PHP - Sanitize Input

In PHP, it is important to ensure that the input data is sanitized properly by removed any undesired characters before it is processed by the server side code. Usually, the users input their data to a PHP web application through a HTML form. If the form data consists of any undesired characters, it may prove to be harmful, hence an appropriate cleansing operation must be performed.

Input sanitization can be done with the help of one or more of the following functions in PHP.

The htmlspecialchars() Function

This function converts special characters to HTML entities.

```
htmlspecialchars(
   string $string,
   int $flags = ENT_QUOTES | ENT_SUBSTITUTE | ENT_HTML401,
   ?string $encoding = null,
   bool $double_encode = true
): string
```

In HTML, certain characters have special significance. This htmlspecialchars() function is used to encode special characters in HTML entities. This is useful when you want to display user input as HTML and want to prevent script injection attacks.

The following **special characters** are translated as shown –

Character	Replaced by
& (ampersand)	&amp;
" (double quote)	", unless **ENT_NOQUOTES** is set
' (single quote)	' (for **ENT_HTML401**) or ' (for **ENT_XML1, ENT_XHTML** or **ENT_HTML5**), but only when **ENT_QUOTES** is set
< (less than)	&lt;
> (greater than)	&gt;

Flag Constants

The **flags** parameter is a bitmask of one or more of the following flags, which specify how to handle quotes, invalid code unit sequences and the used document type.

Sr.No	Constant & Description
1	**ENT_COMPAT** Will convert double-quotes and leave single-quotes alone.
2	**ENT_QUOTES** Will convert both double and single quotes.
3	**ENT_NOQUOTES** Will leave both double and single quotes unconverted.
4	**ENT_IGNORE** discard invalid code unit sequences instead of returning an empty string.
5	**ENT_SUBSTITUTE** Replace invalid code unit sequences with a Unicode Replacement Character U+FFFD (UTF-8) or &#xFFFD;
6	**ENT_DISALLOWED** Replace invalid code points for the given document type with a Unicode Replacement Character U+FFFD (UTF-8) or &#xFFFD; (otherwise) instead of leaving them as is. This may be useful.
7	**ENT_HTML401** Handle code as HTML 4.01.
8	**ENT_XML1** Handle code as XML 1.
9	**ENT_XHTML** Handle code as XHTML.
10	**ENT_HTML5** Handle code as HTML 5.

Example

Take a look at the following example –

```php
<?php
   $str = 'Welcome To "PHP Tutorial" by <b>TutorialsPoint</b>';
   echo htmlspecialchars($str);
?>
```

It will produce the following **output** –

```
Welcome To "PHP Tutorial" by <b>TutorialsPoint</b>
```

The strip_tags() Function

The strip_tags() function removes all the HTML and PHP tags from a given string.

```php
strip_tags(string $string, array|string|null $allowed_tags = null): string
```

This function is very useful when you want ensure that the user input doesn't contain any potentially malicious tags.

The **allowed_tags** parameter is an optional second parameter to specify tags which should not be stripped. These are either given as string, or as an array.

Example

Take a look at the following example –

</> Open Compiler

```php
<?php
   $text = '<p>Hello World</p><!-- Comment -->
      <a href="/test.html">Click Here</a>';
   echo strip_tags($text);
   echo "\n";

   // Allow <p> and <a>
   echo strip_tags($text, '<p><a>');
?>
```

It will produce the following **output** –

```
Hello World
    Click Here
Hello World



    Click Here
```


The addslashes() Function

The addslashes() function adds backslashes to a string.

```
addslashes(string $string): string
```

The function returns a string with backslashes added before characters that need to be escaped. These characters are –

- Single Quote (')
- Double Quote (")
- Backslash (\)
- NUL (The NUL Byte)

Use this function when you are storing user input in a database and want to prevent SQL injection attacks.

Example

Take a look at the following example –

</> Open Compiler

```
<?php
   $text = "Newton's Laws";
   $str = addslashes($text);

   // prints the escaped string
```

```
   echo($str);
?>
```

It will produce the following **output** –

```
Newton\'s Laws
```

The filter_var() Function

With the help of a specific filter flag, you can use filter_var() function to sanitize user input.

```
filter_var(mixed $value, int $filter =
   FILTER_DEFAULT, array|int $options = 0): mixed
```

The $value parameter is a variable whose value needs to be sanitized. The $filter parameter is any of the predefined filter constants.

Sr.No	ID & Description
1	**FILTER_SANITIZE_EMAIL** Remove all characters except letters, digits and !#$%&'*+-=?^_`{|}~@.[].
2	**FILTER_SANITIZE_ENCODED** URL-encode string, optionally strip or encode special characters.
3	**FILTER_SANITIZE_ADD_SLASHES** Apply addslashes(). (Available as of PHP 7.3.0).
4	**FILTER_SANITIZE_NUMBER_FLOAT** Remove all characters except digits, +- and optionally .,eE.
5	**FILTER_SANITIZE_NUMBER_INT** Remove all characters except digits, plus and minus sign.
6	**FILTER_SANITIZE_SPECIAL_CHARS** HTML-encode '"<>& and characters with ASCII value less than 32, optionally strip or encode other special characters.
7	**FILTER_SANITIZE_FULL_SPECIAL_CHARS** Equivalent to calling htmlspecialchars() with **ENT_QUOTES** set. Encoding quotes can be disabled by setting **FILTER_FLAG_NO_ ENCODE_QUOTES**.

8	**FILTER_SANITIZE_URL** Remove all characters except letters, digits and $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=.
9	**FILTER_UNSAFE_RAW**

Example

The following code shows how you can sanitize Email data –

```
</>                                                    Open Compiler
<?php
   $a = 'abc def@xyz.com';

   $sa = filter_var($a, FILTER_SANITIZE_EMAIL);
   echo "$sa";
?>
```

It will produce the following **output** –

```
abcdef@xyz.com
```

Example

The following code shows how you can sanitize URLs –

```
</>                                                    Open Compiler
<?php
   $a = "http://example.c o m";

   $sa = filter_var($a, FILTER_SANITIZE_URL);
   echo "$sa";
?>
```

It will produce the following **output** –

```
http://example.com
```