**PHP 大小写敏感**

在 PHP 中，所有用户定义的函数、类和关键词（例如 if、else、echo 等等）都对大小写不敏感。

在下面的例子中，所有这三条 echo 语句都是合法的（等价）：

<!DOCTYPE html>

<html>

<body>

<?php

ECHO "Hello World!<br>";

echo "Hello World!<br>";

EcHo "Hello World!<br>";

?>

</body>

</html>

不过在 PHP 中，所有变量都对大小写敏感。

## 创建 PHP 变量

PHP 没有创建变量的命令。

变量会在首次为其赋值时被创建：

### 实例

<?php

$txt="Hello world!";

$x=5;

$y=10.5;?>

**注释：**如果您为变量赋的值是文本，请用引号包围该值。

## PHP 是一门类型松散的语言

在上面的例子中，请注意我们不必告知 PHP 变量的数据类型。

PHP 根据它的值，自动把变量转换为正确的数据类型。

## PHP 变量作用域

在 PHP 中，可以在脚本的任意位置对变量进行声明。

变量的作用域指的是变量能够被引用/使用的那部分脚本。

PHP 有三种不同的变量作用域：

* local（局部）
* global（全局）
* static（静态）

## Local 和 Global 作用域

函数**之外**声明的变量拥有 Global 作用域，只能在函数以外进行访问。

函数**内部**声明的变量拥有 LOCAL 作用域，只能在函数内部进行访问。

下面的例子测试了带有局部和全局作用域的变量：

图形用户界面, 应用程序

描述已自动生成

global 关键词用于访问函数内的全局变量

如果要在函数内调用函数外的全局变量则添加global即可访问。

文本

描述已自动生成

但是注意了 变量的顺序不能反了加入要实现相加的话 代码从上往下遍历的 否则代码对的也报错因为下面还没识别到

## PHP static 关键词

通常，当函数完成/执行后，会删除所有变量。不过，有时我需要不删除某个局部变量。实现这一点需要更进一步的工作。

要完成这一点，请在您首次声明变量时使用 **static** 关键词：

<?php

function myTest() {

static $x=0;

echo $x;

$x++;

}

图片包含 文本

描述已自动生成

myTest();

myTest();

myTest();

?>

**PHP echo 和 print 语句**

echo 和 print 之间的差异：

* echo - 能够输出一个以上的字符串
* print - 只能输出一个字符串，并始终返回 1

**提示：**echo 比 print 稍快，因为它不返回任何值。

echo 是一个语言结构，有无括号均可使用：echo 或 echo()

print 也是语言结构，有无括号均可使用：print 或 print()。

**<?php**

**echo "<h2>PHP 很有趣！</h2>";**

**echo "Hello world!<br>";**

**echo "我计划学习 PHP！<br>";**

**echo "这段话", "由", "多个", "字符串", "串接而成。";**

**?>**

PHP 很有趣！

Hello world!  
我计划学习 PHP！  
这段话由多个字符串串接而成。

<?php

$txt1="Learn PHP";

$txt2="W3School.com.cn";

$cars=array("Volvo","BMW","SAAB");

echo $txt1;

echo "<br>";

echo "Study PHP at $txt2";

echo "My car is a {$cars[0]}";

?>

## PHP 字符串

字符串是字符序列，比如 "Hello world!"。

**字符串可以是引号内的任何文本。您可以使用单引号或双引号：**

## PHP NULL 值

特殊的 NULL 值表示变量无值。NULL 是数据类型 NULL 唯一可能的值。

NULL 值标示变量是否为空。也用于区分空字符串与空值数据库。

可以通过把值设置为 NULL，将变量清空：<?php

$x="Hello world!";

$x=null;

var\_dump($x);

?>

## PHP 对象

对象是存储数据和有关如何处理数据的信息的数据类型。

在 PHP 中，必须明确地声明对象。

首先我们必须声明对象的类。对此，我们使用 class 关键词。类是包含属性和方法的结构。

然后我们在对象类中定义数据类型，然后在该类的实例中使用此数据类型：

<?php

class Car

{

var $color;

function Car($color="green") {

$this->color = $color;

}

function what\_color() {

return $this->color;

}

}

?>

**PHP 整数**

整数是没有小数的数字。

整数规则：

* 整数必须有至少一个数字（0-9）
* 整数不能包含逗号或空格
* 整数不能有小数点
* 整数正负均可
* 可以用三种格式规定整数：十进制、十六进制（前缀是 0x）或八进制（前缀是 0）

## PHP 浮点数

浮点数是有小数点或指数形式的数字。

## PHP 逻辑

逻辑是 true 或 false。

$x=true;

$y=false;

## PHP 数组

数组在一个变量中存储多个值。

## PHP strlen() 函数

strlen() 函数返回字符串的长度，以字符计。

下例返回字符串 "Hello world!" 的长度：

<?php

echo strlen("Hello world!");

?>

## PHP 常量

常量是单个值的标识符（名称）。在脚本中无法改变该值。

有效的常量名以字符或下划线开头（常量名称前面没有 $ 符号）。

**注释：**与变量不同，常量贯穿整个脚本是自动全局的。

## 设置 PHP 常量

如需设置常量，请使用 define() 函数 - 它使用三个参数：

1. 首个参数定义常量的名称
2. 第二个参数定义常量的值
3. 可选的第三个参数规定常量名是否对大小写不敏感。默认是 false。

下例创建了一个**对大小写敏感的常量**，值为 "Welcome to W3School.com.cn!"：

### 实例

<?php

define("GREETING", "Welcome to W3School.com.cn!");

echo GREETING;

?>

下例创建了一个**对大小写不敏感的常量**，值为 "Welcome to W3School.com.cn!"：

### 实例

<?php

define("GREETING", "Welcome to W3School.com.cn!", true);

echo greeting;

?>

## PHP 字符串运算符

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **运算符** | **名称** | **例子** | **结果** | **显示结果** |
| . | 串接 | $txt1 = "Hello" $txt2 = $txt1 . " world!" | 现在 $txt2 包含 "Hello world!" | [显示结果](https://www.w3school.com.cn/tiy/s.asp?f=demo_php_operator_string1) |
| .= | 串接赋值 | $txt1 = "Hello" $txt1 .= " world!" | 现在 $txt1 包含 "Hello world!" | [显示结果](https://www.w3school.com.cn/tiy/s.asp?f=demo_php_operator_string2) |

<?php

$a = "Hello";

$b = $a . " world!";

echo $b; // 输出 Hello world!

$x="Hello";

$x .= " world!";

echo $x; // 输出 Hello world!

?>

<?php

$favfruit="orange";

# Switch语句

switch ($favfruit) {

case "apple":

echo "Your favorite fruit is apple!";

break;

case "banana":

echo "Your favorite fruit is banana!";

break;

case "orange":

echo "Your favorite fruit is orange!";

break;

default:

echo "Your favorite fruit is neither apple, banana, or orange!";

}

?>

## PHP foreach 循环

foreach 循环只适用于数组，并用于遍历数组中的每个键/值对。

<?php

$colors = array("red","green","blue","yellow");

foreach ($colors as $value) {

echo "$value <br>";

}

?>

**在 PHP 中创建数组**

在 PHP 中， array() 函数用于创建数组：

array();

在 PHP 中，有三种数组类型：

* **索引数组** - 带有数字索引的数组
* **关联数组** - 带有指定键的数组
* **多维数组** - 包含一个或多个数组的数组
* 关联数组是使用您分配给数组的指定键的数组。
* 有两种创建关联数组的方法：
* $age=array("Bill"=>"35","Steve"=>"37","Elon"=>"43");
* $age['Bill']="63";
* $age['Steve']="56";
* $age['Elon']="47";
* <?php
* $age=array("Bill"=>"63","Steve"=>"56","Elon"=>"47");
* echo "Elon is " . $age['Elon'] . " years old.";
* ?>

## 遍历索引数组

如需遍历并输出索引数组的所有值，您可以使用 for 循环，就像这样：

<?php

$cars=array("porsche","BMW","Volvo");

$arrlength=count($cars);

for($x=0;$x<$arrlength;$x++) {

echo $cars[$x];

echo "<br>";

}

?>

## 对数组进行升序排序 - sort()

下面的例子按照字母升序对数组 $cars 中的元素进行排序：

## 对数组进行降序排序 - rsort()

下面的例子按照字母降序对数组 $cars 中的元素进行排序：

sort($numbers);

rsort($cars);

## 根据值对数组进行升序排序 - asort()

下面的例子根据值对关联数组进行升序排序：

### 实例

<?php

$age=array("Bill"=>"63","Steve"=>"56","Elon"=>"47");

asort($age);

?>

根据 56 47 来排序

## 根据键对数组进行升序排序 - ksort()

下面的例子根据键对关联数组进行升序排序：

### 实例

<?php

$age=array("Bill"=>"63","Steve"=>"56","Elon"=>"47");

ksort($age);

?>

根据bill 来排序

**PHP 全局变量 - 超全局变量**

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
* $\_SERVER
* $\_REQUEST
* $\_POST
* $\_GET
* $\_FILES
* $\_ENV
* $\_COOKIE
* $\_SESSION

## PHP $\_REQUEST

PHP $\_REQUEST 用于收集 HTML 表单提交的数据。

下面的例子展示了一个包含输入字段及提交按钮的表单。当用户通过点击提交按钮来提交表单数据时, 表单数据将发送到 <form> 标签的 action 属性中指定的脚本文件。在这个例子中，我们指定文件本身来处理表单数据。如果您需要使用其他的 PHP 文件来处理表单数据，请修改为您选择的文件名即可。然后，我们可以使用超级全局变量 $\_REQUEST 来收集 input 字段的值：

## PHP $\_POST

PHP $\_POST 广泛用于收集提交 method="post" 的 HTML 表单后的表单数据。$\_POST 也常用于传递变量。

下面的例子展示了一个包含输入字段和提交按钮的表单。当用户点击提交按钮来提交数据后，表单数据会发送到 <form> 标签的 action 属性中指定的文件。在本例中，我们指定文件本身来处理表单数据。如果您希望使用另一个 PHP 页面来处理表单数据，请用更改为您选择的文件名。然后，我们可以使用超全局变量 $\_POST 来收集输入字段的值：

## PHP $\_GET

PHP $\_GET 也可用于收集提交 HTML 表单 (method="get") 之后的表单数据。

$\_GET 也可以收集 URL 中的发送的数据。

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。

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。限制在大约 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。

**注释：**绝不能使用 GET 来发送密码或其他敏感信息！

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**。

此外 POST 支持高阶功能，比如在向服务器上传文件时进行 multi-part 二进制输入。

不过，由于变量未显示在 URL 中，也就无法将页面添加到书签。

### 什么是 $\_SERVER["PHP\_SELF"] 变量？

$\_SERVER["PHP\_SELF"] 是一种超全局变量，它返回当前执行脚本的文件名。

**因此，$\_SERVER["PHP\_SELF"] 将表单数据发送到页面本身，而不是跳转到另一张页面。这样，用户就能够在表单页面获得错误提示信息。**

$\_SERVER["PHP\_SELF"] 变量能够被黑客利用！

如果您的页面使用了 PHP\_SELF，用户能够输入下划线然后执行跨站点脚本（XSS）。

**提示：**跨站点脚本（Cross-site scripting，XSS）是一种计算机安全漏洞类型，常见于 Web 应用程序。XSS 能够使攻击者向其他用户浏览的网页中输入客户端脚本。

## 如果避免 $\_SERVER["PHP\_SELF"] 被利用？

通过使用 htmlspecialchars() 函数能够避免 $\_SERVER["PHP\_SELF"] 被利用。

<?php

// 定义变量并设置为空值

$name = $email = $gender = $comment = $website = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$name = test\_input($\_POST["name"]);

$email = test\_input($\_POST["email"]);

$website = test\_input($\_POST["website"]);

$comment = test\_input($\_POST["comment"]);

$gender = test\_input($\_POST["gender"]);

}

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

请注意在脚本开头，我们检查了表单是否使用 $\_SERVER["REQUEST\_METHOD"] 进行提交。如果 REQUEST\_METHOD 是 POST，那么表单已被提交 - 并且应该对其进行验证。如果未提交，则跳过验证并显示一个空白表单。

不过，在上面的例子中，所有输入字段都是可选的。即使用户未输入任何数据，脚本也能正常工作。

下一步是制作必填输入字段，并创建需要时使用的错误消息。

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

if (empty($\_POST["name"])) {

$nameErr = "姓名是必填的";

} else {

$name = test\_input($\_POST["name"]);

}

姓名：<input type="text" name="name">

<span class="error">\* <?php echo $nameErr;?></span>

## 什么是 Cookie？

**cookie 常用于识别用户**。cookie 是服务器留在用户计算机中的小文件。每当相同的计算机通过浏览器请求页面时，它同时会发送 cookie。通过 PHP，您能够创建并取回 cookie 的值。

## 如何创建 cookie？

setcookie() 函数用于设置 cookie。

**注释：**setcookie() 函数必须位于 <html> 标签之前。

在下面的例子中，我们将创建名为 "user" 的 cookie，把为它赋值 "Alex Porter"。我们也规定了此 cookie 在一小时后过期：

<?php

setcookie("user", "Alex Porter", time()+3600);

?>

<html>

<body>

</body>

</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）。

PHP 的 $\_COOKIE 变量用于取回 cookie 的值。 **//必须大写**

在下面的例子中，我们取回了名为 "user" 的 cookie 的值，并把它显示在了页面上：

<?php

// 输出一个cookie

echo $\_COOKIE["user"];

// 输出全部cookie

print\_r($\_COOKIE);

?>

在下面的例子中，我们使用 isset() 函数来确认是否已设置了 cookie：

<html>

<body>

<?php

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

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

else

echo "Welcome guest!<br />";

?>

</body>

</html>

## 如何删除 cookie？

当删除 cookie 时，您应当使过期日期变更为过去的时间点。

删除的例子：

<?php

// set the expiration date to one hour ago

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

?>

## 如果浏览器不支持 cookie 该怎么办？

如果您的应用程序涉及不支持 cookie 的浏览器， 用表单传输 form

**PHP session 变量用于存储有关用户会话的信息，或更改用户会话的设置。Session 变量保存的信息是单一用户的，并且可供应用程序中的所有页面使用。**

## PHP Session 变量

当您运行一个应用程序时，您会打开它，做些更改，然后关闭它。这很像一次会话。计算机清楚你是谁。它知道你何时启动应用程序，并在何时终止。但是在因特网上，存在一个问题：服务器不知道你是谁以及你做什么，这是由于 HTTP 地址不能维持状态。

通过在服务器上存储用户信息以便随后使用，PHP session 解决了这个问题（比如用户名称、购买商品等）。不过，会话信息是临时的，在用户离开网站后将被删除。如果您需要永久储存信息，可以把数据存储在数据库中。

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

## 开始 PHP Session

在您把用户信息存储到 PHP session 中之前，首先必须启动会话。

**注释：**session\_start() 函数必须位于 <html> 标签之前：

上面的代码会向服务器注册用户的会话，以便您可以开始保存用户信息，同时会为用户会话分配一个 UID。

**存储和取回 session 变量的正确方法是使用 PHP $\_SESSION 变量：**

<?php

session\_start();

// store session data

$\_SESSION['views']=1;

?>

<html>

<body>

<?php

//retrieve session data

echo "Pageviews=". $\_SESSION['views'];

?>

</body>

</html>

输出：

Pageviews=1

## 终结 Session

如果您希望删除某些 session 数据，可以使用 unset() 或 session\_destroy() 函数。

unset() 函数用于释放指定的 session 变量：

<?php

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

?>

您也可以通过 session\_destroy() 函数彻底终结 session：

<?php

session\_destroy();

?>

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

## 基本的错误处理：使用 die() 函数

**在 PHP 中，默认的错误处理很简单。一条消息会被发送到浏览器，这条消息带有文件名、行号以及一条描述错误的消息。**

<?php

if(!file\_exists("welcome.txt"))

{

die("File not found");

}

else

{

$file=fopen("welcome.txt","r");

}

?>

现在，假如文件不存在，您会得到类似这样的错误消息：

File not found

比起之前的代码，上面的代码更有效，这是由于它采用了一个简单的错误处理机制在错误之后终止了脚本。

不过，简单地终止脚本并不总是恰当的方式。让我们研究一下用于处理错误的备选的 PHP 函数。

**连接数据库 sql server**

**<?php**

**header("Content-Type:text/html;charset=utf-8");**

**// 定义常量 define(常量名,值) 常量名一般大写**

**define('DB\_HOST','localhost');**

**define('DB\_USER','root');**

**define('DB\_PASS','');**

**define('DB\_NAME','gbbook');**

**define('DB\_CHARSET','utf8');**

**date\_default\_timezone\_set("PRC");**

**<?php**

**include "config.inc.php";**

**$sql="select \* from user";**

**$mysqli = new mysqli(DB\_HOST,DB\_USER,DB\_PASS,DB\_NAME);**

**if ($mysqli -> connect\_error) {**

**die("连接错误：".$mysqli -> connect\_error);**

**}**

**$mysqli -> set\_charset(DB\_CHARSET);**

mysqli\_set\_charset() 函数规定当与数据库服务器进行数据传送时要使用的默认字符集。

**$result = $mysqli -> query($sql);**

mysqli\_query() 函数执行某个针对数据库的查询。

## 语法

mysqli\_query(connection,query,resultmode);

**?>**

**上述是sqlserver语法。 和mysql语法引入有区别。**

#### 引用文件 "time.inc" 的正确方法是？

<?php require("time.inc"); ?>