传
                $res = uploadfile($upfile,$path,$types);

                //判断是否成功
                if($res['error']==true){
                    //上传成功
                }
            </?php>
        </center>
    </body>
</html>
```

```
$picname = $res['info']; //获取图片名

//执行图片缩放
updateImage($picname,$path,"m_",150,150); //压缩一个中号图
updateImage($picname,$path,"s_",50,50); //压缩一个小号图*/
//信息添加
//添加信息：图片原名、现名、类型、大小、添加时间
$content =
$upfile['name']."'##".$picname."##".$upfile['type']."'##".$upfile['size']."'##".time()."@@";
//写入到 picinfo.txt 信息文件中
file_put_contents("picinfo.txt",$content,FILE_APPEND);
echo "发布成功！";
}else{
//报错
echo "上传失败！ ".$res['info'];
}
?>
</center>
</body>
</html>
```

◆ doDel.php

```
<html>
<head>
<title>我的相册</title>
<meta charset="utf-8">
</head>
<body>
<center>
<?php include("menu.php"); //导入导航栏 ?>

<h3>删除相册信息</h3>
<?php
//浏览图片信息
//1. 读取 picinfo.txt 内容
$contents = trim(file_get_contents("picinfo.txt"), "@");

//2. 以@拆分一条条图片信息
$list = explode("@", $contents);

//3. 获取要删除的信息
$info = $list[$_GET['id']];
$pic = explode("#", $info);
```

```

//4. 删除信息
unset($list[$_GET['id']]);

//5. 删图片
unlink("./pic/".$pic[1]);
unlink("./pic/m_".$pic[1]);
unlink("./pic/s_".$pic[1]);

//信息放回去
file_put_contents("picinfo.txt",implode("@@",$list)."@@");
echo "删除成功！";
?>
</table>
</center>
</body>
</html>

```

◆ menu.php

```

<h2>我的相册</h2>
<a href="index.php">浏览相册</a> |
<a href="add.php">发布信息</a>
<hr/>

```

◆ picinfo.txt

```

@@@Chrysanthemum.jpg##142486131515391.jpg##image/jpeg##879394##1424861315@@@Penguins.jpg##1
42486141719865.jpg##image/jpeg##777835##1424861417@@

```

◆ download.php

```

<?php
//执行下载
//1. 读取 picinfo.txt 内容
$contents = trim(file_get_contents("picinfo.txt"), "@");

//2. 以@@拆分一条条图片信息
$list = explode("@@", $contents);

//3. 获取要下载的信息
$info = $list[$_GET['id']];
$pic = explode("#", $info);

//设置头部信息
header("Content-Type:".$pic[2]); //设置下载类型
header("Content-Disposition:attachment;filename=".$pic[0]); //设置下载文件名
header("Content-Length:".$pic[3]); //设置下载大小

```

```
//输出下载信息  
readfile("./pic/".$pic[1]);
```

◆ functions.php

```
<?php  
//函数库文件  
  
/**  
 * 文件上传函数  
 * @param array $upfile 上传文件信息,如: $_FILES['pic'];  
 * @param string $path 上传文件保存路径,  
 * @param array $typelist 允许上传文件类型, 默认空数组表示不限  
 *          如: array("image/jpg","image/jpeg","image/gif","image/png"); //允许的上传文件类型  
 * @param int $maxsize 允许上传文件大小, 0 表示不限制  
 * @return array 返回数组, 其中下标 error(布尔值)表示是否成功、info 失败表示错误信息, 成功表示  
上传的文件名  
 */  
function uploadfile($upfile,$path,$typelist=array(),$maxsize=0){  
    //1. 定义一个返回结果  
    $res=array("error"=>false,"info"=>"");  
    //2.判断上传错误信息  
    if($upfile['error']>0){  
        switch($upfile['error']){  
            case 1: $info="上传的文件超过了 php.ini 中限制的值"; break;  
            case 2: $info="上传文件的大小超过了 HTML 表单中 MAX_FILE_SIZE 选项指定的值";  
        break;  
            case 3: $info="文件只有部分被上传"; break;  
            case 4: $info="没有文件上传"; break;  
            case 6: $info="找不到临时文件夹"; break;  
            case 7: $info="临时文件写入失败"; break;  
            default: $info="未知错误！ "; break;  
        }  
        $res['info']=$info;  
        return $res;  
    }  
  
    //3.判断上传文件类型(可选)  
    if(count($typelist)>0 && !in_array($upfile['type'],$typelist)){  
        $res['info']="上传文件类型错误: ".$upfile['type'];  
        return $res;  
    }  
  
    //4.判断上传文件大小(可选)
```

```

if($maxsize>0 && $upfile['size']>$maxsize){
    $res['info']="上传文件大小超出！ ";
    return $res;
}

//5.处理上传的文件名
$ext = pathinfo($upfile['name'],PATHINFO_EXTENSION);//获取上传文件的后缀名
do{ //随机一个文件名并判断是否已存在。
    $filename = time().rand(10000,99999).".". $ext;
}while(file_exists($path."/". $filename));

//6.判断并执行文件上传
//判断是否是有效的上传文件
if(is_uploaded_file($upfile['tmp_name'])){
    //执行文件上传处理，并判断是否成功
    if(move_uploaded_file($upfile['tmp_name'],$path."/". $filename)){
        $res['info']=$filename;
        $res['error']=true;
    }else{
        $res['info']="上传文件失败！ ";
    }
}else{
    $res['info']='无效上传文件！ ';
}
return $res;
}

/***
 * 图片等比缩放函数
 * @param string $picname 被缩放的图片名称。如: "a.jpg"
 * @param string $path 被缩放图片的存放路径。如: "uploads/images"目录
 * @param string $prix 缩放后图片名的前缀名。默认为: "s_"
 * @param int $maxwidth 缩放后的最大宽
 * @param int $maxheight 缩放后最大的高
 * @return 无返回值
 */

```

```

function updateImage($picname,$path,$prix="s_", $maxwidth=100,$maxheight=100){
    //1. 定义获取基本信息
    $path = rtrim($path,"/"); //去除后面多余的"/"
    $info = getimagesize($path."/". $picname); //获取图片文件的属性信息
    $width = $info[0]; //原图片的宽度
    $height = $info[1]; //原图片的高度

```

```
//2. 计算压缩后的尺寸
if(($maxwidth/$width)<($maxheight/$height)){
    $w=$maxwidth;//新图片的宽度
    $h=($maxwidth/$width)*$height;//新图片的高度
}else{
    $h=$maxheight;//新图片的宽度
    $w=($maxheight/$height)*$width;//新图片的高度
}

//3. 创建图像源
$newim =imagecreateTrueColor($w,$h); //新图片源
//根据原图片类型来创建图片源
switch($info[2]){
    case 1://gif 格式
        $srcim = imageCreateFromgif($path."/". $picname);
        break;
    case 2://jpeg 格式
        $srcim = imageCreateFromjpeg($path."/". $picname);
        break;
    case 3://png 格式
        $srcim = imageCreateFrompng($path."/". $picname);
        break;
    default:
        exit("无效的图片格式!");
        break;
}

//4. 执行缩放处理
imagecopyresized($newim,$srcim,0,0,0,0,$w,$h,$width,$height);

//5. 输出保存绘画
//header("Content-Type:".$info['mime']); //设置响应类型为图片格式
//根据原图片类型来保存新图片
switch($info[2]){
    case 1://gif 格式
        imagegif($newim,$path."/". $prix.$picname); //保存
        //imagegif($newim);//输出
        break;
    case 2://jpeg 格式
        imagejpeg($newim,$path."/". $prix.$picname);
        //imagejpeg($newim);
        break;
}
```

```
case 3://png 格式
    imagepng($newim,$path."/". $prix.$picname);
    //imagepng($newim);
    break;
}

//6. 销毁资源
imageDestroy($newim);
imageDestroy($srcim);
}
```

=====

文件上传和下载笔记

=====

一、 php.ini 的配置信息

file_uploads = On /Off 是否允许文件上传
upload_max_filesize=2M 上传的文件的最大大小
post_max_size = 8M POST 数据所允许的最大大小
upload_tmp_dir 上传文件放置的临时目录

注意配置： upload_max_filesize 的大小一定要小于 post_max_size 的配置大小。

二、（发送客户端）上传的 form 表单：

- 1、 表单必须是 post 提交
- 2、 上传的类型： enctype="multipart/form-data"
- 3、 上传使用的表单项

<input type="file" name=".." />
4.(可选)上传大小限制的表单隐藏域： MAX_FILE_SIZE,
<input type="hidden" name="MAX_FILE_SIZE" value="大小字节"/>
注意：此字段必须在文件输入字段之前（常放在 form 标签后面）

三、（接收服务器端）：

1. 使用\$_FILES 全局数组来接收上传信息

在每个上传的文件里,\$_FILES 中都会有 5 个属性：

- error: 上传的错误号： 0--4
 - 0: 表示没有发生任何错误。
 - 1: 上传的文件超过了 php.ini 中 upload_max_filesize 选项限制的值。
 - 2: 表示上传文件大小超出了 HTML 表单隐藏域属性的 MAX_FILE_SIZE 元素所指定的最大值。
 - 3: 表示文件只被部分上传。
 - 4: 表示没有上传任何文件。

```
6: 找不到临时文件夹  
7: 文件写入失败  
name: 上传的文件名  
size: 文件的大小  
type: 文件类型  
tmp_name: 临时文件  
  
2: is_uploaded_file() //是否是上传文件  
3: move_uploaded_file () //执行移动上传文件
```

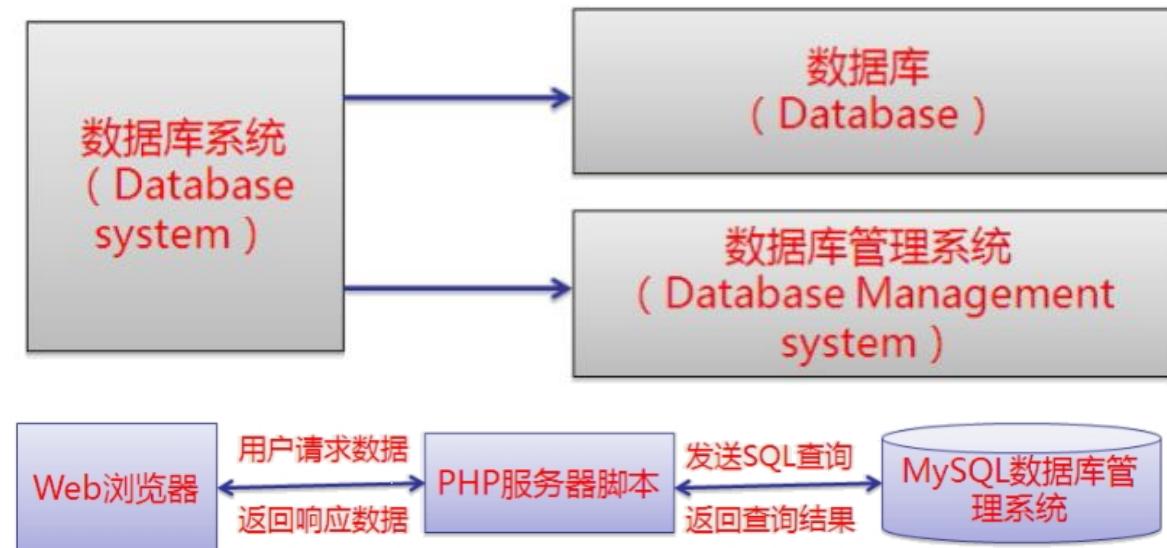
四、 下载设置:

```
header("Content-Type:类型"); //指定响应类型  
header("Content-Disposition:attachment;filename=文件名"); //**执行下载文件名  
header("Content-Length:文件大小");  
  
readfile("./uploads/".$picname); //读取并输出图片内容;
```

十四、MySQL 数据表的设计

1. 数据库的应用

数据库是计算机应用系统中的一种专门管理数据资源的系统。数据库就是一组经过计算机整理后的数据，储存在一个或者多个文件中，而管理这个数据库的软件就是数据库管理系统。



网站的主体架构未发生变化，而网站的内容信息会随时间发生变化

2. 数据表概念

数据表是数据库中的基本对象元素，以记录（行）和字段（列）组成的二维结构用于存储数据。数据表由表结构和表内容两部分组成，先建立表结构，然后才能输入数据。数据表结构设计主要包括

字段名称、字段类型和字段属性的设置。

通常情况下，同一个数据库中可以有多个数据表，但表名必须是唯一的，表中每一条记录描述了一个相关信息的集合，每一个字段必须为唯一的，每个字段都需要指定数据类型。

3.数据列四大数据类型

数值类数据列类型

字符串类数据列类型

日期和时间类数据列类型

NULL 值

3.1 数值类数据列类型

数据列类型	存储空间	说明	取值范围
TINYINT	1字节	非常小的整数	带符号值：-128~127 无符号值：0~255
SMALLINT	2字节	较小的整数	带符号值：-32768~32767 无符号值：0~65535
MEDIUMINT	3字节	中等大小的整数	带符号值：-8388608~8388607 无符号值：0~16777215
INT	4字节	标准整数	带符号 值：-2147483648~2147483647 无符号值：0~4294967295
BIGINT	8字节	大整数	带符号值：-2 ⁶³ ~-2 ⁶³ -1 无符号值：0~2 ⁶⁴ -1
FLOAT	4或8字节	单精度浮点数	最小非零值：+- 1.175494351E-38 最大非零值：+- 3.402823466E+38

3.2 数值类数据列类型

数据列类型	存储空间	说明	取值范围
DOUBLE	8字节	双精度浮点数	最小非零值：+- 2.225073E-308 最大非零值：+- 1.797693E+308
DECIMAL	自定义	以字符串形式表示的浮点数	取决于存储单元字节数

整型注意事项：

》 INT(3)、SMALLINT (3) 等整型后面的数字不会影响数值的存储范围，只会影响显示

》 整型后面的数字只有配合零填充的时候才有实际意义。

》 整型后面的数字可以省略

浮点型注意事项：

》浮点型后面的数字会将存入的数字四舍五入，例如：把一个 1.234 存入 FLOAT(6,1)数据列中，结果是 1.2,6 代表显示长度，1 代表小数位长度，会四舍五入。

3.3 字符串类数据列类型

字符串类型注意事项：

》CHAR 和 VARCHAR 类型的长度范围都在 0~255 之间

》在使用 CHAR 和 VARCHAR 类型时，当我们传入的实际的值的长度大于指定的长度，字符串会被截取至指定长度

》在使用 CHAR 类型时，如果我们传入的值的长度小于指定长度，实际长度会使用空格补至指定长度

》在使用 VARCHAR 类型时，如果我们传入的值的长度小于指定长度，实际长度即为传入字符串的长度，不会使用空格填补

》CHAR 要比 VARCHAR 效率更高，当占用空间较大

》BLOB 和 TEXT 类型是可以存放任意大数据的数据类型

》BLOB 区分大小写，TEXT 不区分大小写

》ENUM 和 SET 类型是特殊的的串类型，其列值必须从固定的串集中选择

》ENUM 只能选择其中一个值，SET 可以选择多个值

3.4 日期和时间类数据列类型

存储日期时，我们可以使用整型来进行存储时间戳，这样做便于我们进行日期的计算

3.5 数据字段属性

UNSIGNED

只能用于设置数值类型，不允许出现负数

最大存储长度会增加一倍

ZEROFILL

只能用于设置数值类型，在数值之前会自动用 0 补齐不足的位数

AUTO_INCREMENT

用于设置字段的自动增长属性，每增加一条记录，该字段的值会自动加 1

NULL 和 NOT NULL

默认为 NULL，即插入值时没有在此字段插入值，默认为 NULL 值，如果指定了 NOT NULL，则

必须在插入值时在此字段填入值

DEFAULT

可以通过此属性来指定一个默认值，如果没有在此列添加值，那么默认添加此值

NULL 值注意事项：

》NULL 意味着“没有值”或“未知值”

》可以测试某个值是否为 NULL

》不能对 NULL 值进行算术计算

》对 NULL 值进行算术运算，其结果还是 NULL

》0 或 NULL 都意味着假，其余值都意味着真

4. 创建索引

在 MySQL 中，主要有四类索引：

主键索引（PRIMARY KEY）

唯一索引（UNIQUE）

常规索引（INDEX）

全文索引（FULLTEXT）

4.1 主键索引

主键索引是关系数据库中最常见的索引类型，主要作用是确定数据表里一条特定的数据记录的位置。我们可以在字段后添加 PRIMARY KEY 来对字段设置为主键索引。

注意：

- 1.最好为每张表指定一个主键，但不是必须指定。
- 2.一个表只能指定一个主键，而且主键的值不能为空
- 3.主键可以有多个候选索引（例如 NOT NULL, AUTO_INCREMENT）

4.2 唯一索引

唯一索引与主键索引一样，都可以防止创建重复的值。但是，不同之处在于，每个数据表中只能有一个主键索引，但可以有多个唯一索引。我们使用关键字 UNIQUE 对字段定义为唯一索引。

4.3 常规索引

常规索引技术是关系数据查询中最重要的技术，如果要提升数据库的性能，索引优化是首先应该考虑的，因为它能使我们的数据库得到最大性能方面的提升。常规索引也存在缺点：

- 1.多占用磁盘空间
- 2.会减慢插入，删除和修改操作
- 3.需要按照索引列上排序格式执行

创建索引我们可以使用 INDEX 和 KEY 关键字随表一同创建。

4.4 全文索引

全文索引在 MySQL 中是一个 FULLTEXT 类型索引，但 FULLTEXT 索引只能用于 MyISAM 表，并且只可以在 CHAR、VARCHAR 或 TEXT 类型的列上创建，也允许创建在一个或多个数据列上。但是 FULLTEXT 是不支持中文全文索引的，所以我们将来会使用效率更高的全文索引引擎 Sphinx。

5. 数据表的类型及存储位置

MySQL 支持 MyISAM、InnoDB、HEAP、BOB、ARCHIVE、CSV 等多种数据表类型，在创建一个新 MySQL 数据表时，可以为它设置一个类型。

MyISAM 和 InnoDB 两种表类型最为重要：

- 1.MyISAM 数据表类型的特点是成熟、稳定和易于管理。
- 2.MyISAM 表类型会产生碎片空间，要经常使用 OPTIMIZE TABLE 命令去清理表空间
- 3.MyISAM 不支持事务处理，InnoDB 支持
- 4.MyISAM 不支持外键，InnoDB 支持
- 5.MyISAM 表类型的数据表效率更高
- 6.MyISAM 表类型的数据表会产生三个文件，InnoDB 表类型表默认只会产生一个文件。

6. 创建表

创建数据表之前，我们应该注意：

1. 创建数据库（如已存在则不需要创建）

2. 选择数据库

3. 在该数据库当中创建数据表

创建数据表需要注意：

1. 指定数据表的名称（数据表不能重名）

2. 指定该表的字段名称、字段数据类型、字段索引

3. 指定表类型和表默认字符集（可省略）

```
#!/php
//preg_grep -- 返回与模式匹配的数组单元

$a = array(
    'zhangTao@lampbrother.net',
    'tao.zhang.tc@163.com',
    'zhangan.sohu.com',
    'qwe@rty.u');

//匹配出上面数组中有效的Email地址

$list = preg_grep("/^[\w0-9_\.\\-]+@[\\w0-9\\-]+(\\.[\\w-z]+){1,2}$/i", $a);

print_r($list);

echo "<hr/>";

//将下面数组中有效日期全部匹配出来 (小括号有子存储功能)
$a = array("2013-08-22", "2012/12-12", "2014-09-20", "2011-02-20", '2013/06/25');
$list = preg_grep('/(\d{4})[\\-\\/]\d{2}[\\-\\/]\d{2}/', $a); //会出现分隔符不统一
//$list = preg_grep('/\d{4}([\\-\\/])\d{2}\1\d{2}/', $a); //\1 获取第一个子存储内容使保持一致
print_r($list);
//此处正则中的\1表示重复前面第一个括号的内容 (要求一样)。
```