注释

//这就是注释

/\*这也是注释\*/

扩展名.php

结构

<?php

/\*所有代码写在这个里边\*/

?>

定义变量

$name=10;

变量以 $ 符号开头，其后是变量的名称

变量名称必须以字母或下划线开头，不能以数字开头

echo name;

打印/输出

echo "<br>"

换行

后端编写的代码必须放在服务器的环境下运行

放在服务器对应的目录里，通过服务器运行

本机服务器目录：E:\Programs\wampserver\www

* $arr=array(1,3,5)

$arr=[1, 3, 5]

创建数组

$arr[1]

取出元素

$arr=array(3, 'hello', 4=>'me', 'legal', true)

则'legal'的索引是5，true的是6

* $obj=array("name"=>"wang", "age"=>18);

$obj=['name'=>'wang', 'age'=>43];

创建关联数组

$obj["name"]取出值

* unset(array[9])

删除

count($arr);

返回个数

遍历数组

foreach( 数组变量 as [键=>] 值){

}

$my\_array = array("Dog","Cat","Horse");

list($a, $b, $c) = $my\_array;

list在一次操作中给一组变量赋值。变量可以没有预先声明

each($arr)

返回一个关联数组。属性名1，value，值为value；属性名0，key，值为key。内部指针向后移动

Array ( [1] => Peter [value] => Peter [0] => 0 [key] => 0 )

while( list($key, $value)=each($arr)){

}

但是list和each一起用，each返回的数组不成立了。只返回一个key一个value，赋给list中的两个变量

current($arr) - 返回数组中的当前元素的值

key($arr) – 返回键

end($arr) - 将内部指针指向数组中的最后一个元素，并输出

next($arr) - 将内部指针指向数组中的下一个元素，并输出

prev($arr) - 将内部指针指向数组中的上一个元素，并输出

reset($arr) - 将内部指针指向数组中的第一个元素，并输出

用each前先reset()。不然出不来

| **函数** | **功能** |
| --- | --- |
| array\_shift | 弹出数组中的第一个元素 |
| array\_unshift | 在数组的开始处压入元素 |
| array\_push | 向数组的末尾处压入元素 |
| array\_pop | 弹出数组末尾的最后一个元素 |

|  |  |
| --- | --- |
| array\_combine() | 生成一个数组,用一个数组的值作为键名,另一个数组值作为值 |
| range() | 创建并返回一个包含指定范围的元素的数组。 |
| compact() | 创建一个由参数所带变量组成的数组 |
| array\_fill() | 用给定的值生成数组 |
| array\_chunk() | 把一个数组分割为新的数组块 |
| array\_merge() | 把两个或多个数组合并为一个数组 |
| array\_slice() | 在数组中根据条件取出一段值，并返回 |
| array\_diff() | 返回两个数组的差集数组 |
| array\_search() | 在数组中搜索给定的值，如果成功则返回相应的键名 |
| array\_splice() | 把数组中的一部分去掉并用其它值取代 |
| array\_sum() | 计算数组中所有值的和 |
| in\_array() | 检查数组中是否存在某个值 |
| array\_key\_exists() | 检查给定的键名或索引是否存在于数组中 |
| shuffle() | 将数组打乱,保留键值 |
| count() | 计算数组中的单元数目或对象中的属性个数 |
| array\_flip() | 返回一个键值反转后的数组 |
| array\_keys() | 返回数组所有的键,组成一个数组 |
| array\_values() | 返回数组中所有值，组成一个数组 |
| array\_reverse() | 返回一个元素顺序相反的数组 |
| array\_count\_values() | 统计数组中所有的值出现的次数 |
| array\_rand() | 从数组中随机抽取一个或多个元素,注意是键名 |
| array\_unique() | 删除重复值，返回剩余数组 |
| sort() | 按升序对给定数组的值排序,不保留键名 |
| rsort() | 对数组逆向排序,不保留键名 |
| asort() | 对数组排序,保持索引关系 |
| arsort() | 对数组逆向排序,保持索引关系 |
| ksort() | 按键名对数组排序 |
| krsort() | 将数组按照键逆向排序 |
| natsort() | 用自然顺序算法对数组中的元素排序 |
| natcasesort() | 自然排序,不区分大小写 |
| array\_filter() | 去掉数组中的空元素或者预定元素 |
| extract | 将键变为变量名，将值变为变量值 |

-> 对象成员访问符号

$this->$name = $value

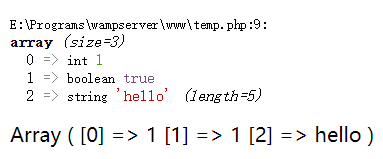
$this 当前类 $name属性

给当前类的属性赋值

打印数组或对象不能用echo

var\_dump($arr)返回一个变量的类型、长度和值,可以打印数组

print\_r($arr)



HTML中的表单Form标签可以发送请求

<form action="www.baidu.com" method="get/post">//通过get或post提交

<input type="text" name="username"><br>

<input type="password" name="userpwd"><br>

<input type="submit" value="提交"><br>

</form>

action规定当提交表单时向何处发送表单数据。

**Get请求**

请求的数据存储到URL后面

$\_GET

获取前端通过get提交的数据

print\_r($\_GET)

就能看到拿到的数据

echo $\_GET["username"]

echo $\_GET["userpwd"]

Post请求

将发送的数据放到请求头里边

获取

echo $\_POST["username"]

GET 和 POST 都创建数组（例如，array( key => value, key2 => value2, key3 => value3, ...)）。此数组包含键/值对，其中的键是表单控件的名称，而值是来自用户的输入数据。

GET 和 POST 被视作 $\_GET 和 $\_POST。它们是超全局变量，这意味着对它们的访问无需考虑作用域 - 无需任何特殊代码，您能够从任何函数、类或文件访问它们。

$\_GET 是通过 URL 参数传递到当前脚本的变量数组。

$\_POST 是通过 HTTP POST 传递到当前脚本的变量数组。

何时使用 GET

通过 GET 方法从表单发送的信息对任何人都是可见的（所有变量名和值都显示在 URL 中）。GET 对所发送信息的数量也有限制。限制在大于 2000 个字符。不过，由于变量显示在 URL 中，把页面添加到书签中也更为方便。

GET 可用于发送非敏感的数据。

何时使用 POST

通过 POST 方法从表单发送的信息对其他人是不可见的（所有名称/值会被嵌入 HTTP 请求的主体中），并且对所发送信息的数量也无限制。

提示：开发者偏爱 POST 来发送表单数据

<form action="a.php" enctype="multipart/form-data">

<input type="file">

</form>

上传文件

上传二进制文件，form标签必须添加enctype="multipart/form-data"

接收时是$\_FILES

获取上传文件的名称

获取上传文件的路径

$fileInfo=$\_FILES["upFile"];

$fileName=$fileInfo["name"]

move\_upload\_file($fileInfo["tmp\_name"], "./source".$fileName);

用 . 连接

$.ajax({

url: "p11.php",

type: "get",

data: "userName=linran&userPwd=123",

success: function(){

alert("成功后执行");

},

error: function(){

alert("失败后执行");

}

})

**数据库**

* 打开一个到 MySQL 服务器的新的连接：

$conn=mysqli\_connect("localhost","root","password，wamp的话填"" ", "数据库名", "数据库端口号默认3306");

* 检查连接

if (!$conn)

{

die("连接失败: " . mysqli\_connect\_errno());//返回上一次连接错误的错误代码，一个数字

die("连接错误: " . mysqli\_connect\_error()); //返回上一次连接错误的错误描述

}

* 修改连接的数据库

mysqli\_select\_db($conn,"project1");

* 设置字符集

mysqli\_set\_charset($conn, "utf8")

* 创建数据库

$sql = "CREATE DATABASE myDB";

if (mysqli\_query($conn, $sql)) {

echo "数据库创建成功";

} else {

echo "Error creating database: " . mysqli\_error($conn);

}

同理创建表

* 插入数据

$sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli\_query($conn, $sql)) {

echo "新记录插入成功";

} else {

echo "Error: " . $sql . "<br>" . mysqli\_error($conn);

}

* 插入多条记录

$sql = "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('John', 'Doe', 'john@example.com');";

$sql .= "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('Mary', 'Moe', 'mary@example.com');";

$sql .= "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('Julie', 'Dooley', 'julie@example.com')";

if (mysqli\_multi\_query($conn, $sql)) {

echo "新记录插入成功";

} else {

echo "Error: " . $sql . "<br>" . mysqli\_error($conn);

}

* 查找数据

$sql = "SELECT id, firstname, lastname FROM MyGuests";

$result = mysqli\_query($conn, $sql);

if (mysqli\_num\_rows($result) > 0) {

while($row = mysqli\_fetch\_assoc($result)) {

echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";

}

//释放结果集

mysqli\_free\_result($result);

} else {

echo "0 结果";

}

用$result存储结果，mysqli\_fetch\_assoc读取每一行，$row["attr"]访问数据

mysqli\_fetch\_array($result,resulttype);

resulttype 可选。可以取以下值

MYSQLI\_ASSOC返回关联数组

MYSQLI\_NUM返回索引数组

MYSQLI\_BOTH返回索引和关联

相当于使用以下函数

* mysqli\_fetch\_asso() 返回关联数组
* mysqli\_fetch\_row() 返回索引数组
* mysqli\_fetch\_object() 返回对象

取值：

$row["id"]

$row[0]

$row->id

$fieldinfo=mysqli\_fetch\_field\_direct($result,1);// 字段号介于 0 和 字段数-1 之间

echo $fieldinfo->name;

返回包含字段定义信息的对象。如果没有可用信息则返回 FALSE。该对象有下列属性:

name - 列名

orgname - 原始的列名（如果指定了别名）

table - 表名

orgtable - 原始的表名（如果指定了别名）

def - 该字段的默认值

max\_length - 字段的最大宽度

length - 在表定义中规定的字段宽度

charsetnr - 字段的字符集号

flags - 字段的位标志

type - 用于字段的数据类型

decimals - 整数字段，小数点后的位数

mysqli\_fetch\_field(result);

返回包含下一字段信息的对象

* 关闭连接

mysqli\_close($conn);

* 返回服务器信息

echo mysqli\_get\_server\_info($conn);

echo mysqli\_get\_server\_version($conn);

**json**

访问本地json文件

file\_get\_contents("j.txt");

PHP单引号及双引号均可以修饰字符串类型的数据，如果修饰的字符串中含有变量（例$name）；最大的区别是： 双引号会替换变量的值，而单引号会把它当做字符串输出。

<?php

$name="string";

echo "字符串" . '$name';

echo "字符串" ."$name";

?>

结果：

字符串$name//单引号不解析

字符串string//双引号解析变量

echo "$nameeeeee";//啥也不显示

echo "{$name}eeeee"//有大括号；返回stringeeeee

echo "hello, '$name' world"//双引号中间插单引号，解析

unset($num)

把$num销毁成null

empty($num)

$num为null或false时返回true

isset($n1, $n2, $n3);

n1, n2, n3中有一个null就返回false

gettype("this is a string")

获得变量类型，返回一个string

is\_int 是否为整型

is\_bool 是否为布尔

is\_float 是否是浮点

is\_string 是否是字符串

is\_array 是否是数组

is\_object 是否是对象

is\_null 是否为空

is\_resource 是否为资源

is\_scalar 是否为标量

is\_numeric 是否为数值类型

is\_callable 是否为函数

function f1(&$test1, $test2=10){

$test1 .= "world";

}

$t1=34;

$t2=&$t1;按引用传递，改变t2也改变t1，改变t1也改变t2

PHP函数名不区分大小写

可以有默认参数，有默认参数的必须在后面

* 传值，不会改变传进去的参数，传对象进去也不改变;&$test1传引用，改变原变量
* str .= "hello", 千万可不敢用成+=

function f2($test3){

if(is\_callable($test3)){

echo "test3 is a function";

}

}

function t3(){

}

f2('t3');

* 函数作为参数时必须放在单引号里，不然报错

$t1='f2';

function f2(){

echo 'fefafea';

}

$t1();

* 变量函数

$q=12;

function demo(){

$GLOBALS['h']="fjhiiaf";

$GLOBALS['q']=1;

static $count=1;

$count++;

}

demo();

echo $h;echo $q;

* 通过$GLOBALS在函数内访问、修改或创建全局变量
* static创建静态变量

function foo()

{

echo '调一下我才会执行定义函数bar的过程<br />';

function bar()

{

echo '在foo函数内部有个函数叫bar函数<br />';

}

}

* 函数foo内有一个bar内部函数，调用bar之前必须先调用一次foo
* 然后foo就不能再调用了。foo只能调用一次

include 'p1.php';

require 'testmysql.php';

include如果被包含的文件不存在报一条警告，require失败直接报错，不继续执行

时区

1884年在华盛顿召开国际经度会议规定将全球划分为24个时区

在中国采用首都北京所在地东八区的时间为全国统一使用时间

unix时间戳

从1970 年 1月1日零时开始经过的秒数。

$t=time();

获取当前unix时间戳

设置时区

define('TIME\_ZONE','Asia/Shanghai');

date\_default\_timezone\_set(TIME\_ZONE);

常用时区

Asia/Shanghai – 上海   
Asia/Chongqing – 重庆   
Asia/Urumqi – 乌鲁木齐   
Asia/Hong\_Kong – 香港   
Asia/Macao – 澳门   
Asia/Taipei – 台北   
Asia/Singapore – 新加坡

echo date('y-m-d h-i-s');

echo date('Y-M-D');

输出

180925 08-45-30

2018SepTue

| 字符 | 说明 | 返回值 |
| --- | --- | --- |
| d | 月份中的第几天，有前导零的2 位数字 | 01 到31 |
| D | 英文星期几，3个字母 | Mon到Sun |
| j | 月份中的第几天，没有前导零 | 1 到31 |
| I | 英文星期几 | Sunday到 Saturday |
| N | 1格式数字表示的星期 | 1（表示星期一）到7（表示星期天) |
| S | 每月天数后面的英文后缀，2个字符 | st，nd，rd或者th。可以和jg一起用 |
| w | 星期中的第几天，数字表示 | 0（表示星期天）到 6（表示星期六） |
| z | 一年中的第几天 | 0到366 |
| W | 年份中的第几周，每周从星期一开始 | 42（当年的第42周） |
| F | 月份，完整的文本格式 | January 到 December |
| m | 数字表示月份，有前导零 | 01 到 12 |
| M | 3个字母缩写表示的月份 | Jan 到Dec |
| n | 数字表示月份，没有前导零 | 1 到 12 |
| t | 给定月份所应有的天数 | 28 到 31 |
| L | 是否为闰年 | 如果是闰年为1，否则为o |
| o | 格式年份数字 | 例如2007 |
| Y | 4 位数字完整表示年份 | 例如1999或2008 |
| y | 2 位数字表示的年份 | 例如99或08 |
| a | 小写的上午和下午值 | am或pm |
| A | 大写的上午和下午值 | AM或PM |
| g | 小时，12小时格式，没有前导零 | 1到12 |
| G | 小时，24小时格式，没有前导零 | 0 到 23 |
| i | 有前导零的分钟数 | 00 到 59 |
| s | 秒数，有前导零 | 00到59 |
| e | 时区标识 | Asia/Shanghai或UTC |
| U | 从Unix纪元开始至今的秒数 | 长整型数字 |

getdate ([ int $timestamp = time()] )

函数的返回值是一个根据timestamp得到的包含有时间信息的数组（对象）。如果没有参数，则会返回当前的时间

| 键名 | 说明 | 返回值 |
| --- | --- | --- |
| secnods | 秒 | 数字0到 59 |
| minutes | 分钟 | 数字0到59 |
| hours | 小时 | 数字 0到 23 |
| mday | 月份中第几天 | 数字 1到 31 |
| wday | 星期中第几天 | 数字0（表示星期天）到6（表示星期六） |
| mon | 月份 | 数字 1 到 12 |
| year | 年 | 4 位数字表示的完整年份 |
| yday | 一年中第几天 | 数字0到365 |
| weekday | 星期几的英文 | Sunday到 Saturday |
| month | 月份的英文 | January 到 December |
| 0 | 自从Unix纪元开始的秒数 | 长整型数字 |

bool checkdate ( int $month , int $day , int $year )

日期验证函数

如果是有效的时间就返回真，如果不是有效的时间就返回假

<?php

var\_dump(checkdate(12, 31, 2018));

var\_dump(checkdate(2, 29, 2011));

?>

输出

bool(true)  
bool(false)

mktime(hour,minute,second,month,day,year,[is\_dst]);

mktime函数返回一个Unix时间戳

<?php

echo mktime(13 ,15 , 30, 8,18, 2008) ;

?>

输出1219036530

例如，下面的代码实现对时间的运算：

<?php

echo date("m-d-Y h:m:s")."\n";

echo date("m-d-Y h:m:s",mktime(10,15,35,date("m"),date("d"),date("Y")))."\n";

echo date("m-d-Y h:m:s",mktime(10,15,35,date("m"),date("d")-30,date("Y")))."\n";

?>

我们使用的时候还经常需要使用到另外一个函数：strtotime()。

它的语法格式如下：

int strtotime ( string $time [, int $now = time() ] )

它能将将英文文本的日期时间描述解析为 Unix 时间戳。

参数：

    1.传入一个字符串的时间

    2.可选参数为是否传入unix时间戳，如果不传则是当前的unix时间戳。

我们来实验一下，看看手册中提供的一些例子：

<?php

//now为现在的当前时间

echo strtotime("now")."<br />";

//2000年9月10日

echo strtotime("10 September 2000")."<br />";

//当前时间加一天

echo strtotime("+1 day")."<br />";

//当前时间加一周

echo strtotime("+1 week")."<br />";

//当前时间加一周2天4小时2秒

echo strtotime("+1 week 2 days 4 hours 2 seconds")."<br />";

//下一个星期四

echo strtotime("next Thursday")."<br />";

//上一个星期一

echo strtotime("last Monday")."<br />";

?>

通过上例发现，把一些时间通通加上了或者减去了英文中所表示的指定时间段。

| 函数名 | 描述 | 实例 |
| --- | --- | --- |
| trim() | 删除字符串两端的空格或其他预定义字符 | "$str = "\r\nHello World!\r\n"; echo trim($str); |
| rtrim() | 删除字符串右边的空格或其他预定义字符 | "$str = "Hello World!\n\n"; echo rtrim($str);" |
| chop() | rtrim()的别名 | 同上 |
| ltrim() | 删除字符串左边的空格或其他预定义字符 | "$str = "\r\nHello World!"; echo ltrim($str);" |
| dirname() | 回路径中的目录部分 | echo dirname("c:/testweb/home.php"); |
| str\_pad() | 把字符串填充为指定的长度 | $str = "Hello World"; echo str\_pad($str,20,"."); |
| str\_repeat() | 重复使用指定字符串 | echo str\_repeat(".",13); |
| str\_split() | 把字符串分割到数组中 | print\_r(str\_split("Hello")); |
| strrev() | 反转字符串 | echo strrev("Hello World!"); |
| wordwrap() | 按照指定长度对字符串进行折行处理 | "$str = ""An example on a long word is: Supercalifragulistic""; echo wordwrap($str,15);" |
| str\_shuffle() | 随机地打乱字符串中所有字符 | echo str\_shuffle("Hello World"); |
| parse\_str() | 将字符串解析成变量 | "parse\_str("id=23&name=John%20Adams",$myArray); print\_r($myArray);" |
| number\_format() | 通过千位分组来格式化数字 | "echo number\_format("1000000"); echo number\_format("1000000",2); echo number\_format("1000000",2,"","",""."");" |
| strtolower() | 字符串转为小写 | echo strtolower("Hello WORLD!"); |
| strtoupper() | 字符串转为大写 | echo strtoupper("Hello WORLD!"); |
| ucfirst() | 字符串首字母大写 | echo ucfirst("hello world"); |
| ucwords() | 字符串每个单词首字符转为大写 | echo ucwords("hello world"); |
| htmlentities() | 把字符转为HTML实体 | $str = ""John & 'Adams'""; echo htmlentities($str, ENT\_COMPAT); |
| htmlspecialchars() | 预定义字符转html编码 |  |
| nl2br() | \n转义为 标签 | echo nl2br("One line.\nAnother line."); |
| strip\_tags() | 剥去 HTML、XML 以及 PHP 的标签 | echo strip\_tags("Hello world!"); |
| addcslashes() | 在指定的字符前添加反斜线转义字符串中字符 | $str = ""Hello, my name is John Adams." echo $str; echo addcslashes($str,'m');" |
| stripcslashes() | 删除由addcslashes()添加的反斜线 | echo stripcslashes("Hello, \my na\me is Kai Ji\m."); |
| addslashes() | 指定预定义字符前添加反斜线 | $str = "Who's John Adams?";echo addslashes($str); |
| stripslashes() | 删除由addslashes()添加的转义字符 | echo stripslashes("Who\'s John Adams?"); |
| quotemeta() | 在字符串中某些预定义的字符前添加反斜线 | $str = "Hello world. (can you hear me?)"; echo quotemeta($str); |
| chr() | 从指定的 ASCII 值返回字符 | echo chr(052); |
| ord() | 返回字符串第一个字符的 ASCII值 | echo ord("hello"); |
| strcasecmp() | 不区分大小写比较两字符串 | echo strcasecmp("Hello world!","HELLO WORLD!"); |
| strcmp() | 区分大小写比较两字符串 |  |
| strncmp() | 比较字符串前n个字符,区分大小写 |  |
| strncasecmp() | 比较字符串前n个字符,不区分大小写 | int strncasecmp ( string $str1 , string $str2 , int $len ) |
| strnatcmp() | 自然顺序法比较字符串长度,区分大小写 | int strnatcmp ( string $str1 , string $str2 ) |
| strnatcasecmp() | 自然顺序法比较字符串长度,不区分大小写 | int strnatcasecmp ( string $str1 , string $str2 ) |
| chunk\_split() | 将字符串分成小块 | str chunk\_split(str $body[,int $len[,str $end]]) |
| strtok() | 切开字符串 | str strtok(str $str,str $token) |
| explode() | 使用一个字符串为标志分割另一个字符串，返回一个数组 | array explode(str $sep,str $str[,int $limit]) |
| implode() | 同join,将数组值用预订字符连接成字符串 | string implode ( string $glue , array $pieces ) |
| substr() | 截取字符串 | string substr ( string $string , int $start [, int $length ] ) |
| str\_replace() | 字符串替换操作,区分大小写 | mix str\_replace(mix $search,,mix $replace,mix $subject[,int &$num]) |
| str\_ireplace() | 字符串替换操作,不区分大小写 | mix str\_ireplace ( mix $search , mix $replace , mix $subject [, int &$count ] ) |
| substr\_count() | 统计一个字符串,在另一个字符串中出现次数 | int substr\_count ( string $haystack , string $needle [, int $offset = 0 [, int $length ]] ) |
| substr\_replace() | 替换字符串中某串为另一个字符串 | mixed substr\_replace ( mixed $string , string $replacement , int $start [, int $length ] ) |
| similar\_text() | 返回两字符串相同字符的数量 | int similar\_text(str $str1,str $str2) |
| strchr() | 返回一个字符串在另一个字符串中开始位置到结束的字符串 | string strstr ( string $str, string $needle , bool $before\_needle ) |
| strrchr() | 返回一个字符串在另一个字符串中最后一次出现位置开始到末尾的字符串 | string strrchr ( string $haystack , mixed $needle ) |
| stristr() | 返回一个字符串在另一个字符串中开始位置到结束的字符串，不区分大小写 | string stristr ( string $haystack , mixed $needle [, bool $before\_needle = false ] ) |
| strtr() | 转换字符串中的某些字符 | string strtr ( string $str , string $from , string $to ) |
| strpos() | 寻找字符串中某字符最先出现的位置 | int strpos ( string $haystack , mixed $needle [, int $offset = 0 ] ) |
| stripos() | 寻找字符串中某字符最先出现的位置,不区分大小写 | int stripos ( string $haystack , string $needle [, int $offset ] ) |
| strrpos() | 寻找某字符串中某字符最后出现的位置 | int strrpos ( string $haystack , string $needle [, int $offset = 0 ] ) |
| strripos() | 寻找某字符串中某字符最后出现的位置,不区分大小写 | int strripos ( string $haystack , string $needle [, int $offset ] ) |
| strspn() | 返回字符串中首次符合mask的子字符串长度 | int strspn ( string $str1 , string $str2 [, int $start [, int $length ]] ) |
| strcspn() | 返回字符串中不符合mask的字符串的长度 | int strcspn ( string $str1 , string $str2 [, int $start [, int $length ]] ) |
| str\_word\_count() | 统计字符串含有的单词数 | mix str\_word\_count(str $str,[]) |
| strlen() | 统计字符串长度 | int strlen(str $str) |
| count\_chars() | 统计字符串中所有字母出现次数(0..255) | mixed count\_chars ( string $string [, int $mode ] ) |
| md5() | 字符串md5编码 | $str = "Hello"; echo md5($str) |
| iconv |  |  |
| mb\_substr | 获取字符串的部分 | string mb\_substr ( string $str , int $start [, int $length = NULL [, string $encoding = mb\_internal\_encoding() ]] ) |
| mb\_http\_output | 设置/获取 HTTP 输出字符编码 | mixed mb\_http\_output ([ string $encoding = mb\_http\_output() ] ) |
| mb\_strlen | 获取字符串的长度 | mixed mb\_strlen ( string $str [, string $encoding = mb\_internal\_encoding() ] ) |
| iconv | 字符串按要求的字符编码来转换 | string iconv ( string $in\_charset , string $out\_charset , string $str ) |
| iconv\_substr | 截取字符串的部分 |  |
| iconv\_get\_encoding | 获取 iconv 扩展的内部配置变量 |  |
| mb\_substr\_count | 统计字符串出现的次数 |  |
| mb\_check\_encoding | 检查字符串在指定的编码里是否有效 |  |
| mb\_strrpos | 查找字符串在一个字符串中最后出现的位置 |  |
| mb\_split | 使用正则表达式分割多字节字符串 |  |
| parse\_url | 解释URL成为一个数组 |  |

**正则表达式**

int preg\_match ( string $正则 , string $字符串 [, array &$结果] )

| 原子 | 说明 |
| --- | --- |
| \d | 匹配一个0-9 |
| \D | 除了0-9以外的所有字符 |
| \w | a-zA-Z0-9\_ |
| \W | 除了0-9A-Za-z\_以外的所有字符 |
| \s | 匹配所有空白字符\n \t \r 空格 |
| \S | 匹配所有非空白字符 |
| [ ] | 指定范围的原子 |

| 元字符 | 功能说明 |
| --- | --- |
| \* | 是代表匹配前面的一个原子，匹配0次或者任意多次前面的字符。 |
| + | 匹配一次或多前前面的一个字符 |
| ? | 前面的字符可有可无【可选】 有或没有 |
| . | 更标准一些应该把点算作原子。匹配除了\n以外的所有字符 |
| document/2015-09-04/55e92c421434d | 或者。注：它的优先级最低了。 |
| ^ | 必须要以抑扬符之后的字符串开始 |
| $ | 必须要以$之前的字符结尾 |
| \b | 词边界 |
| \B | 非边界 |
| {m} | 有且只能出现m次 |
| {n,m} | 可以出现n到m次 |
| {m,} | 至少m次，最大次数不限制 |
| () | 改变优先级或者将某个字符串视为一个整体，匹配到的数据取出来也可以使用它 |

**文件**操作

readfile("c:\\boot.ini");

直接读一个文件，输出

$filestring = file\_get\_contents("hello.txt");

读txt内容，返回一个字符串，效率比fread高

$fp = fopen('NoAlike.txt', "r"); //返回一个resource

$contents = fread($fp, 1024); //返回一个string，规定要读取的最大字节数。文件指针向后移动

fclose($fp);

| 模式 | 说明 |
| --- | --- |
| r | 只读方式打开，将文件指针指向文件头。 |
| r+ | 读写方式打开，将文件指针指向文件头。 |
| w | 写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建 |
| w+ | 读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建 |
| a | 写入方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建 |
| a+ | 读写方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之 |
| x | 创建并以写入方式打开，将文件指针指向文件头。如果文件已存在，则 fopen() 调用失败并返回 FALSE，并生成一条 E\_WARNING 级别的错误信息。如果文件不存在则尝试创建 |
| x+ | 创建并以读写方式打开，将文件指针指向文件头。如果文件已存在，则 fopen() 调用失败并返回 FALSE，并生成一条 E\_WARNING 级别的错误信息。如果文件不存在则尝试创建 |

| 模式 | 说明 |
| --- | --- |
| t | windows下将\n转为\r\n |
| b | 二进制打开模式 |

Windows 下提供了一个文本转换标记（'t'）可以透明地将 \n 转换为 \r\n。  
'b' 来强制使用二进制模式。

要使用这些标记，必须作为mode 参数的最后一个字符。

例如fopen($fpath, 'rb')

**错误处理**

配置文件路径"E:\Programs\wampserver\bin\php\php7.2.4\php.ini"

display\_errors = On

是否将错误信息输出到网页

修改ini后需要重启服务器

ini\_set('display\_errors' , 0 );

在当前php代码中设置

ini\_get('display\_errors')

获取

| **错误类型** | **说明** |
| --- | --- |
| E\_ERROR | 错误，文件直接中断 |
| E\_WARNING | 警告，问题比较严重。但是还会继续向下运行 |
| E\_NOTICE | 提示，有些小问题不会影响到程序。常发生在项目未定义 |
| E\_PARSE | 编译时语法解析错误。解析错误仅仅由分析器产生。 |
| E\_ALL | 所有的错误 |
| E\_STRICT | 启用PHP对代码的修改建议，以确保代码具有最佳的互操作性和向前兼容性。 |
| E\_DEPRECATED | 启用后将会对在未来版本中可能无法正常工作的代码给出警告。 |

error\_reporting 报告错误类型

1.在php.ini中修改error\_reporting参数。若error\_reporting参数设置为0。整个PHP引擎发错误均不会显示、输出、记录。日志记录中，也不会记录。

显示所有错误

error\_reporting = E\_ALL

显示所有错误但排除提示

error\_reporting = E\_ALL & ~ E\_NOTICE

显示所有错误，但排除提示、兼容性和未来兼容性

error\_reporting ＝ E\_ALL & ~E\_NOTICE & ~E\_STRICT & ~E\_DEPRECATED

2.在运行的xxxx.php文件中开始处，我们可以使用error\_reporting()函数达到目标。

error\_reporting(0); //关闭所有的错误显示

error\_reporting(E\_ALL); //显示所有错误

error\_reporting(E\_ALL & ~ E\_NOTICE); //显示所有错误，但不显示提示

//读取一个不存在的adsaf.txt文件

@$fp = fopen('adsaf.txt','r'); // @符单行不显示错误//请不用或者少用@

错误日志

| **参数** | **配置项** | **说明** |
| --- | --- | --- |
| log\_errors | on/off | 是否开启日志记录 |
| log\_errors\_max\_len | 整型，默认1024 | 单行错误最大记录长度 |
| error\_log | syslog或者指定路径 | 错误日志记录在什么地方 |

<?php

//自定义错误处理函数

function customError($errno, $errstr, $errfile, $errline) {

//输出错误消息

echo "<b>Custom error:</b> [$errno] $errstr<br />";

//输出错误文件和错误行

echo "Error on line $errline in $errfile<br />";

echo "Ending Script";

//中止程序运行

exit;

}

//使用set\_error\_handler 绑定用户自定义函数

set\_error\_handler("customError");

$test=2;

//触发自定义错误

if ($test > 1) {

trigger\_error("A custom error has been triggered");

}

?>

phpinfo()

输出 PHP 当前状态的大量信息，包含了 PHP 编译选项、启用的扩展、PHP 版本、服务器信息和环境变量（如果编译为一个模块的话）、PHP环境变量、操作系统版本信息、path 变量、配置选项的本地值和主值、HTTP 头和PHP授权信息(License)

因为每个系统安装得有所不同，phpinfo() 常用于在系统上检查配置设置和预定义变量

**Cookie**

cookie 常用于识别用户。

* setcookie() 函数必须位于 <html> 标签之前。

setcookie(name, value, expire, path, domain);

创建名为 "user" 的 cookie，并为它赋值 "runoob"。规定了此 cookie 在一小时后过期：

<?php

setcookie("user", "runoob", time()+3600);

?>

<html>

.....

注释：在发送 cookie 时，cookie 的值会自动进行 URL 编码，在取回时进行自动解码。（为防止 URL 编码，请使用 setrawcookie() 取而代之。）

* $\_COOKIE取回 cookie 的值。

echo $\_COOKIE["user"];

print\_r($\_COOKIE);

isset() 函数确认是否已设置了 cookie

if (isset($\_COOKIE["user"]))

echo "欢迎 " . $\_COOKIE["user"] . "!<br>";

else

echo "普通访客!<br>";

* 删除 Cookie

删除 cookie 时，使过期日期变更为过去的时间点。

setcookie("user", "", time()-3600);

PHP **Session**

session 会话

Session是一种服务器端技术。服务器可以为每个用户的浏览器创建一个为其独享的session文件。每个浏览器有一份session

Session 的工作机制是：为每个访客创建一个唯一的 id (UID)，并基于这个 UID 来存储变量。UID 存储在 cookie 中，或者通过 URL 进行传导。

* session\_start()函数必须位于 <html> 标签之前：

session\_start();

为用户会话分配一个 UID，初始化session

* 存储、获取 Session 变量

使用 PHP $\_SESSION 变量：

<?php

session\_start();

$\_SESSION['views']=1;

?>

<html>

<head>

<meta charset="utf-8">

<title>菜鸟教程(runoob.com)</title>

</head>

<body>

<?php

echo "浏览量：". $\_SESSION['views'];

var\_dump($\_SESSION);

?>

</body>

</html>

输出：

浏览量：1

* 销毁 Session

1.unset() 函数用于释放指定的 session 变量：

session\_start();

if(isset($\_SESSION['views']))

{

unset($\_SESSION['views']);

}

2.session\_destroy();

session\_destroy() 将重置 session，您将失去所有已存储的 session 数据。

* Session数据默认存在时间是1440s（24分钟），可以在php.ini中修改， session.gc\_maxlifetime = 1440
* Session文件的存放路径是可以修改的，可以通过修改php.ini改变sesion文件存放路径，本机session.save\_path ="E:/Programs/wampserver/tmp"

**PHP定界符**

定义格式化的大文本，即文本中不需要转义字符

* heredoc双引号定界符

$n="abcdef";

$str=<<<"ABC"

这里，<br>

想输什么输什么<br>

除了换行需要写成<br>

里边的变量将会被解析

比如$n

ABC;

echo $str;

ABC即定界符。可以自由定义，放在双引号中

* nowdoc单引号定界符

$str=<<<'ABC'

这里面的变量不解析

ABC;

echo $str;

pathinfo('1.txt');

返回

array (size=4)

'dirname' => string '.' (length=1)

'basename' => string '1.txt' (length=5)

'extension' => string 'txt' (length=3)

'filename' => string '1' (length=1)

pathinfo()返回一个关联数组，包括这个文件的路径信息

$data=(pathinfo('1.txt'));

echo $data['extension']

获取文件后缀

* URL编码

$url=urlencode('Linran的 个人主页space');

$password=md5(123123);//加密密码

echo "index.php?user=$url&password=$password";

输出index.php?user=Linran%E7%9A%84+%E4%B8%AA%E4%BA%BA%E4%B8%BB%E9%A1%B5space&password=4297f44b13955235245b2497399d7a93

中文被换成了%+16进制数字，一个汉字三个百分号3个两位16进制数字

* 空格变成加号
* ! @ # $ % ^ & \* ( ) [ ] { } ; , . / < > ?

分别变成

%21+%40+%23+%24+%25+%5E+%26+%2A+%28+%29+%5B+%5D+%7B+%7D+%3B+%2C+.+%2F+%3C+%3E+%3F

* 解码

echo urldecode('Linran%E7%9A%84+%E4%B8%AA%E4%BA%BA%E4%B8%BB%E9%A1%B5');

输出 Linran的 个人主页