Data Filtering

Introduction

This extension serves to validate and filter data coming from some insecure source, such as user input.

The following filters currently exist; be sure to read the Filter Constants section for information that describes the behavior of each constant:

Existing filters

ID	Name	Options	Flags	Description
FILTER_VALIDATE_INT	"int"	*min_range*, *max_range*	**FILTER_FLAG_ALLOW_OCTAL**, **FILTER_FLAG_ALLOW_HEX**	Validates value as integer, optionally from the specified range.
FILTER_VALIDATE_BOOLEAN	"boolean"		**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_FLOAT	"float"	*decimal*	**FILTER_FLAG_ALLOW_THOUSAND**	Validates value as float.
FILTER_VALIDATE_REGEXP	"validate_regexp"	*regexp*		Validates value against *regexp*, a Perl-compatible regular expression.
FILTER_VALID	**"validate_url"**		**FILTER_FLAG_**	**Validates value**

ATE_URL			**PATH_REQUIRED**, **FILTER_FLAG_QUERY_REQUIRED**	as URL, optionally with required components.
FILTER_VALIDATE_EMAIL	"validate_email"			Validates value as e-mail.
FILTER_VALIDATE_IP	"validate_ip"		**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_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_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_SPECIAL_CHARS	"special_chars"		**FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_ENCODE_HIGH**	HTML-escape '"<>&and characters with ASCII value less than 32, optionally strip or

				encode other special characters.
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**	Do nothing, optionally strip or encode special characters.
FILTER_SANITIZE_EMAIL	"email"			Remove all characters except letters, digits and *!#$%&'*+-/=?^_`{|}~@.[]*.
FILTER_SANITIZE_URL	"url"			Remove all characters except letters, digits and *$-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=*.
FILTER_SANITIZE_NUMBER_INT	"number_int"			Remove all characters except digits, plus and minus sign.
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_MAGIC_QUOTES	"magic_quotes"			Apply addslashes().
FILTER_CALLBACK	"callback"	callback function or method		Call user-defined function to filter data.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

A short installation note: just type

```
$ pecl install filter
```

in your console.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

Filter Configuration Options

Name	Default	Changeable	Changelog
filter.default	"unsafe_raw"	PHP_INI_PERDIR	PHP_INI_ALL in filter <= 0.9.4. Available since PHP 5.2.0.
filter.default_flags	NULL	PHP_INI_PERDIR	PHP_INI_ALL in filter <= 0.9.4. Available since PHP 5.2.0.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

`filter.default` string
: Filter all $_GET, $_POST, $_COOKIE and $_REQUEST data by this filter. Original data can be accessed through filter_input(). Accepts the name of the filter you like to use by default. See the existing filter list for the list of the filter names.

`filter.default_flags` integer
: Default flags

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

INPUT_POST (integer)
POST variables.

INPUT_GET (integer)
GET variables.

INPUT_COOKIE (integer)
COOKIE variables.

INPUT_ENV (integer)
ENV variables.

INPUT_SERVER (integer)
SERVER variables.

INPUT_SESSION (integer)
SESSION variables. (not implemented yet)

INPUT_REQUEST (integer)
REQUEST variables. (not implemented yet)

FILTER_FLAG_NONE (integer)
No flags.

FILTER_REQUIRE_SCALAR (integer)
Flag used to require scalar as input

FILTER_REQUIRE_ARRAY (integer)
Require an array as input.

FILTER_FORCE_ARRAY (integer)
Always returns an array.

FILTER_NULL_ON_FAILURE (integer)
Use NULL instead of FALSE on failure.

FILTER_VALIDATE_INT (integer)
ID of "int" filter.

FILTER_VALIDATE_BOOLEAN (integer)
ID of "boolean" filter.

FILTER_VALIDATE_FLOAT (integer)
ID of "float" filter.

FILTER_VALIDATE_REGEXP (integer)
ID of "validate_regexp" filter.

FILTER_VALIDATE_URL (integer)
ID of "validate_url" filter.

FILTER_VALIDATE_EMAIL (integer)
ID of "validate_email" filter.

FILTER_VALIDATE_IP (integer)
ID of "validate_ip" filter.

FILTER_DEFAULT (integer)
ID of default ("string") filter.

FILTER_UNSAFE_RAW (integer)
ID of "unsafe_raw" filter.

FILTER_SANITIZE_STRING (integer)
ID of "string" filter.

FILTER_SANITIZE_STRIPPED (integer)
ID of "stripped" filter.

FILTER_SANITIZE_ENCODED (integer)
ID of "encoded" filter.

FILTER_SANITIZE_SPECIAL_CHARS (integer)
ID of "special_chars" filter.

FILTER_SANITIZE_EMAIL (integer)
ID of "email" filter.

FILTER_SANITIZE_URL (integer)
ID of "url" filter.

FILTER_SANITIZE_NUMBER_INT (integer)
ID of "number_int" filter.

FILTER_SANITIZE_NUMBER_FLOAT (integer)
ID of "number_float" filter.

FILTER_SANITIZE_MAGIC_QUOTES (integer)
ID of "magic_quotes" filter.

FILTER_CALLBACK (integer)
ID of "callback" filter.

FILTER_FLAG_ALLOW_OCTAL (integer)
Allow octal notation (*0[0-7]+*) in "int" filter.

FILTER_FLAG_ALLOW_HEX (integer)

Allow hex notation (*0x[0-9a-fA-F]+*) in "int" filter.

FILTER_FLAG_STRIP_LOW (integer)
Strip characters with ASCII value less than 32.

FILTER_FLAG_STRIP_HIGH (integer)
Strip characters with ASCII value greater than 127.

FILTER_FLAG_ENCODE_LOW (integer)
Encode characters with ASCII value less than 32.

FILTER_FLAG_ENCODE_HIGH (integer)
Encode characters with ASCII value greater than 127.

FILTER_FLAG_ENCODE_AMP (integer)
Encode *&*.

FILTER_FLAG_NO_ENCODE_QUOTES (integer)
Don't encode *'* and *"*.

FILTER_FLAG_EMPTY_STRING_NULL (integer)
(No use for now.)

FILTER_FLAG_ALLOW_FRACTION (integer)
Allow fractional part in "number_float" filter.

FILTER_FLAG_ALLOW_THOUSAND (integer)
Allow thousand separator (*,*) in "number_float" filter.

FILTER_FLAG_ALLOW_SCIENTIFIC (integer)
Allow scientific notation (*e*, *E*) in "number_float" filter.

FILTER_FLAG_SCHEME_REQUIRED (integer)
Require scheme in "validate_url" filter.

FILTER_FLAG_HOST_REQUIRED (integer)
Require host in "validate_url" filter.

FILTER_FLAG_PATH_REQUIRED (integer)
Require path in "validate_url" filter.

FILTER_FLAG_QUERY_REQUIRED (integer)
Require query in "validate_url" filter.

FILTER_FLAG_IPV4 (integer)
Allow only IPv4 address in "validate_ip" filter.

FILTER_FLAG_IPV6 (integer)
Allow only IPv6 address in "validate_ip" filter.

FILTER_FLAG_NO_RES_RANGE (integer)
Deny reserved addresses in "validate_ip" filter.

FILTER_FLAG_NO_PRIV_RANGE (integer)
Deny private addresses in "validate_ip" filter.

Filter Functions

filter_has_var

filter_has_var -- Checks if variable of specified type exists

Description

bool **filter_has_var** (int `$type`, string `$variable_name`)

Parameters

`type`
: One of **INPUT_GET**, **INPUT_POST**, **INPUT_COOKIE**, **INPUT_SERVER**, **INPUT_ENV**.

`variable_name`
: Name of a variable to check.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

filter_id

filter_id -- Returns the filter ID belonging to a named filter

Description

int **filter_id** (string `$filtername`)

Parameters

`filtername`
: Name of a filter to get.

Return Values

ID of a filter on success or **NULL** if filter doesn't exist.

See Also

- filter_list()

filter_input_array

filter_input_array -- Gets external variables and optionally filters them

Description

mixed **filter_input_array** (int `$type` [, mixed `$definition`])

This function is useful for retrieving many values without repetitively calling filter_input().

Parameters

`type`
: One of **INPUT_GET**, **INPUT_POST**, **INPUT_COOKIE**, **INPUT_SERVER**, **INPUT_ENV**, **INPUT_SESSION**, or **INPUT_REQUEST**.

`definition`
: An array defining the arguments. A valid key is a string containing a variable name and a valid value is either a filter type, or an array optionally specifying the filter, flags and options. If the value is an array, valid keys are *filter* which specifies the filter type, *flags* which specifies any flags that apply to the filter, and *options* which specifies any options that apply to the filter. See the example below for a better understanding. This parameter can be also an integer holding a filter constant. Then all values in the input array are filtered by this filter.

Return Values

An array containing the values of the requested variables on success, or **FALSE** on failure. An array value will be **FALSE** if the filter fails, or **NULL** if the variable is not set. Or if the flag **FILTER_NULL_ON_FAILURE** is used, it returns **FALSE** if the variable is not set and **NULL** if the filter fails.

Examples

Example #1 - A filter_input_array() example

```
<?php
error_reporting(E_ALL | E_STRICT);
/* data actually came from POST
$_POST = array(
    'product_id'    => 'libgd<script>',
    'component'     => '10',
    'versions'      => '2.0.33',
    'testscalar'    => array('2', '23', '10', '12'),
    'testarray'     => '2',
);
```

```
*/

$args = array(
   'product_id'   => FILTER_SANITIZE_ENCODED,
   'component'    => array('filter'    => FILTER_VALIDATE_INT,
                           'flags'     => FILTER_REQUIRE_ARRAY,
                           'options'   => array('min_range' => 1,
'max_range' => 10)
                          ),
   'versions'     => FILTER_SANITIZE_ENCODED,
   'doesnotexist' => FILTER_VALIDATE_INT,
   'testscalar'   => array(
                           'filter' => FILTER_VALIDATE_INT,
                           'flags'  => FILTER_REQUIRE_SCALAR,
                          ),
   'testarray'    => array(
                           'filter' => FILTER_VALIDATE_INT,
                           'flags'  => FILTER_REQUIRE_ARRAY,
                          )

);

$myinputs = filter_input_array(INPUT_POST, $args);

var_dump($myinputs);
echo "\n";
?>
```

The above example will output:

```
array(6) {
 ["product_id"]=>
 array(1) {
   [0]=>
   string(17) "libgd%3Cscript%3E"
 }
 ["component"]=>
 array(1) {
   [0]=>
   int(10)
 }
 ["versions"]=>
 array(1) {
   [0]=>
   string(6) "2.0.33"
 }
 ["doesnotexist"]=>
 NULL
 ["testscalar"]=>
 bool(false)
 ["testarray"]=>
 array(1) {
   [0]=>
   int(2)
 }
}
```

See Also

- filter_input()
- filter_var_array()

filter_input

filter_input -- Gets a specific external variable by name and optionally filters it

Description

mixed **filter_input** (int `$type`, string `$variable_name` [, int `$filter` [, mixed `$options`]])

Parameters

`type`
: One of **INPUT_GET**, **INPUT_POST**, **INPUT_COOKIE**, **INPUT_SERVER**, **INPUT_ENV**, **INPUT_SESSION** (not implemented yet) and **INPUT_REQUEST** (not implemented yet).

`variable_name`
: Name of a variable to get.

`filter`
: Filter to apply. Defaults to **FILTER_DEFAULT**.

`options`
: Associative array of options or bitwise disjunction of flags. If filter accepts options, flags can be provided in "flags" field of array.

Return Values

Value of the requested variable on success, **FALSE** if the filter fails, or **NULL** if the *`variable_name`* variable is not set. If the flag **FILTER_NULL_ON_FAILURE** is used, it returns **FALSE** if the variable is not set and **NULL** if the filter fails.

Examples

Example #2 - A filter_input() example

```
<?php
$search_html = filter_input(INPUT_GET, 'search',
FILTER_SANITIZE_SPECIAL_CHARS);
$search_url = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_ENCODED);
echo "You have searched for $search_html.\n";
echo "<a href='?search=$search_url'>Search again.</a>";
?>
```

The above example will output something similar to:

```
You have searched for Me &#38; son.
```

```
<a href='?search=Me%20%26%20son'>Search again.</a>
```

See Also

- filter_var()
- filter_input_array()
- filter_var_array()

filter_list

filter_list -- Returns a list of all supported filters

Description

array **filter_list** (void)

Return Values

Returns an array of names of all supported filters, empty array if there are no such filters. Indexes of this array are not filter IDs, they can be obtained with filter_id() from a name instead.

Examples

Example #3 - A filter_list() example

```
<?php
print_r(filter_list());
?>
```

The above example will output something similar to:

```
Array
(
   [0] => int
   [1] => boolean
   [2] => float
   [3] => validate_regexp
   [4] => validate_url
   [5] => validate_email
   [6] => validate_ip
   [7] => string
   [8] => stripped
   [9] => encoded
   [10] => special_chars
   [11] => unsafe_raw
   [12] => email
   [13] => url
   [14] => number_int
   [15] => number_float
   [16] => magic_quotes
   [17] => callback
)
```

filter_var_array

filter_var_array -- Gets multiple variables and optionally filters them

Description

mixed **filter_var_array** (array `$data` [, mixed `$definition`])

This function is useful for retrieving many values without repetitively calling filter_var().

Parameters

`data`
An array with string keys containing the data to filter.

`definition`
An array defining the arguments. A valid key is a string containing a variable name and a valid value is either a filter type, or an array optionally specifying the filter, flags and options. If the value is an array, valid keys are *filter* which specifies the filter type, *flags* which specifies any flags that apply to the filter, and *options* which specifies any options that apply to the filter. See the example below for a better understanding. This parameter can be also an integer holding a filter constant. Then all values in the input array are filtered by this filter.

Return Values

An array containing the values of the requested variables on success, or **FALSE** on failure. An array value will be **FALSE** if the filter fails, or **NULL** if the variable is not set.

Examples

Example #4 - A filter_var_array() example

```
<?php
error_reporting(E_ALL | E_STRICT);
$data = array(
   'product_id'    => 'libgd<script>',
   'component'     => '10',
   'versions'      => '2.0.33',
   'testscalar'    => array('2', '23', '10', '12'),
   'testarray'     => '2',
);

$args = array(
   'product_id'   => FILTER_SANITIZE_ENCODED,
   'component'    => array('filter'    => FILTER_VALIDATE_INT,
```

```
                            'flags'     => FILTER_FORCE_ARRAY,
                            'options'   => array('min_range' => 1,
'max_range' => 10)
                           ),
   'versions'     => FILTER_SANITIZE_ENCODED,
   'doesnotexist' => FILTER_VALIDATE_INT,
   'testscalar'   => array(
                            'filter' => FILTER_VALIDATE_INT,
                            'flags'  => FILTER_REQUIRE_SCALAR,
                           ),
   'testarray'    => array(
                            'filter' => FILTER_VALIDATE_INT,
                            'flags'  => FILTER_FORCE_ARRAY,
                           )

);

$myinputs = filter_var_array($data, $args);

var_dump($myinputs);
echo "\n";
?>
```

The above example will output:

```
array(6) {
 ["product_id"]=>
 array(1) {
   [0]=>
   string(17) "libgd%3Cscript%3E"
 }
 ["component"]=>
 array(1) {
   [0]=>
   int(10)
 }
 ["versions"]=>
 array(1) {
   [0]=>
   string(6) "2.0.33"
 }
 ["doesnotexist"]=>
 NULL
 ["testscalar"]=>
 bool(false)
 ["testarray"]=>
 array(1) {
   [0]=>
   int(2)
 }
}
```

See Also

- filter_input_array()

- filter_var()
- filter_input()

filter_var

filter_var -- Filters a variable with a specified filter

Description

mixed **filter_var** (mixed `$variable` [, int `$filter` [, mixed `$options`]])

Parameters

`variable`
: Value to filter.

`filter`
: ID of a filter to use. Defaults to **FILTER_SANITIZE_STRING**.

`options`
: Associative array of options or bitwise disjunction of flags. If filter accepts options, flags can be provided in "flags" field of array. For the "callback" filter, callback type should be passed.

Return Values

Returns the filtered data, or **FALSE** if the filter fails.

Examples

Example #5 - A filter_var() example

```
<?php
var_dump(filter_var('bob@example.com', FILTER_VALIDATE_EMAIL));
var_dump(filter_var('example.com', FILTER_VALIDATE_URL,
FILTER_FLAG_SCHEME_REQUIRED));
?>
```

The above example will output:

```
string(15) "bob@example.com"
bool(false)
```

See Also

- filter_var_array()
- filter_input()
- filter_input_array()
- information about the callback type