PHP

#### 详细参考文档：<http://php.net/manual/zh/index.php>

1. 起步

PHP 是什么？

PHP 写在哪？

PHP 能做啥？ 超文本标记是用普通文本描述富文本的一种式

PHP（PHP: Hypertext Preprocessor）（递归缩写PHP中的开头的P是取自PHP:中的P）是一种被广泛应用的脚本语言，它可以被嵌入到 HTML中，尤其适合做动态网站开发开发。是弱类型语言，嵌在html页面中的

我们接下来会在 PHP 中看到的许多代码特性和其他编程语言类似，例如：变量、函数、循环，等等。 代码语法看 起来不同，但是在概念上是基本类似的。

我们使用 PHP 的目的就是能让静态网页变成动态网页，能称之为动态网页的核心就是让 HTML 上的内容不再被写 死，而是通过在 HTML 中嵌入一段可以在服务端执行的代码，从而达到动态网页的目标。

例如：我们需要有一个网页，这个网页每次打开都可显示当前的年月日，如果采用 HTML 处理：

* 1. <!DOCTYPE html>
  2. <html lang="en">
  3. <head>
  4. <meta charset="UTF‐8">
  5. <title>当前日期</title>
  6. </head>
  7. <body>

8 <h1>2020‐01‐01</h1>

1. </body>
2. </html>

#### 我们必须每天到服务器上修改这个网页，从而让它保持显示最新日期，但是有了 PHP 这种能够在服务端执行的脚

本语言就可以很轻松实现：

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF‐8">
5. <title>当前日期</title>
6. </head>
7. <body>

8 <h1><?php echo date('Y‐m‐d'); ?></h1>

1. </body>
2. </html>

#### 从以上这个最最简单的基础案例就能看出：PHP 无外乎为了可以在网页中动态输出最新内容的一种技术手段。

历史使人明智：<http://php.net/manual/zh/history.php.php>

* 1. PHP 标记

<http://php.net/manual/zh/language.basic-syntax.phpmode.php>

<?php 可以让代码进入“PHP 模式”

#### ?> 可以让代码退出“PHP 模式”

还有一种标记就是<**script** language="php"></**script**>

* + 1. <!DOCTYPE html>
    2. <html lang="en">
    3. <head>
    4. <meta charset="UTF‐8">

5 <title>这是一个包含 PHP 脚本的网页</title>

1. </head>
2. <body>

8 <h1>这是一个包含 PHP 脚本的网页</h1>

9 <p>这里原封不动的输出</p>

10

11 <?php

12 // 这里是 PHP 代码，必须满足 PHP 语法

1. $foo = 'bar';
2. echo $foo;
3. ?>

16

17 <p>这里也不变</p>

18

19 <p><?php echo '<b>这还是 PHP 输出的</b>'; ?></p>

1. </body>
2. </html>

#### 类似于在 HTML 中使用 JavaScript，但是不同的是 JavaScript 运行在客户端，而 PHP 运行在服务端。

只有处于 **PHP** 标记内部的代码才是 **PHP** 代码，**PHP** 标记以外都原封不动。

1.1.1. 省略结束标记

#### 如果 PHP 代码段处于整个文件的末尾，建议（必须）删除结束标记，这样不会有额外的空行产生。

意思就是：如果这个代码里面后面不会有html页面的输出的话，或者说后面不会有原封不动的内容输出的话

?>这个结束标记就可以不用写了。

要是写了的话，后面有很多空行的话，会占字节，不写的话，就不会占字节

1.2. 输出内容方式

#### echo：

注意：echo只能打印简单的数据，像字符串或者数字或者布尔值等，不能打印数组或者对象

echo false结果什么也没有

1 <?php

2 // echo 是 PHP 中的一个特殊的“指令”，

3 // 不一定需要像函数那样通过 `()` 去使用

4 // 注意：echo 后面紧跟着一个空格

1. echo 'hello php';
2. echo 'hello', 'world';
3. // => `helloworld`

#### print：

1 <?php

2 // print 与 echo 唯一区别就是只能有一个参数，它也是指令性的一个东西，不是一个函数

1. print 'hello php';
2. // print 'hello', 'world';
3. // => Parse error: syntax error ...

#### var\_dump：

一般调试的时候用于输出数据以及数据类型

1 <?php

2 // var\_dump 是一个函数，必须跟上 () 调用

3 // 可以将数据以及数据的类型打印为特定格式

1. var\_dump('hello php');
2. // => 'string(9) "hello php"'

例1：

1. <?php
2. $message = 'hello';
3. echo $message;
4. print 'value1';
5. var\_dump(false);
6. var\_dump(array(
7. '1','2'
8. ));

还有一些输出函数（可以通过查手册自学，用到再说），例如： exit() / print\_r() 等等

注意：在浏览器中看php文件时，页面渲染的不一定是准确的，要看网页源码，那个才是正确的

## 与 HTML 混编

#### 普通嵌入

1 <p><?php echo 'hello'; ?></p>

#### 语句混编

1 <?php if ($age >= 18) { ?>

2 <p>成年人</p>

3 <?php } else { ?>

4 <p>小朋友</p>

5 <?php } ?>

#### 更常见的用法：

1 <?php if ($age > 18): ?>

2 <p>成年人</p>

3 <?php else: ?>

4 <p>小朋友</p>

1. <?php endif ?>

例1：

if(true):

echo "hello";

else:

echo "world";

endif;

## 注释

#### 你可以在代码中添加注释，从而增强我们代码的可阅读性。PHP 中注释有两种方式（与 JavaScript 相同）：

单行注释

1 <?php

2 // 这是一条单行注释

3 # 井号也可以做注释（不要用，有点生僻）

4 $foo = 'hello';

#### 多行注释

1 <?php

2 /\*

3 ......

4 这里可以添加任意行数的注释内容

5 ......

6 \*/

7 $foo = 'hello';

# 语法

#### 编程语言常见的语法

变量 —— 用于临时存放数据的容器，用$声明，例如：$foo = 'hello';

顺序结构 —— 先干什么再干什么

分支结构 —— 如果怎样就怎样否则怎样 循环结构 —— 不断的做某件相同的事

函数 —— 提前设计好一件事怎么干，然后想什么时候干就什么时候干

运算符 —— 数学运算和字符串拼接

字面量 —— 在代码中用某些字符组成，能够表达一个具体的值 这些字符之间表示数据的方式叫做字面量

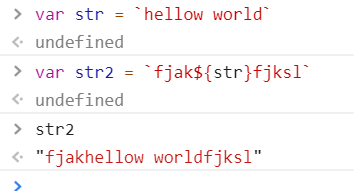
补充：函数在这里和在js中区别就是 作用域的问题，还有就是这里函数不区分大小写 ，而且PHP中很少出现函数套函数的情况出现

PHP 也是另外种编程语言，作为另外一种编程语言，PHP 也具备着绝大多数语言都有的特点，例如变量、条件分支、循环、函数等等，不同的是每个语言都会有自己的语法规定。这里不用太过担心，这些语法规定与之前学习的 编程语言大同小异，对我们来说不会有太大的压力。

着重点：

1. 变量

2. 双引号字符串和单引号字的那个键



3. 指令式的语法

4. foreach

5. 函数作用域问题

6. 字符串拼接

* 1. 变量

变量是编程语言中临时存放数据的容器。

PHP 中申明一个变量是用一个美元符号后面跟变量名来表示。变量名同样是区分大小写的。

PHP 中变量无需声明类型，变量的类型根据值的类型来推断。

1 <?php

2 $foo; // 申明一个变量，变量名为 `foo`，未对其进行赋值

3 $bar = 'baz'; // 申明一个变量，将一个值为 `baz` 的字符串赋值给它

4 echo $foo; // 输出一个变量名为 `foo` 的变量

5 fn($bar); // 将一个变量名为 `foo` 的变量作为 `fn` 的实参传递

* + 1. 数据类型

#### 常见的 PHP 数据类型与 JavaScript 基本一致：

string（字符串）

integer（整型）—— 只能存整数

ﬂoat（浮点型）—— 可以存带小数位的数字

boolean（布尔型） array（数组）

object（对象）

NULL（空） Resource（资源类型）

Callback / Callable（回调或者叫可调用类型）

字符串

PHP 有多种创建字符串的方式：单引号、双引号等。

单引号字符串

不支持特殊的转义符号，例如 \n

如果要表示一个单引号字符内容，可以通过 \' 表达 如果要表示一个反斜线字符内容，可以通过 \\ 表达

双引号字符串

支持转义符号

支持变量解析

1 <?php

2 // ====== 单引号 ======

1. echo 'hello\nworld';
2. // => `hello\nworld`
3. echo 'I\'m a better man';
4. // => `I'm a better man`
5. echo 'OS path: C:\\Windows';
6. // => `OS path: C:\Windows`

9

10 // ====== 双引号 ======

1. echo "hello\nworld";
2. // => `hello
3. // world`
4. $name = 'zce';
5. echo "hello $name";
6. // => `hello zce`

#### 字符串函数

<http://php.net/manual/zh/ref.strings.php> <http://www.w3school.com.cn/php/php_string.asp>

数组

PHP 中数组可以分为两类：

索引数组

与 JavaScript 中的数组基本一致

1 <?php

2 // 定义一个索引数组

3 $arr = array(1, 2, 3, 4, 5);

4 var\_dump($arr);

5

6 // PHP 5.4 以后定义的方式可以用 `[]`

7 $arr2 = [1, 2, 3, 4, 5];

8 var\_dump($arr2);

#### 关联数组

有点类似于 JavaScript中的对象

1 <?php

2 // 注意：键只能是`integer`或者`string`

1. $arr = array('key1' => 'value1', 'key2' => 'value2');
2. var\_dump($arr);

5

6 // PHP 5.4 以后定义的方式可以用 `[]`

7 $arr2 = ['key1' => 'value1', 'key2' => 'value2'];

8 var\_dump($arr2);

### 数据类型转换

#### 参考：<http://php.net/manual/zh/language.types.type-juggling.php>

1 <?php

2 $str = '132';

3 // 将一个内容为数字的字符串强制转换为一个整形的数字

4 $num = (int)$str;

5 // 将一个数字强制转换为布尔值

6 $flag = (bool)$num;

* 1. 运算符

#### 数学运算符，逻辑运算符与 JavaScript 基本一致，无额外特殊情况。

字符串连接（拼接）采用的是比较特殊的 .

1. <?php
2. $name = 'zce';

3 // 拼接 `hey ` 和 `zce`

4 $message = 'hey ' . $name;

5 // 相当于 +=

6 $foo .= $message

## 语句

#### 分号分割

if、switch、while、for、 foreach、function......

* 1. 流程控制

顺序结构

分支结构

if ... else switch ... case

循环结构

for

while

foreach --- 专门用来遍历数组

1. <?php
2. $arr = array('name' => 'zhangsan', 'age' => '18');

3

1. foreach ($arr as $key => $value) {
2. echo $key . ' ' . $value;
3. }

#### 指令式的 if、for、foreach、while 单独掌握

1 <?php

2 // 指令式就是将开始 { 换成 : 结束 } 换成 endif;

3 if ($i > 0) :

1. echo 'ok'
2. endif;

6

7 // for foreach while 也是一样

8 for ($i = 0; $i < 10; $i++) :

9 echo $i;

10 endfor;

## 函数

#### 定义与使用函数的方式与 JavaScript 相同：

1 <?php

2 // 函数名不区分大小写

1. function foo ($name, $title) {
2. echo "$name ($title)";
3. }

6

7 // 调用

8 foo('zce', 'UFO');

9 Foo('zgd', 'SBO'); // 大小写不区分

#### 注意：使用方式有点差异（函数名不区分大小写），但是不要这么搞！！！

建议在 PHP 中采用下划线式（snake\_case）做命名规则，不管是函数还是变量

1. 特性

3.1. 变量作用域

#### 关于变量作用域这一点，PHP 与绝大多数语言也都不同：默认函数内不能访问函数所在作用域的成员。 在 JavaScript 中，我们可以在函数作用域中使用父级作用域中的成员：

1 var top = 'top variable'

2

1. function foo () {
2. var sub = 'sub variable'

5

1. console.log(top)
2. // => `top variable`

8

9 function bar () {

1. console.log(top)
2. // => `top variable`
3. console.log(sub)
4. // => `sub variable`
5. }

15

1. bar()
2. }

18

19 foo()

#### 而在 PHP 中：

1. <?php
2. $top = 'top variable';

3

1. function foo () {
2. $sub = 'sub variable';

6

7 echo $top;

8 // => 无法拿到

9

1. function bar () {
2. echo $top;

12 // => 无法拿到

13

14 echo $sub;

15 // => 无法拿到

16 }

17

1. bar();
2. }

20

21 foo();

如果需要访问全局变量，可以通过 global 关键字声明：

1. <?php
2. $top = 'top variable';

3

4 function foo () {

5 // 声明在当前作用域中获取全局作用域中的 `$top`

6 global $top;

7

8 $sub = 'sub variable';

9

1. echo $top;
2. // => `top variable`

12

13 function bar () {

14 // 声明在当前作用域中获取全局作用域中的 `$top` 和 `$bar`

15 global $top, $bar;

16

1. echo $top;
2. // => `top variable`

19

20 echo $sub;

21 // => 任然无法拿到，因为 `$sub` 不再全局范围，而是在 `foo` 函数中定义

22 }

23

1. bar();
2. }

26

27 foo();

注意：在php中，\n这个换行操作符是对源码进行操作，在浏览器中是无作用的；而<br>这个标签在浏览器中起换行作用，在源码中会出现这个标签，但是在源码中不换行。

3.2. 超全局变量

#### <http://www.w3school.com.cn/php/php_superglobals.asp>

PHP 中的许多预定义变量都是“超全局的”，这意味着它们在一个脚本的全部作用域中都可用。在函数或方法中无需

执行 global $variable; 就可以访问它们。 这些超全局变量是：

$GLOBALS — 引用全局作用域中可用的全部变量

$\_SERVER — 获取服务端相关信息

$\_REQUEST — 获取提交参数

$\_POST — 获取 POST 提交参数

$\_GET — 获取 GET 提交参数

$\_FILES — 获取上传文件

$\_ENV — 操作环境变量

$\_COOKIE — 操作 Cookie

$\_SESSION — 操作 Session

本节会介绍一些超全局变量，并会在稍后的章节讲解其他的超全局变量。

3.2.1. $GLOBALS

#### $GLOBALS 这种全局变量用于在 PHP 脚本中的任意位置访问全局变量（从函数或方法中均可）。 PHP 在名为 $GLOBALS[index] 的数组中存储了所有全局变量。变量的名字就是数组的键。 下面的例子展示了如何使用超级全局变量 $GLOBALS：

1 <?php

2 $x = 75;

3 $y = 25;

4

1. function foo () {
2. $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
3. }

8

1. foo();
2. echo $z;

11 // => 100

3.3. 常量定义与使用

#### 常量跟变量一样也是一个数据容器，但是不同的是一旦申明过后就不允许被修改。

* + 1. 定义常量

1 <?php

2 // 定义常量使用的是内置的 `define` 函数

3 // 第一个参数是常量的名称，建议采用全大写字母命名，多个单词下划线分隔

4 // 第二个参数是常量中存放的数据，可以是任意类型

5 // 第三个参数是常量名称是否区不分大小写，默认 false 区分大小写

1. define('SYSTEM\_NAME', '阿里百秀');
2. define('SYSTEM\_ENABLE', true);
   * 1. 使用常量

1 <?php

2 // 直接通过常量的名称访问常量

3 // 与变量不同的是不需要用 $

1. echo SYSTEM\_NAME;
2. echo SYSTEM\_ENABLE;

3.4. 载入其他文件

#### 通常情况下，当一个文件中的代码过长，自然会想到要拆分到多个文件中。随着开发经验的积累，慢慢的会发现， 除了文件过程应该拆分文件，更应该做的事情是根据用途去划分。

不管你是怎样想的，核心问题都是一样：怎么将代码拆分到多个文件中？

PHP 中引入其他 PHP 文件有四种方式：

require require\_once include include\_once

四种方式的对比：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **require** | **require\_once** | **include** | **include\_once** |
| 被载入文件如果不存在是否影响继续运行 | Y | Y | N | N |
| 多次调用是否会重复执行被载入的文件 | Y | N | Y | N |

总结来说：

横向分为两类：require 和 include 两种，区别在于 require 会因为载入文件不存在而停止当前文件执行，而

include 不会。

纵向分为两类：xxx 和 xxx\_once，区别在于代码中每使用一次 xxx 就执行一次载入的文件，而 xxx\_once 只 会在第一次使用是执行。

使用层面：

include 一般用于载入公共文件，这个文件的存在与否不能影响程序后面的运行，（一般用于载入部分html页面）

require 用于载入不可缺失的文件 至于是否采用一次载入（用于载入必要的代码或者配置文件）

（once）这种方式取决于被载入的文件

1. 常用 API

（Application Programming Interface）应用程序编程接口

接口都是提供某种特定能力的事物，特点是有输入有输出，而在我们开发时用到的接口称之为API（应用程序编程接口）任何编程语言本身并没有太多的能力，具体的能力大多数都来源于 API。

PHP 的能力来源于它有 1000+ 内置函数，不是每一个函数都默认直接可以使用，有一些需要安装或者启用额外 的"插件" 扩展

接口的大致理解：有输入，有输出

学任何一个语言切记一定不要把精力放在学习API上面，要放在学语法上面，因为语法上的东西是可以学完的，API是学不完的，把常用的记住，然后学会掌握共性的东西，就像php中的API，就像strpos，我们只要关注我传进去什么参数，它返回给我什么就可以了，不用都记下来，学不完的

4.1. 字符串处理

#### 宽字符集需要开启 php\_mbstring 扩展

* + 1. 开启 PHP 扩展
       1. 将PHP目录中的 php.ini-development 复制一个 修改为 php.ini
       2. 修改扩展文件所在目录 extension\_dir

3. 修改文件中的部分选项（; 是注释符）

4. 在 Apache 配置文件中申明一下 php.ini 的所在目录

补充：

1.记住phpinfo();可以查看插件是否运行

2.碰到的问题：一个虚拟主机暂时不知什么错误导致apache不能运行

利用cmd找到apache中bin目录，利用httpd -t查看具体错误

* + - 1. interactive:交互的意思,shell正常的意思是外壳，，但是在我们这里的意思是脚本，或者命令的一种
      2. R E P L:

R:read

E:evaluation

P:print

L:loop

* + - 1. 简易运行php，在cmd中找到php路径C:Develop\php 然后php -a运行就可以了

在运行时ctrl+shift+C回到php文件目录

4.1.2. 字符串处理函数(都不改变原字符串)

#### 字符串截取：

string substr ( string $string , int $start [, int $length ] )

string mb\_substr ( string $str , int $start [, int $length = NULL [, string $encoding =

mb\_internal\_encoding() ]] )

#### 字符串长度

int strlen ( string $string )

mixed mb\_strlen ( string $str [, string $encoding = mb\_internal\_encoding() ] )

#### 大小写转换：

string strtolower ( string $string )

string strtoupper ( string $string )

#### 去除首尾空白字符

string trim ( string $str [, string $character\_mask = " \t\n\r\0\x0B" ] )

string ltrim ( string $str [, string $character\_mask ] )

string rtrim ( string $str [, string $character\_mask ] )

#### 查找字符串中某些字符首次出现位置

mixed strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )

这个函数找不到字符串的时候返回false，js中indexOf函数找不到字符串的时候返回-1，startsWith或者endsWith找不到返回false

int mb\_strpos ( string $haystack , string $needle [, int $offset = 0 [, string $encoding =

mb\_internal\_encoding() ]] )

#### 字符串替换

意思就是我把什么$search替换成什么$replace，在哪个字符串里面去替换

mixed str\_replace ( mixed $search , mixed $replace , mixed $subject [, int &$count ] )

#### 重复字符串

string str\_repeat ( string $input , int $multiplier )

#### 字符串分割

array explode( string $input, string $char )

补充：js中判断字符串中是否含有某个字符串，像判断str中是否含有http://，利用str.indexOf(‘http’) -->0/-1

Es6中多了一个新的函数判断是否以某个字符串开头：str.startsWith(‘http://’)-->判断是否以http://开头，结果为true/false

注意：这里有个作业，如下，结果图，参考文件C:\www\site1\day1作业 和 names.txt 和视频资料的01-list.php

注意这道题：

1. 注释，只要是写在php外面的东西，都会原封不动的输出

2.strpos函数选择确定字符串位置的时候，注意空格，那也算一位

3.if: else: endif 这样是正确的顺序

4.熟悉混编方式，是在php文件中混编，不是html中

5.注意换行\n 和<br>：它俩在源码和网页中是不一样的

知识利用：

1.

将$contents这个字符串，按行分割成数组$lines

$lines = explode("\n" , $contents);

2.

取数组$lines的值

foreach($lines as $item)

3.

向$data[]数组中添加元素

$data[] = $cols;

4.

确定$cols中’http://’的位置，返回具体数值，没有的话返回false

strpos($cols , 'http://'

5.

取消$cols两侧空格

trim($cols)

6.

从数组第7位开始截取输出

echo substr($cols , 7);

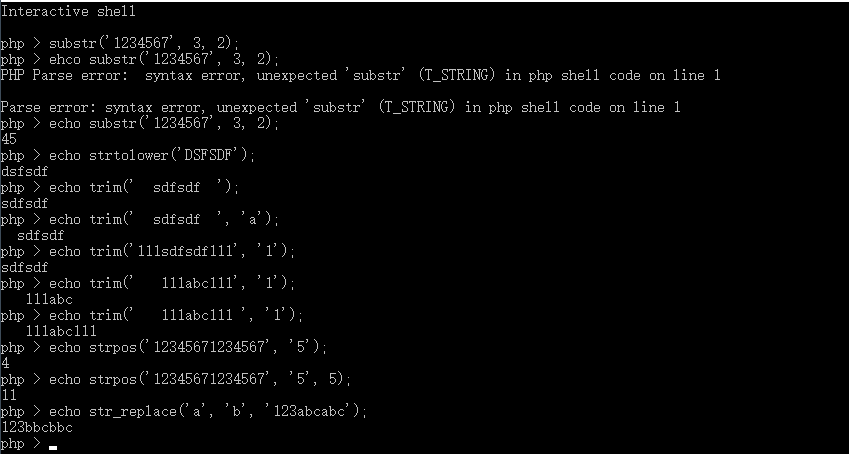
7.

将$cols两侧空格去掉并且小写输出

echo strtolower(trim($cols));



下图：trim中的第二个参数，就是要移除字符串两侧的对象，两侧要是没有，就不移除



## 数组处理

#### 获取关联数组中全部的键 / 值

array\_keys() / array\_values()

#### 判断关联数组中是否存在某个键

array\_key\_exists()

PHP笔记：第二天4

这里有两个重点isset和empty，具体看php笔记

还有注意：

Js中 ‘0’ == false -->true

#### 去除重复的元素

array\_unique()

例1：不改变原数组

var\_dump(array\_unique($dict);

#### 将一个或多个元素追加到数组中

array\_push()

$arr[] = 'new value'

例：

array\_push($dict ,1);

#### 删除数组中最后一个元素

array\_pop()

例：改变原数组

array\_pop($dict);

#### 数组长度

count()

#### 检测存在

in\_array()

例1：返回true或者false

var\_dump(in\_array( '4',$dict ));

补充：返回数组下标

array\_search()

例1：返回在数组中的下标

var\_dump(array\_search( '5',$dict ));

补充js知识点：includes判断一个元素是否在一个数组中出现，字符串中也有includes，判断一个字符串中是否含有某个字符串

例1：

var arr = [1,2,3,4]

arr.includes(5);

--> false

var str = “abc”;

str.includes(‘a’);

-->true

## 时间处理

时间戳： time()

#### 从 Unix 纪元（格林威治时间 1970-01-01 00:00:00）到当前时间的秒数

格式化日期： date()

#### 获取有格式的当前时间

格式化一个指定的时间戳

可以通过 strtotime() 将有格式的时间字符串转换为时间戳

例1：得到当前时间2018年10月19日 20:44:50

<?php  
$str = '2018-10-19 20:44:50';  
$timestamp = strtotime($str);  
echo date('Y年m月d日 H:i:s',$timestamp);

注意：

1.日期格式别写错了，也就是$str中的内容

2.Y-m-d-H:i:s大小写别写错，都有特殊意义的

3.很多字母有特殊含义，所以，有的要用的话记得要转义

例如：<br>，当我们想要在日之后换行的话，就需要将r转义，因为r是有特殊含义的，<b\r>才能实现换行的效果，

## 文件操作

|  |  |  |
| --- | --- | --- |
| 函数 | 描述 | **PHP** |
| [basename()](http://www.w3school.com.cn/php/func_filesystem_basename.asp) | 返回路径中的文件名部分。 | 3 |
| [copy()](http://www.w3school.com.cn/php/func_filesystem_copy.asp) | 复制文件。 | 3 |
| [dirname()](http://www.w3school.com.cn/php/func_filesystem_dirname.asp) | 返回路径中的目录名称部分。 | 3 |
| [disk\_free\_space()](http://www.w3school.com.cn/php/func_filesystem_disk_free_space.asp) | 返回目录的可用空间。 | 4 |
| [disk\_total\_space()](http://www.w3school.com.cn/php/func_filesystem_disk_total_space.asp) | 返回一个目录的磁盘总容量。 | 4 |
| [fclose()](http://www.w3school.com.cn/php/func_filesystem_fclose.asp) | 关闭打开的文件。 | 3 |
| [ﬁle()](http://www.w3school.com.cn/php/func_filesystem_file.asp) | 把文件读入一个数组中。 | 3 |
| [ﬁle\_exists()](http://www.w3school.com.cn/php/func_filesystem_file_exists.asp) | 检查文件或目录是否存在。 | 3 |
| [ﬁle\_get\_contents()](http://www.w3school.com.cn/php/func_filesystem_file_get_contents.asp) | 将文件读入字符串。 | 4 |
| [ﬁle\_put\_contents()](http://www.w3school.com.cn/php/func_filesystem_file_put_contents.asp) | 将字符串写入文件。 | 5 |
| [ﬁlesize()](http://www.w3school.com.cn/php/func_filesystem_filesize.asp) | 返回文件大小。 | 3 |
| [fopen()](http://www.w3school.com.cn/php/func_filesystem_fopen.asp) | 打开一个文件或 URL。 | 3 |
| [glob()](http://www.w3school.com.cn/php/func_filesystem_glob.asp) | 返回一个包含匹配指定模式的文件名/目录的数组。 | 4 |
| [is\_dir()](http://www.w3school.com.cn/php/func_filesystem_is_dir.asp) | 判断指定的文件名是否是一个目录。 | 3 |
| [is\_ﬁle()](http://www.w3school.com.cn/php/func_filesystem_is_file.asp) | 判断指定文件是否为常规的文件。 | 3 |
| [mkdir()](http://www.w3school.com.cn/php/func_filesystem_mkdir.asp) | 创建目录。 | 3 |
| [move\_uploaded\_ﬁle()](http://www.w3school.com.cn/php/func_filesystem_move_uploaded_file.asp) | 将上传的文件移动到新位置。 | 4 |
| [pathinfo()](http://www.w3school.com.cn/php/func_filesystem_pathinfo.asp) | 返回关于文件路径的信息。 | 4 |
| [rename()](http://www.w3school.com.cn/php/func_filesystem_rename.asp) | 重名名文件或目录。 | 3 |
| [rmdir()](http://www.w3school.com.cn/php/func_filesystem_rmdir.asp) | 删除空的目录。 | 3 |
| [unlink()](http://www.w3school.com.cn/php/func_filesystem_unlink.asp) | 删除文件。 | 3 |

#### 参考：http://www.w3school.com.cn/php/php\_ref\_filesystem.asp

补充：

.读取文件的5种方式，有时间看看

<https://blog.csdn.net/weiqing723/article/details/79040472>

php手册下载：

http://php.net/get/php\_enhanced\_zh.chm/from/a/mirror