

可变变量和预定义变量

1. 可变变量

```
$s1 = "abc" ;//这是一个变量，里存的是字符串“abc”
```

```
$abc = 10;
```

```
$mutable_var = $$s1;
```

```
echo "$mutabl_var" ;//输出结果为 10
```

(1) 一个\$后面总是跟着一个变量名，因为 s1 可变，所以 mutable_var 的变量值可以是很多变量的值，它拿变量 s1 的值作为了变量名，来取值；所以这里取的是\$abc 的值，即 10；

(2) 这种连续出现的\$的变量形式，就是所谓的可变变量。

以下是说明这种变量灵活性的应用

```
//以下演示“可变变量”的灵活性：
//所谓可变变量，其实就是变量的名字是可以“动态变化”以获取不同的数据值
$v1 = 1;
$v2 = 12;
$v3 = 33;
$v4 = 44;
$v5 = 115;
//求这5个变量的和；
//其他语言，只能“一个一个”加起来。
$sum = 0; //用于存储总和
for($i = 1; $i <= 5; $i++){
    $v = "v" . $i; //这里，结果其实只是一个“字符串”，比如“v1”，“v2”...
    $sum += $$v;
}
echo "<br />sum = $sum";
```

← → C www.php39.com/day2/6kebian_bianliang.php

10
sum = 205

2. 预定义变量

(1) 综述

1) 主要有：

\$_GET, \$_POST, \$_REQUEST, \$_SERVER, \$GLOBALS, \$_SESSION, \$_COOKIE, \$_FILES

2) 特点：

1). 均是数组

2). 系统定义与维护——即我们不应该给其赋值或者销毁其值，只应该使用

3). 该值

4). 具有超全局作用域——哪里都可以使用

5). 不同情形下可能具有不同的值

(2) \$_POST 变量

- 1) 含义：它代表用户通过以表单以 post 方式(methods = post)提交的时候所提交的所有数据——这个称为 post 数据
- 2) 演示 1：(html 表单提交数据到 php 文件)

```
test.html
<!DOCTYPE html>
<html>
<head>
<title>网页标题</title>
<meta http-equiv="Content-Type" content="text/html" charset="utf-8" />
</head>
<body>
<form action="test.php" method="post">
  输入1:<input type="text" name="data1">
  <br>
  输入2:<input type="text" name="data2">
  <br>
  <input type="submit" value="提交">
</form>
</body>
</html>

test.php
1 <?
2     echo "Hello world<br><br>";
3     echo "当前时间: ".date("Y-m-d H:i:s");
4     echo "<br>";
5
6     $data1 = $_POST["data1"];
7     $data2 = $_POST["data2"];
8
9     if(!empty($_POST)){
10         $data1 = 2;
11         $data2 = 4;
12         echo "$data1<br>$data2<br>";
13         var_dump($_POST);
14     }else{
15         echo "非法的页面访问";
16     }
17 ?>
```

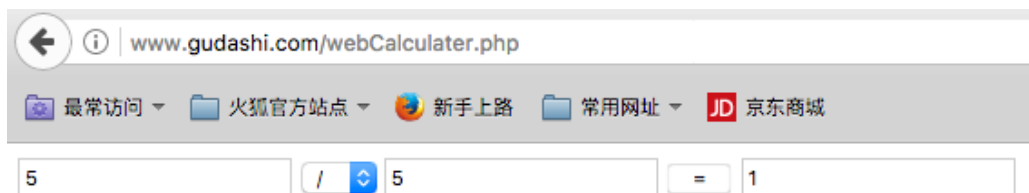
当然，直接访问 php 文件，会报错，为了不报错，需要有数据判空处理。

Isset(var)：判断变量是否存在，或变量是否为空(null)；如果存在，就是 true，否则就是 false，所以不能用 isset(var)判断

Empty(变量)：判断变量的“内容”是否为空(不是 null 的空, 而是没有内容)，基本上，是一些硬性规定，如下数据都是“空的”：0，“”，“0”，false, null, array() 空数组

- 3) 演示 2：网页计算器

```
webCalculator.php
1 <?php
2 <!-- 计算器 -->
3 <!-- 防止第一次加载网页，后面文本框显示报错 -->
4 <!-- 防止第一次加载网页，后面文本框显示报错 -->
5 <!-- 防止第一次加载网页，后面文本框显示报错 -->
6 <!-- 防止第一次加载网页，后面文本框显示报错 -->
7 <!-- 防止第一次加载网页，后面文本框显示报错 -->
8 <!-- 防止第一次加载网页，后面文本框显示报错 -->
9 <!-- 防止第一次加载网页，后面文本框显示报错 -->
10 <!-- 防止第一次加载网页，后面文本框显示报错 -->
11 <!-- 防止第一次加载网页，后面文本框显示报错 -->
12 <!-- 防止第一次加载网页，后面文本框显示报错 -->
13 <!-- 防止第一次加载网页，后面文本框显示报错 -->
14 <!-- 防止第一次加载网页，后面文本框显示报错 -->
15 <!-- 防止第一次加载网页，后面文本框显示报错 -->
16 <!-- 防止第一次加载网页，后面文本框显示报错 -->
17 <!-- 防止第一次加载网页，后面文本框显示报错 -->
18 <!-- 防止第一次加载网页，后面文本框显示报错 -->
19 <!-- 防止第一次加载网页，后面文本框显示报错 -->
20 <!-- 防止第一次加载网页，后面文本框显示报错 -->
21 <!-- 防止第一次加载网页，后面文本框显示报错 -->
22 <!-- 防止第一次加载网页，后面文本框显示报错 -->
23 <!-- 防止第一次加载网页，后面文本框显示报错 -->
24 <!-- 防止第一次加载网页，后面文本框显示报错 -->
25 <!-- 防止第一次加载网页，后面文本框显示报错 -->
26 <!-- 防止第一次加载网页，后面文本框显示报错 -->
27 <!-- 防止第一次加载网页，后面文本框显示报错 -->
28 <!-- 防止第一次加载网页，后面文本框显示报错 -->
29 <!-- 防止第一次加载网页，后面文本框显示报错 -->
30 <!-- 防止第一次加载网页，后面文本框显示报错 -->
31 <!-- 防止第一次加载网页，后面文本框显示报错 -->
32 <!-- 防止第一次加载网页，后面文本框显示报错 -->
33 <!-- 防止第一次加载网页，后面文本框显示报错 -->
34 <!-- 防止第一次加载网页，后面文本框显示报错 -->
35 <!-- 防止第一次加载网页，后面文本框显示报错 -->
36 <!-- 防止第一次加载网页，后面文本框显示报错 -->
37 <!-- 防止第一次加载网页，后面文本框显示报错 -->
38 <!-- 防止第一次加载网页，后面文本框显示报错 -->
39 <!-- 防止第一次加载网页，后面文本框显示报错 -->
40 <!-- 防止第一次加载网页，后面文本框显示报错 -->
41 <!-- 防止第一次加载网页，后面文本框显示报错 -->
42 <!-- 防止第一次加载网页，后面文本框显示报错 -->
43 <!-- 防止第一次加载网页，后面文本框显示报错 -->
44 <!-- 防止第一次加载网页，后面文本框显示报错 -->
45 <!-- 防止第一次加载网页，后面文本框显示报错 -->
46 <!-- 防止第一次加载网页，后面文本框显示报错 -->
47 <!-- 防止第一次加载网页，后面文本框显示报错 -->
```



(3) \$_GET 变量

含义:

它代表用户使用 get 方式(有 5 中 get 方式)提交的时候所提交的所有数据——这个称为 get 数据。

小提示: get, post 没有翻译!

1) 形式 1:

```
<form action="目标文件.php" method="get">
    <input type="text" name="data1">
    <input type="text" name="data2">
    <input type="submit" value="提交">
</form>
```

这种形式的 get 数据, 跟 post 数据类似, 数据内容由用户填写或选择而得到!

2) 形式 2:

```
<a href=" 目标文件.php?data1=5&data2=cctv&age=18" >链接文字</a>
```

说明:

1. 它只是一个链接而已, 只是在链接文件名的后面加上“?”, 然后一个“串接数据”;
2. 数据形式为: 数据项名称=数据值, 相互之间用“&”符号隔开
3. 这种形式的数据同样是”点击链接”就提交的 get 数据, 但用户只能选择点还是不点, 而不能修改数据。

3) 形式 3:

```
<script>
    location.href=" 目标文件.php?data1=5&data2=cctv&age=18" ;
</script>
```

说明：

1. 该语句可以看作是通过 js 技术实现的页面跳转功能，跟 a 标签的链接功能完全一样！
2. 其中该语句，通常都是放在一个函数中，然后函数因为某个事件发生而去调用该语句

4) 形式 4:

```
<script>

    location.assign(“目标文件.php?data1=5&data2=cctv&age=18”)

</script>
```

5) 形式 5:

```
<?php

    //语法形式：header(“location:webCalclater.php(站点下的目录文件)” )

    header(“location:目标文件.php 文件?data1=5&data2=cctv&age=18”)

?>
```

小结：

上述多种 get 形式提交数据，都要理解为：跳转到某个页面，并“同时”携带(提交)一定的 get 数据过去！

不管哪种形式的 get 数据提交，接收 get 数据，都只有一种形式，跟 post 类似：

```
$v1=$_GET[“数据项名称”]; //取得 get 数据项的值；比如$_GET[“data1”],$_GET[“age”];
```

也可以“输出”所有 get 数据：

```
print_r($_GET); 或 var_dump($_GET);
```

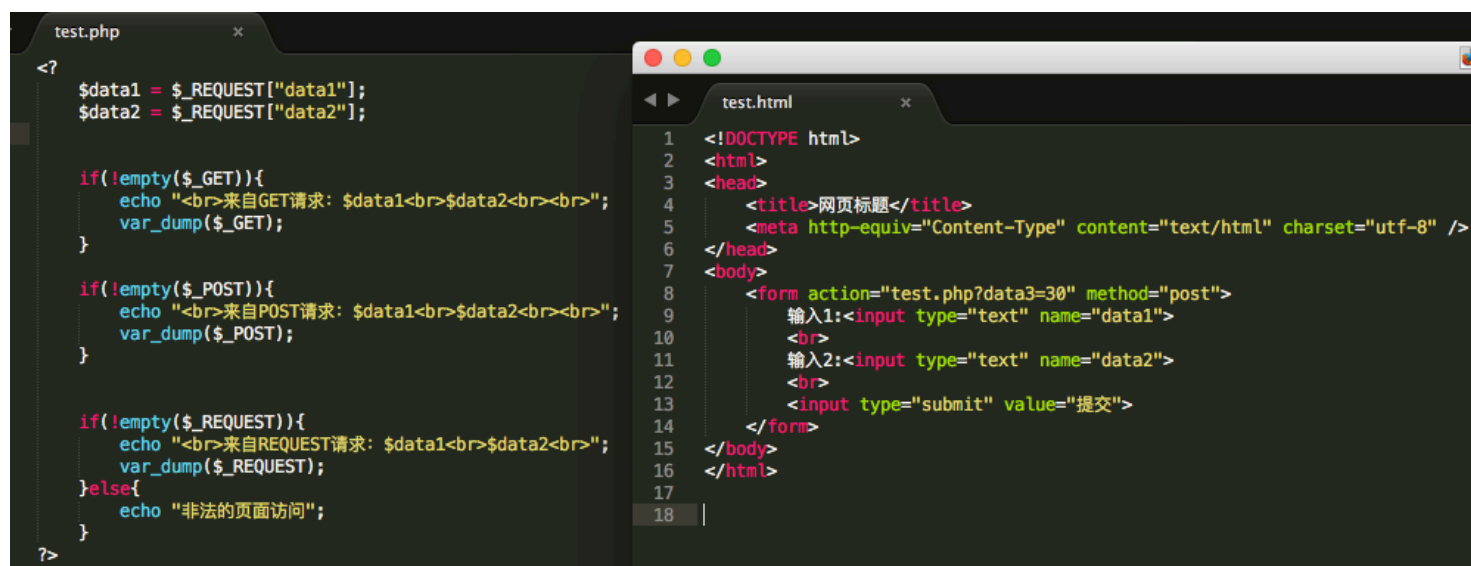
(4) \$_REQUEST 变量

含义：

一句话，它是\$_GET 变量和\$_POST 变量数据的”合集”：即，它里面同时存储了这两种数据。

测试：如下

结果：



The screenshot shows two files in a code editor. The left file, test.php, contains PHP code that checks for GET, POST, and REQUEST data and outputs it. The right file, test.html, contains an HTML form with two text inputs named 'data1' and 'data2', and a submit button. The form's action is 'test.php?data3=30' and its method is 'post'.

```
<?
$data1 = $_REQUEST["data1"];
$data2 = $_REQUEST["data2"];

if(!empty($_GET)){
    echo "<br>来自GET请求: $data1<br>$data2<br><br>";
    var_dump($_GET);
}

if(!empty($_POST)){
    echo "<br>来自POST请求: $data1<br>$data2<br><br>";
    var_dump($_POST);
}

if(!empty($_REQUEST)){
    echo "<br>来自REQUEST请求: $data1<br>$data2<br>";
    var_dump($_REQUEST);
}else{
    echo "非法的页面访问";
}
?>
```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>网页标题</title>
5   <meta http-equiv="Content-Type" content="text/html" charset="utf-8" />
6 </head>
7 <body>
8   <form action="test.php?data3=30" method="post">
9     输入1:<input type="text" name="data1">
10    <br>
11    输入2:<input type="text" name="data2">
12    <br>
13    <input type="submit" value="提交">
14  </form>
15 </body>
16 </html>
17
18
```

来自GET请求: 55
44

array(1) { ["data3"]=> string(2) "30" }
来自POST请求: 55
44

array(2) { ["data1"]=> string(2) "55" ["data2"]=> string(2) "44" }
来自REQUEST请求: 55
44

array(3) { ["data3"]=> string(2) "30" ["data1"]=> string(2) "55" ["data2"]=> string(2) "44" }

html 的表单提交，同时提交了 get 和 post 数据！

注意：

当 post 数据和 get 数据的数据项名称相同时（其实我们反对这么做），默认是 post 数据覆盖了 get 数据。

不过这个状况同样可以在 php.ini 中设置：

默认时：

```
693 ; http://php.net/request-order
694 request_order = "GP" 就是GET和POST，后者覆盖前者。
695
```

可修改为：

```
693 ; http://php.net/request-order
694 request_order = "PG" 此时就是GET数据覆盖POST数据
695
```

如果能明确鉴定 post、get 就应该使用 get、post，使用 request 其实是不大推荐的，因为不知道数据来源！这是不清晰的编程模式。

(5) \$_SERVER 变量

含义：它代表在一次浏览网页的过程中的浏览器端的一些信息和服务器端的一些信息。

注意：这种信息，随着不同的页面，和不同的服务器，以及不同的时刻，都可能不同！

要求：大约有 30 个左右的信息，我们只要知道其中的 5 个左右

```
$_SERVER['REMOTE_ADDR']: 获取访问者的 ip 地址  
$_SERVER['SERVER_ADDR']: 获取服务器所在的 ip 地址  
$_SERVER['SERVER_NAME']: 获取服务器的名字，其实就是站点设置中的 servername  
$_SERVER['DOCUMENT_ROOT']: 获取站点的真实物理地址，其实就是站点设置中的 documentroot  
$_SERVER['PHP_SELF']: 获取当前网页地址（不含域名部分）  
$_SERVER['SCRIPT_FILENAME']: 获取当前网页地址物理路径  
$_SERVER['QUERY_STRING']: 获取当前网页地址中的所有 get 数据（就是？号后面部分），但只是一个整体的字符串而已。
```

(6) \$GLOBALS 变量

含义：它也是一个“重复性数据”，它里面存储了我们自己定义的所有“全局变量”。

举例：

```
$v1 = 1; // 定义了一个全局变量
```

此时，就有了这样一个数据：\$GLOBALS['v1']，其值就是 1

```
echo $v1; // 输出 1
```

```
echo $GLOBALS[ 'v1' ]; // 输出 1
```

应用：有些地方局部范围，全局变量不能使用，但如果想用之前定义的全局变量，可以用 GLOBALS，这是超全局作用域。

GLOBALS 内也有一些内部定义的比如 POST、GET 等。但是我们通常不会通过这种方式去读取。