**什么是 PHP 过滤器？**

PHP 过滤器用于验证和过滤来自非安全来源的数据。

验证和过滤用户输入或自定义数据是任何 Web 应用程序的重要组成部分。

设计 PHP 的过滤器扩展的目的是使数据过滤更轻松快捷。

## 为什么使用过滤器？

几乎所有 web 应用程序都依赖外部的输入。这些数据通常来自用户或其他应用程序（比如 web 服务）。通过使用过滤器，您能够确保应有程序获得正确的输入类型。

您应该始终对外部数据进行过滤！

输入过滤是最重要的应用程序安全课题之一。

### 什么是外部数据？

* 来自表单的输入数据
* Cookies
* 服务器变量
* 数据库查询结果

**函数和过滤器**

如需过滤变量，请使用下面的过滤器函数之一：

* filter\_var() - 通过一个指定的过滤器来过滤单一的变量
* filter\_var\_array() - 通过相同的或不同的过滤器来过滤多个变量
* filter\_input - 获取一个输入变量，并对它进行过滤
* filter\_input\_array - 获取多个输入变量，并通过相同的或不同的过滤器对它们进行过滤

## filter\_var() 函数通过指定的过滤器过滤变量。

如果成功，则返回已过滤的数据，如果失败，则返回 false。

### 语法

filter\_var(variable, filter, options)

|  |  |
| --- | --- |
| **参数** | **描述** |
| variable | 必需。规定要过滤的变量。 |
| filter | 可选。规定要使用的过滤器的 ID。 |
| options | 规定包含标志/选项的数组。检查每个过滤器可能的标志和选项。 |

在下面的例子中，我们用 filter\_var() 函数验证了一个整数：

**<?php**$int = 123;  
**if** (filter\_var($int, *FILTER\_VALIDATE\_INT*)){  
 **echo**("Integer is not valid");  
 **echo** filter\_var($int, *FILTER\_VALIDATE\_INT*); //练习开启，实际环境不用开启  
}  
**else**{  
 **echo**("Integer is valid");  
}

## Validating 和 Sanitizing

### Validating 过滤器：

* 用于验证用户输入
* 严格的格式规则（比如 URL 或 E-Mail 验证）
* 如果成功则返回预期的类型，如果失败则返回 FALSE

### Sanitizing 过滤器：

* 用于允许或禁止字符串中指定的字符
* 无数据格式规则
* 始终返回字符串
* FILTER\_SANITIZE\_EMAIL 从字符串中删除电子邮件的非法字符
* FILTER\_VALIDATE\_EMAIL 验证电子邮件地址

## 选项和标志

选项和标志用于向指定的过滤器添加额外的过滤选项。

不同的过滤器有不同的选项和标志。

在下面的例子中，我们用 filter\_var() 和 "min\_range" 以及 "max\_range" 选项验证了一个整数：

<?php

$var=300;

$int\_options = array(

"options"=>array

(

"min\_range"=>0,

"max\_range"=>256

)

);

if(!filter\_var($var, FILTER\_VALIDATE\_INT, $int\_options))

{

echo("Integer is not valid");

}

else

{

echo("Integer is valid");

}

?>

就像上面的代码一样，选项必须放入一个名为 "options" 的相关数组中。如果使用标志，则不需在数组内。

由于整数是 "300"，它不在指定的范围内，以上代码的输出将是 "Integer is not valid"。

## 验证输入

让我们试着验证来自表单的输入。

我们需要作的第一件事情是确认是否存在我们正在查找的输入数据。

然后我们用 filter\_input() 函数过滤输入的数据。

在下面的例子(只是解释代码，并不能实际运行)中，输入变量 "email" 被传到 PHP 页面：

**<?php  
if**(!filter\_has\_var(*INPUT\_GET*, "email"))  
{  
 **echo**("Input type does not exist");  
}  
**else**{  
 **if** (!filter\_input(*INPUT\_GET*, "email", *FILTER\_VALIDATE\_EMAIL*))  
 {  
 **echo** "E-Mail is not valid";  
 }  
 **else** {  
 **echo** "E-Mail is valid";  
 }  
}  
**?>**

### 例子解释：

上面的例子有一个通过 "GET" 方法传送的输入变量 (email)：

1. 检测是否存在 "GET" 类型的 "email" 输入变量
2. 如果存在输入变量，检测它是否是有效的邮件地址

## 净化输入

让我们试着清理一下从表单传来的 URL。

首先，我们要确认是否存在我们正在查找的输入数据。

然后，我们用 filter\_input() 函数来净化输入数据。

在下面的例子中，输入变量 "url" 被传到 PHP 页面：

<?php

if(!filter\_has\_var(INPUT\_POST, "url"))

{

echo("Input type does not exist");

}

else

{

$url = filter\_input(INPUT\_POST, "url", FILTER\_SANITIZE\_URL);

}

?>

### 例子解释：

上面的例子有一个通过 "POST" 方法传送的输入变量 (url)：

1. 检测是否存在 "POST" 类型的 "url" 输入变量
2. 如果存在此输入变量，对其进行净化（删除非法字符），并将其存储在 $url 变量中

假如输入变量类似这样："http://www.W3非o法ol.com.c字符n/"，则净化后的 $url 变量应该是这样的：

<http://www.W3School.com.cn/>

## 过滤多个输入

表单通常由多个输入字段组成。为了避免对 filter\_var 或 filter\_input 重复调用，我们可以使用 filter\_var\_array 或 the filter\_input\_array 函数。

在本例中，我们使用 filter\_input\_array() 函数来过滤三个 GET 变量。接收到的 GET 变量是一个名字、一个年龄以及一个邮件地址：

<?php

$filters = array

(

"name" => array

(

"filter"=>FILTER\_SANITIZE\_STRING

),

"age" => array

(

"filter"=>FILTER\_VALIDATE\_INT,

"options"=>array

(

"min\_range"=>1,

"max\_range"=>120

)

),

"email"=> FILTER\_VALIDATE\_EMAIL,

);

$result = filter\_input\_array(INPUT\_GET, $filters);

if (!$result["age"])

{

echo("Age must be a number between 1 and 120.<br />");

}

elseif(!$result["email"])

{

echo("E-Mail is not valid.<br />");

}

else

{

echo("User input is valid");

}

?>

### 例子解释：

上面的例子有三个通过 "GET" 方法传送的输入变量 (name, age and email)

1. 设置一个数组，其中包含了输入变量的名称，以及用于指定的输入变量的过滤器
2. 调用 filter\_input\_array 函数，参数包括 GET 输入变量及刚才设置的数组
3. 检测 $result 变量中的 "age" 和 "email" 变量是否有非法的输入。（如果存在非法输入，）

filter\_input\_array() 函数的第二个参数可以是数组或单一过滤器的 ID。

如果该参数是单一过滤器的 ID，那么这个指定的过滤器会过滤输入数组中所有的值。

如果该参数是一个数组，那么此数组必须遵循下面的规则：

* 必须是一个关联数组，其中包含的输入变量是数组的键（比如 "age" 输入变量）
* 此数组的值必须是过滤器的 ID ，或者是规定了过滤器、标志以及选项的数组

## 使用 Filter Callback

通过使用 FILTER\_CALLBACK 过滤器，可以调用自定义的函数，把它作为一个过滤器来使用。这样，我们就拥有了数据过滤的完全控制权。

您可以创建自己的自定义函数，也可以使用已有的 PHP 函数。

规定您准备用到过滤器函数的方法，与规定选项的方法相同。

在下面的例子中，我们使用了一个自定义的函数把所有 "\_" 转换为空格：

<?php

function convertSpace($string)

{

return str\_replace("\_", " ", $string);

}

$string = "Peter\_is\_a\_great\_guy!";

echo filter\_var($string, FILTER\_CALLBACK, array("options"=>"convertSpace"));

?>

以上代码的结果是这样的：

Peter is a great guy!

### 例子解释：

上面的例子把所有 "\_" 转换成空格：

1. 创建一个把 "\_" 替换为空格的函数
2. 调用 filter\_var() 函数，它的参数是 FILTER\_CALLBACK 过滤器以及包含我们的函数的数组

# 总结：过滤器函数只是选择过滤变量的数量，重点是options参数，切换过滤的内容

如需完整的函数和过滤器列表，请访问我们的 [PHP Filter 参考手册](http://www.w3school.com.cn/php/php_ref_filter.asp)。