# PHP Filters

In this tutorial you will learn how to sanitize and validate user inputs in PHP.

## Validating and Sanitizing Data with Filters

Sanitizing and validating user input is one of the most common tasks in a web application. To make this task easier PHP provides native filter extension that you can use to sanitize or validate data such as e-mail addresses, URLs, IP addresses, etc.

To validate data using filter extension you need to use the PHP's `filter_var()` function. The basic syntax of this function can be given with:

```
filter_var(variable, filter, options)
```

This function takes three parameters out of which the last two are optional. The first parameter is the value to be filtered, the second parameter is the ID of the filter to apply, and the third parameter is the array of options related to filter. Let's see how it works.

## Sanitize a String

The following example will sanitize a string by removing all HTML tags from it:

**Example** **Run this code »**

```php
<?php
// Sample user comment
$comment = "<h1>Hey there! How are you doing today?</h1>";

// Sanitize and print comment string
$sanitizedComment = filter_var($comment, FILTER_SANITIZE_STRING);
echo $sanitizedComment;
?>
```

The output of the above example will look something like this:

```
Hey there! How are you doing today?
```

## Validate Integer Values

The following example will validate whether the value is a valid integer or not.

**Example** **Run this code »**

```php
<?php
// Sample integer value
$int = 20;

// Validate sample integer value
if(filter_var($int, FILTER_VALIDATE_INT)){
    echo "The <b>$int</b> is a valid integer";
} else{
    echo "The <b>$int</b> is not a valid integer";
}
?>
```

In the above example, if variable `$int` is set to 0, the example code will display invalid integer message. To fix this problem, you need to explicitly test for the value 0, as follow:

**Example** **Run this code »**

```php
<?php
// Sample integer value
$int = 0;

// Validate sample integer value
if(filter_var($int, FILTER_VALIDATE_INT) === 0 || filter_var($int,
FILTER_VALIDATE_INT)){
    echo "The <b>$int</b> is a valid integer";
} else{
    echo "The <b>$int</b> is not a valid integer";
}
?>
```

---

## Validate IP Addresses

The following example will validate whether the value is a valid IP address or not.

**Example** **Run this code »**

```php
<?php
// Sample IP address
```

```php
$ip = "172.16.254.1";

// Validate sample IP address
if(filter_var($ip, FILTER_VALIDATE_IP)){
    echo "The <b>$ip</b> is a valid IP address";
} else {
    echo "The <b>$ip</b> is not a valid IP address";
}
?>
```

You can further apply validation for IPV4 or IPV6 IP addresses by using the `FILTER_FLAG_IPV4` or `FILTER_FLAG_IPV6` flags, respectively. Here's an example:

**Example** **Run this code »**

```php
<?php
// Sample IP address
$ip = "172.16.254.1";

// Validate sample IP address
if(filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6)){
    echo "The <b>$ip</b> is a valid IPV6 address";
} else {
    echo "The <b>$ip</b> is not a valid IPV6 address";
}
?>
```

## Sanitize and Validate Email Addresses

The following example will show you how to sanitize and validate an e-mail address.

**Example** **Run this code »**

```php
<?php
// Sample email address
$email = "someone@@example.com";

// Remove all illegal characters from email
$sanitizedEmail = filter_var($email, FILTER_SANITIZE_EMAIL);

// Validate email address
```

```php
if($email == $sanitizedEmail && filter_var($email, FILTER_VALIDATE_EMAIL)){
    echo "The $email is a valid email address";
} else{
    echo "The $email is not a valid email address";
}
?>
```

**Note:** The `FILTER_SANITIZE_EMAIL` filter removes all invalid characters from the provided email address string except letters, digits and `!#$%&'*+-=?^_`{|}~@.[]`.

## Sanitize and Validate URLs

The following example will show you how to sanitize and validate a url.

**Example** **Run this code »**

```php
<?php
// Sample website url
$url = "http:://www.example.com";

// Remove all illegal characters from url
$sanitizedUrl = filter_var($url, FILTER_SANITIZE_URL);

// Validate website url
if($url == $sanitizedUrl && filter_var($url, FILTER_VALIDATE_URL)){
    echo "The $url is a valid website url";
} else{
    echo "The $url is not a valid website url";
}
?>
```

**Note:** The `FILTER_SANITIZE_URL` filter removes all invalid characters from the provided URL string except letters, digits and `$-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=.`

You can also check whether a URL contains query string or not by using the flag `FILTER_FLAG_QUERY_REQUIRED`, as shown in the following example:

**Example** **Run this code »**

```php
<?php
// Sample website url
$url = "http://www.example.com?topic=filters";

// Validate website url for query string
if(filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_REQUIRED))
{
    echo "The <b>$url</b> contains query string";
} else{
    echo "The <b>$url</b> does not contain query string";
}
?>
```

See the tutorial on HTML URL to learn about the different components of a URL.

---

## Validate Integers Within a Range

The following example will validate whether the supplied value is an integer or not, as well as whether it lies within the range of 0 to 100 or not.

**Example** **Run this code »**

```php
<?php
// Sample integer value
$int = 75;

// Validate sample integer value
if(filter_var($int, FILTER_VALIDATE_INT, array("options" =>
array("min_range" => 0,"max_range" => 100)))){
    echo "The <b>$int</b> is within the range of 0 to 100";
} else{
    echo "The <b>$int</b> is not within the range of 0 to 100";
}
?>
```