🎬 Filter Functions

- filter_has_var — Checks if variable of specified type exists
- filter_id — Returns the filter ID belonging to a named filter
- filter_input_array — Gets external variables and optionally filters them
- filter_input — Gets a specific external variable by name and optionally filters it
- filter_list — Returns a list of all supported filters
- filter_var_array — Gets multiple variables and optionally filters them
- filter_var — Filters a variable with a specified filter

filter_var

(PHP 5 >= 5.2.0)

filter_var — Filters a variable with a specified filter

Description ¶

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

Parameters ¶

`variable`

Value to filter.

`filter`

The ID of the filter to apply. The Types of filters manual page lists the available filters.

`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, callable type should be passed. The callback must accept one argument, the value to be filtered, and return the value after filtering/sanitizing it.

```
<?php
// for filters that accept options, use this format
$options = array(
    'options' => array(
        'default' => 3, // value to return if the filter fails
        // other options here
        'min_range' => 0
    ),
    'flags' => FILTER_FLAG_ALLOW_OCTAL,
);
$var = filter_var('0755', FILTER_VALIDATE_INT, $options);

// for filter that only accept flags, you can pass them directly
$var = filter_var('oops', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE
);

// for filter that only accept flags, you can also pass as an array
$var = filter_var('oops', FILTER_VALIDATE_BOOLEAN,
                  array('flags' => FILTER_NULL_ON_FAILURE));

// callback validate filter
function foo($value)
{
    // Expected format: Surname, GivenNames
    if (strpos($value, ", ") === false) return false;
    list($surname, $givennames) = explode(", ", $value, 2);
    $empty = (empty($surname) || empty($givennames));
    $notstrings = (!is_string($surname) || !is_string($givennames));
    if ($empty || $notstrings) {
        return false;
    } else {
        return $value;
    }
}
$var = filter_var('Doe, Jane Sue', FILTER_CALLBACK, array('options' => 'f
oo'));
?>
```

Return Values ¶

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

Examples ¶

Example #1 A filter_var() example

```
<?php
var_dump(filter_var('bob@example.com', FILTER_VALIDATE_EMAIL));
var_dump(filter_var('http://example.com', FILTER_VALIDATE_URL, FILTER_FLAG_PATH
_REQUIRED));
?>
```

The above example will output:

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

```
bool(false)
```

Example #1 Sanitizing and validating email addresses

```php
<?php
$a = 'joe@example.org';
$b = 'bogus - at - example dot org';
$c = '(bogus@example.org)';

$sanitized_a = filter_var($a, FILTER_SANITIZE_EMAIL);
if (filter_var($sanitized_a, FILTER_VALIDATE_EMAIL)) {
    echo "This (a) sanitized email address is considered valid.\n";
}

$sanitized_b = filter_var($b, FILTER_SANITIZE_EMAIL);
if (filter_var($sanitized_b, FILTER_VALIDATE_EMAIL)) {
    echo "This sanitized email address is considered valid.";
} else {
    echo "This (b) sanitized email address is considered invalid.\n";
}

$sanitized_c = filter_var($c, FILTER_SANITIZE_EMAIL);
if (filter_var($sanitized_c, FILTER_VALIDATE_EMAIL)) {
    echo "This (c) sanitized email address is considered valid.\n";
    echo "Before: $c\n";
    echo "After:  $sanitized_c\n";
}
?>
```

The above example will output:

```
This (a) sanitized email address is considered valid.
This (b) sanitized email address is considered invalid.
This (c) sanitized email address is considered valid.
Before: (bogus@example.org)
After: bogus@example.org
```

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_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.