PHP filter_input

Summary: in this tutorial, you will learn how to use the PHP `filter_input()` function to get an external variable by name and filter it.

Introduction to PHP filter_input() function

The PHP `filter_input()` function allows you to get an external variable by its name and filter it using one or more built-in filters.

The following shows the syntax of the `filter_input()` function:

```
filter_input ( int $type , string $var_name , int $filter = FILTER_DEFAULT ,
array|int $options = 0 ) : mixed
Code language: PHP (php)
```

The `filter_input()` function has the following parameters:

- `$type` is one of `INPUT_GET`, `INPUT_POST`, `INPUT_COOKIE`, `INPUT_SERVER`, and `INPUT_ENV`.
- `$var_name` is the name of the variable to filter.
- `$filter` is the filter id to apply. Here's the list of valid filters. If you omit the `$filter` argument, the `filter_input()` function will use the `FILTER_DEFAULT` filter id, which doesn't filter anything.
- `$options` is an associative array that consists of one or more options. When a filter accepts the options, you can use one or more flags. If you want to use multiple flags, you need to separate them by the (`|`) e.g., `FILTER_SANITIZE_ENCODED | FILTER_SANITIZE_SPECIAL_CHARS`.

The `filter_input()` function returns `null`, `false`, or the filtered value according to the following rules:

- If the `$var_name` is not set, the `filte_input()` function returns `null`.
- If the filter fails, the `filter_input()` function returns `false`.
- Otherwise, it returns the filtered value of the requested variable.

PHP filter_input() function example

The following example uses the `filter_input()` function to sanitize data for a search form:

```
<?php

$term_html = filter_input(INPUT_GET, 'term', FILTER_SANITIZE_SPECIAL_CHARS);
$term_url = filter_input(INPUT_GET, 'term', FILTER_SANITIZE_ENCODED);

?>
<form action="search.php" method="get">
    <label for="term"> Search </label>
    <input type="search" name="term" id="term" value="<?php echo $term_html ?>">
    <input type="submit" value="Search">
</form>
```

```
<?php

if (null !== $term_html) {
        echo "The search result for <mark> $term_html </mark>.";
}
Code language: HTML, XML (xml)
```

How the form works.

The form contains an input with type `search` and a submit button.

When you enter a search term, e.g., `how to use the filter_input function` and click the submit button; the form uses the GET method to append the term query string to the URL, e.g.,

```
http://localhost/search.php?term=how+to+use+the+filter_input+function
Code language: plaintext (plaintext)
```

This search form submits to itself (`search.php`).

The `filter_input()` function sanitizes the search term using the `FILTER_SANITIZE_SPECIAL_CHARS` and `FILTER_SANITIZE_ENCODED` filters.

The `FILTER_SANITIZE_SPECIAL_CHARS` filter returns a value for showing on the search field and the `FILTER_SANITIZE_ENCODED` filter returns a value for displaying on the page.

filter_input vs. filter_var

If a variable doesn’t exist, the `filter_input()` function returns `null` while the `filter_var()` function returns an empty string and issues a notice of an undefined index.

Suppose you have a page with the following URL:

```
http://localhost/search.php
Code language: JavaScript (javascript)
```

The following `filter_input()` function returns `null` and doesn’t raise any error when you get the `term` variable from the INPUT_GET:

```
<?php

$term = filter_input(INPUT_GET, 'term', FILTER_SANITIZE_SPECIAL_CHARS);

var_dump($term);
Code language: HTML, XML (xml)
```

Output:

```
NULL
Code language: plaintext (plaintext)
```

However, the `filter_var()` function returns an empty string and issues an error:

```
<?php

$term = filter_var($_GET['term'], FILTER_SANITIZE_SPECIAL_CHARS);
var_dump($term);
Code language: HTML, XML (xml)
```

Output:

```
Notice: Undefined index: term in ...\search.php on line 3
string(0) ""
Code language: plaintext (plaintext)
```

Therefore, you often use the `isset()` or [`filter_has_var()`]() function to check if a variable is set before passing it to the `filter_var()` function like this:

```
<?php

if (isset($_GET['term'])) {
    $term = filter_var($_GET['term'], FILTER_SANITIZE_SPECIAL_CHARS);
    var_dump($term);
}
Code language: HTML, XML (xml)
```

Also, the `filter_input()` function doesn't get the current values of the `$_GET`, `$_POST`, … superglobal variables. Instead, it uses the original values submitted in the HTTP request. For example:

```
<?php

$_GET['term'] = 'PHP'; // doesn't have any effect on INPUT_GET
$term = filter_input(INPUT_GET, 'term', FILTER_SANITIZE_SPECIAL_CHARS);

var_dump($term);
Code language: PHP (php)
```

Output:

```
NULL
Code language: plaintext (plaintext)
```

This example attempts to assign a value to the `$_GET['term']` variable. However, the `filter_input()` doesn't read the term from the current `$_GET` variable. Therefore, the script displays NULL.

On the other hand, the `filter_var()` function does read values from the current `$_GET` variable. For example:

```
<?php

$_GET['term'] = 'PHP';
$term = filter_var($_GET['term'], FILTER_SANITIZE_SPECIAL_CHARS);

var_dump($term);
Code language: HTML, XML (xml)
```

Output:

```
string(3) "PHP"
Code language: JavaScript (javascript)
```

Summary

- Use the PHP `filter_input()` function to sanitze and validate data from external variables.

https://www.phptutorial.net/php-tutorial/php-filter_input/

PHP | filter_input() Function

- Last Updated : 14 Feb, 2019

The filter_input() is an inbuilt function in PHP which is used to get the specific external variable by name and filter it. This function is used to validate variables from insecure sources, such as user input from form. This function is very much useful to prevent some potential security threat like SQL Injection.

Syntax:

```
filter_input( $type, $variable_name, $filter, $options)
```

Parameters: This function accepts four parameters as mentioned above and described below:

- **$type:** It is mandatory parameter and used to check the type of input. The list of filters are:
    - INPUT_GET
    - INPUT_POST
    - INPUT_COOKIE
    - INPUT_SERVER
    - INPUT_ENV
- **$variable_name:** It is required parameter. It is used to hols the name of variable which is to be checked.
- **$filter:** It is an optional parameter. It holds the name or ID of the filter. If this parameter is not set then FILTER_DEFAULT is used.
- **$options:** It is an optional parameter and used to specify one or more flags/options to use. It check for possible options and flags in each filter. If filter options are accepted then flags can be provided in “flags” field of array.

Return Value: It returns the value of the variable on success or False on failure. If parameter is not set then return NULL. If the flag FILTER_NULL_ON_FAILURE is used, it returns FALSE if the variable is not set and NULL if the filter fails.

Example 1:

```
<?php
// PHP program to validate email using filter

if (isset($_GET["email"])) {
    if (!filter_input(INPUT_GET, "email",
            FILTER_VALIDATE_EMAIL) === false) {
        echo("Valid Email");
    } else {
        echo("Invalid Email");
    }
}

?>
```

Output:

```
Valid Email
```

Example 2:

```php
<?php

// Input type:INPUT_GET input name:search
// filter name:FILTER_SANITIZE_SPECIAL_CHARS
$search_variable_data = filter_input(INPUT_GET,
            'search', FILTER_SANITIZE_SPECIAL_CHARS);

// Input type:INPUT_GET input name:search
// filter name:FILTER_SANITIZE_ENCODED
$search_url_data = filter_input(INPUT_GET,
            'search', FILTER_SANITIZE_ENCODED);

echo "Search for $search_variable_data.\n";

echo "<a href='?search=$search_url_data'>Search again.</a>";

?>
```

Output:

```
Search for tic tac & toc. Search again.
```

References: http://php.net/manual/en/function.filter-input.php

https://www.geeksforgeeks.org/php-filter_input-function/