This PHP filters is used to validate and filter data coming from insecure sources, like user input.

Validate filters

Listing of filters for validation				
ID	**Name**	**Options**	**Flags**	**Description**
FILTER_VALIDATE_BOOLEAN	"boolean"	default	**FILTER_NULL_ON_FAILURE**	Returns **TRUE** for "1", "true", "on" and "yes". Returns **FALSE** otherwise. If **FILTER_NULL_ON_FAILURE** is set, **FALSE** is returned only for "0", "false", "off", "no", and "", and **NULL** is returned for all non-boolean values.
FILTER_VALIDATE_EMAIL	"validate_email"	default		Validates value as e-mail.
FILTER_VALIDATE_FLOAT	"float"	default, decimal	**FILTER_FLAG_ALLOW_THOUSAND**	Validates value as float.
FILTER_VALIDATE_INT	"int"	default, min_range, max_range	**FILTER_FLAG_ALLOW_OCTAL**, **FILTER_FLAG_ALLOW_HEX**	Validates value as integer, optionally from the specified range.
FILTER_VALIDATE_IP	"validate_ip"	default	**FILTER_FLAG_IPV4**, **FILTER_FLAG_IPV6**, **FILTER_FLAG_NO_PRIV_RANGE**, **FILTER_FLAG_NO_RES_RANGE**	Validates value as IP address, optionally only IPv4 or IPv6 or not from private or reserved ranges.
FILTER_VALIDATE_REGEXP	"validate_regexp"	default, regexp		Validates value against `regexp`, a Perl-compatible regular expression.
FILTER_VALIDATE_URL	"validate_url"	default	**FILTER_FLAG_PATH_REQUIRED**, **FILTER_FLAG_QUERY_REQUIRED**	Validates value as URL (according to » http://www.faqs.org/rfcs/rfc2396), optionally with required components. Beware a valid URL may not specify the HTTP protocol `http://` so

Listing of filters for validation				
ID	**Name**	**Options**	**Flags**	**Description**
				further validation may be required to determine the URL uses an expected protocol, e.g. `ssh://` or `mailto:`. Note that the function will only find ASCII URLs to be valid; internationalized domain names (containing non-ASCII characters) will fail.

Sanitize filters

List of filters for sanitization				
ID	**Name**	**Options**	**Flags**	**Description**
FILTER_SANITIZE_EMAIL	"email"			Remove all characters except letters, digits and *!#$%&'*+-/=?^_`{|}~@.[]*.
FILTER_SANITIZE_ENCODED	"encoded"		**FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_ENCODE_LOW**, **FILTER_FLAG_ENCODE_HIGH**	URL-encode string, optionally strip or encode special characters.
FILTER_SANITIZE_MAGIC_QUOTES	"magic_quotes"			Apply addslashes().
FILTER_SANITIZE_NUMBER_FLOAT	"number_float"		**FILTER_FLAG_ALLOW_FRACTION**, **FILTER_FLAG_ALLOW_THOUSAND**, **FILTER_FLAG_ALLOW_SCIENTIFIC**	Remove all characters except digits, +- and optionally .,*eE*.
FILTER_SANITIZE_NUMBER_INT	"number_int"			Remove all characters except digits, plus and minus sign.
FILTER_SANITIZE_SPECIAL_CHARS	"special_chars"		**FILTER_FLAG_STRIP_LOW**,	HTML-escape '"<>& and characters with

List of filters for sanitization				
ID	**Name**	**Options**	**Flags**	**Description**
			FILTER_FLAG_STRIP_HIGH, **FILTER_FLAG_ENCODE_HIGH**	ASCII value less than 32, optionally strip or encode other special characters.
FILTER_SANITIZE_FULL_SPECIAL_CHARS	"full_special_chars"		**FILTER_FLAG_NO_ENCODE_QUOTES**,	Equivalent to calling htmlspecialchars() with **ENT_QUOTES** set. Encoding quotes can be disabled by setting **FILTER_FLAG_NO_ENCODE_QUOTES**. Like htmlspecialchars(), this filter is aware of the default_charset and if a sequence of bytes is detected that makes up an invalid character in the current character set then the entire string is rejected resulting in a 0-length string. When using this filter as a default filter, see the warning below about setting the default flags to 0.
FILTER_SANITIZE_STRING	"string"		**FILTER_FLAG_NO_ENCODE_QUOTES**, **FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_ENCODE_LOW**, **FILTER_FLAG_ENCODE_HIGH**, **FILTER_FLAG_ENCODE_AMP**	Strip tags, optionally strip or encode special characters.
FILTER_SANITIZE_STRIPPED	"stripped"			Alias of "string" filter.
FILTER_SANITIZE_URL	"url"			Remove all characters except letters, digits and $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=

List of filters for sanitization				
ID	**Name**	**Options**	**Flags**	**Description**
				_.+!'(),{}|\\^~[]`<>#%";/?:@&=.*
FILTER_UNSAFE_RAW	"unsafe_raw"		**FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_ENCODE_LOW, FILTER_FLAG_ENCODE_HIGH, FILTER_FLAG_ENCODE_AMP**	

Example 1: for FILTER_VALIDATE_EMAIL

```
<?php
$email = "someone@exa mple.com";

if(!filter_var($email, FILTER_VALIDATE_EMAIL))
 {
 echo "E-mail is not valid";
 }
else
 {
 echo "E-mail is valid";
 }
?>
```

The output of the code will be:

E-mail is not valid

The FILTER_VALIDATE_URL filter validates value as a URL.

Possible flags:

- FILTER_FLAG_SCHEME_REQUIRED - Requires URL to be an RFC compliant URL (like http://example)
- FILTER_FLAG_HOST_REQUIRED - Requires URL to include host name (like http://www.example.com)

- FILTER_FLAG_PATH_REQUIRED - Requires URL to have a path after the domain name (like www.example.com/example1/test2/)
- FILTER_FLAG_QUERY_REQUIRED - Requires URL to have a query string (like "example.php?name=Peter&age=37")

Example 1

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

if(!filter_var($url, FILTER_VALIDATE_URL))
  {
  echo "URL is not valid";
  }
else
  {
  echo "URL is valid";
  }
?>
```

The output of the code will be:
URL is valid

Example 2

```
<?php
$url = "example.php?name=Peter&age=37";

if(!filter_var($url, FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED))
  {
  echo "URL is not valid";
  }
else
  {
  echo "URL is valid";
  }
?>
```

The output of the code will be:
URL is valid

The FILTER_VALIDATE_REGEXP filter validates value against a Perl-compatible regular expression.

```
<?php
$string = "Match this string";

var_dump(filter_var($string, FILTER_VALIDATE_REGEXP,
array("options"=>array("regexp"=>"/^M(.*)/"))))
?>
```

The output of the code will be:

string(17) "Match this string"

The FILTER_VALIDATE_BOOLEAN filter validates value as a boolean option.

```
<?php
$var="yes";

var_dump(filter_var($var, FILTER_VALIDATE_BOOLEAN));
?>
```

The output of the code will be:

bool(true)

The FILTER_SANITIZE_URL filter removes all illegal URL characters from a string.

This filter allows all letters, digits and $-_.+!*'(),{}|\\^~[]`"><#%;/?:@&=

```
<?php
$var="http://www.w3schoo��ls.co�m";

var_dump(filter_var($var, FILTER_SANITIZE_URL));
?>
```

The output of the code will be:

string(24) "http://www.w3schools.com"

The FILTER_SANITIZE_EMAIL filter removes all illegal e-mail characters from a string.

This filter allows all letters, digits and $-_.+!*'{}|^~[]`#%/?@&=

```
<?php
$var="some(one)@exa\\mple.com";

var_dump(filter_var($var, FILTER_SANITIZE_EMAIL));
?>
```

The output of the code will be:

string(19) "someone@example.com"