

Welcome!



Share your information

What is Wrong Here?

```
<?php
    $sql = "
        SELECT card_num, card_name, card_expiry
        FROM credit_cards
        WHERE uid = '{$_GET['uid']}'
    ";
?>
```

<http://example.com/script.php?uid=42>

```
SELECT card_num, card_name, card_expiry
FROM credit_cards
WHERE uid = '42'
```

!

<http://example.com/script.php?uid=42'%20or%20''='>

```
SELECT card_num, card_name, card_expiry
FROM credit_cards
WHERE uid = '42' or ''=''
```

What is Wrong Here?

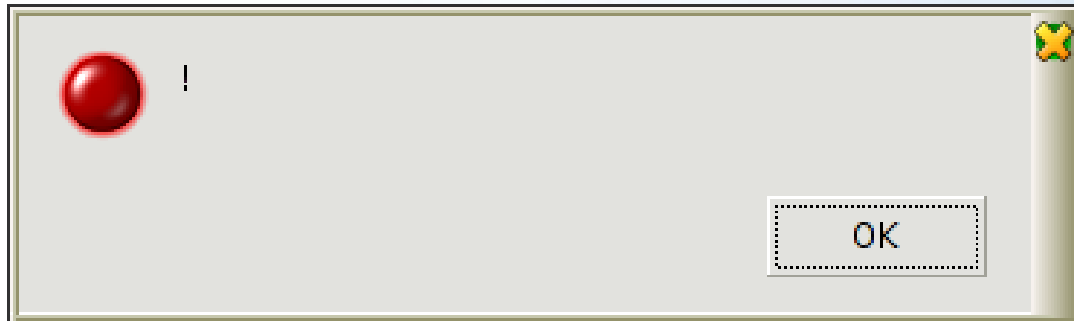
```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    Name: <?php echo $_GET['name']; ?>
  </body>
</html>
```

<http://example.com/script.php?name=derick>

Name: derick

!

[http://example.com/script.php?name=<script>alert\('!'\);</script>](http://example.com/script.php?name=<script>alert('!');</script>)



```
<?php
    $uid = (int) $_GET['uid'];
    $sql = "
        SELECT card_num, card_name, card_expiry
        FROM credit_cards
        WHERE uid = '{$uid}'
    ";
?>
```

<http://example.com/script.php?uid=42>

```
SELECT card_num, card_name, card_expiry
FROM credit_cards
WHERE uid = '42'
```

:-)

<http://example.com/script.php?uid=42'%20or%20''='>

```
SELECT card_num, card_name, card_expiry
FROM credit_cards
WHERE uid = '42'
```

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    Name: <?php echo htmlentities($_GET['name']); ?>
  </body>
</html>
```

<http://example.com/script.php?name=derick>

Name: derick

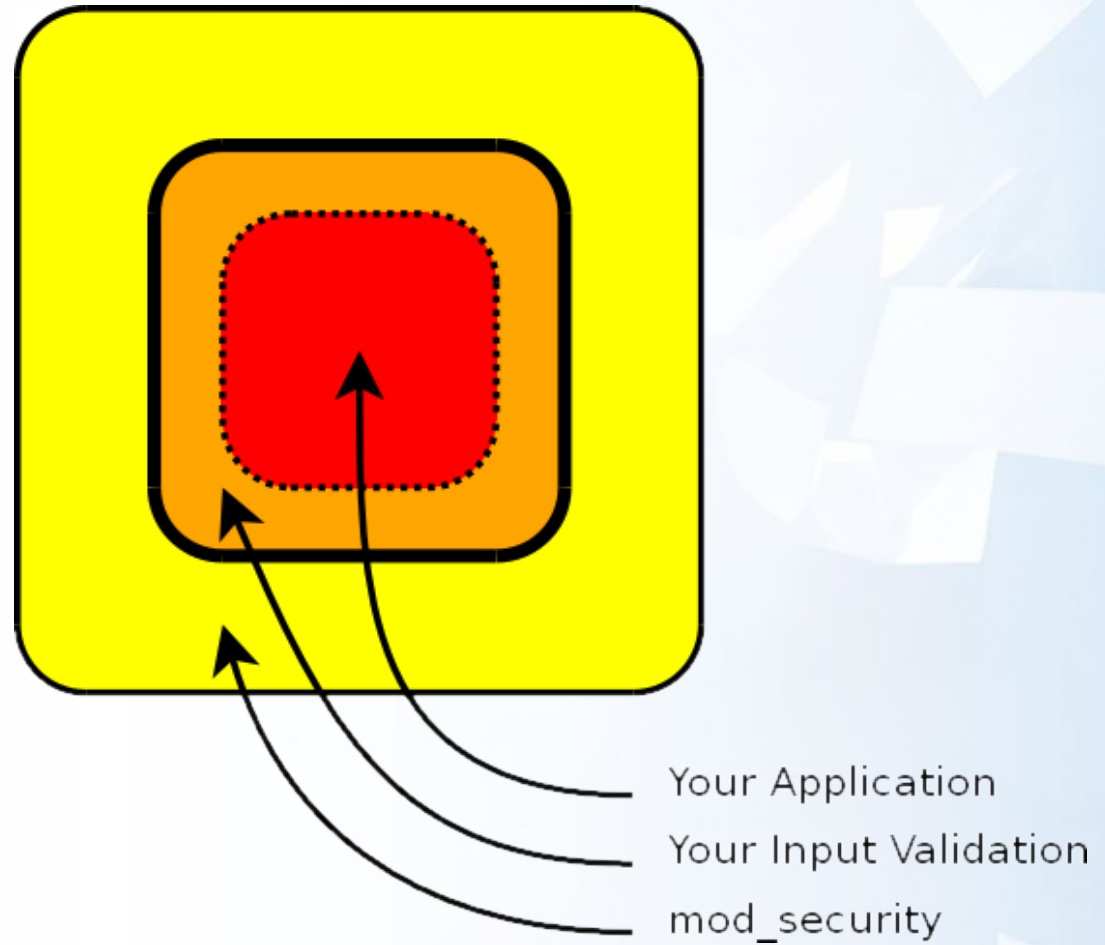
:)

[http://example.com/script.php?name=<script>alert\('!'\);</script>](http://example.com/script.php?name=<script>alert('!');</script>)

<script>alert('!');</script>

Multiple Levels of Defense

Share your information

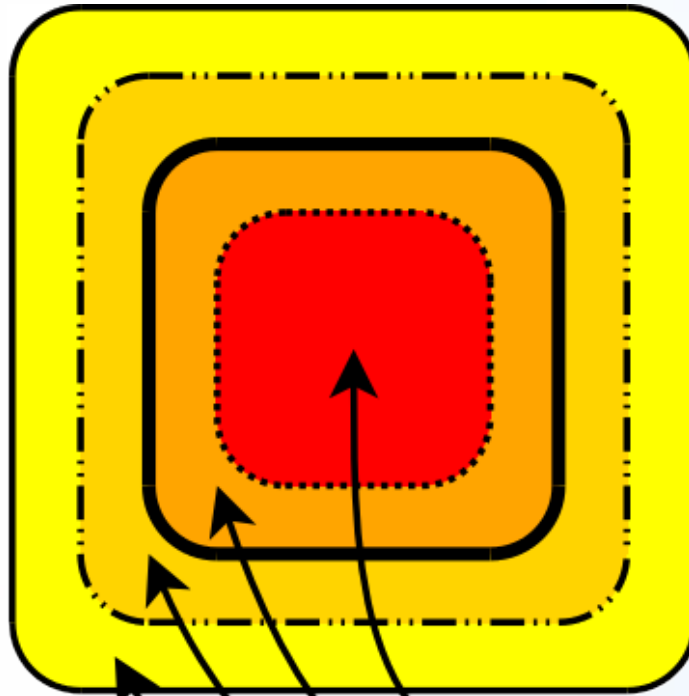


mod_security

- Apache module
- Can be used to avoid certain attack
- Is not PHP specific

Multiple Levels of Defense++

Share your information



Your Application

Your Input Validation

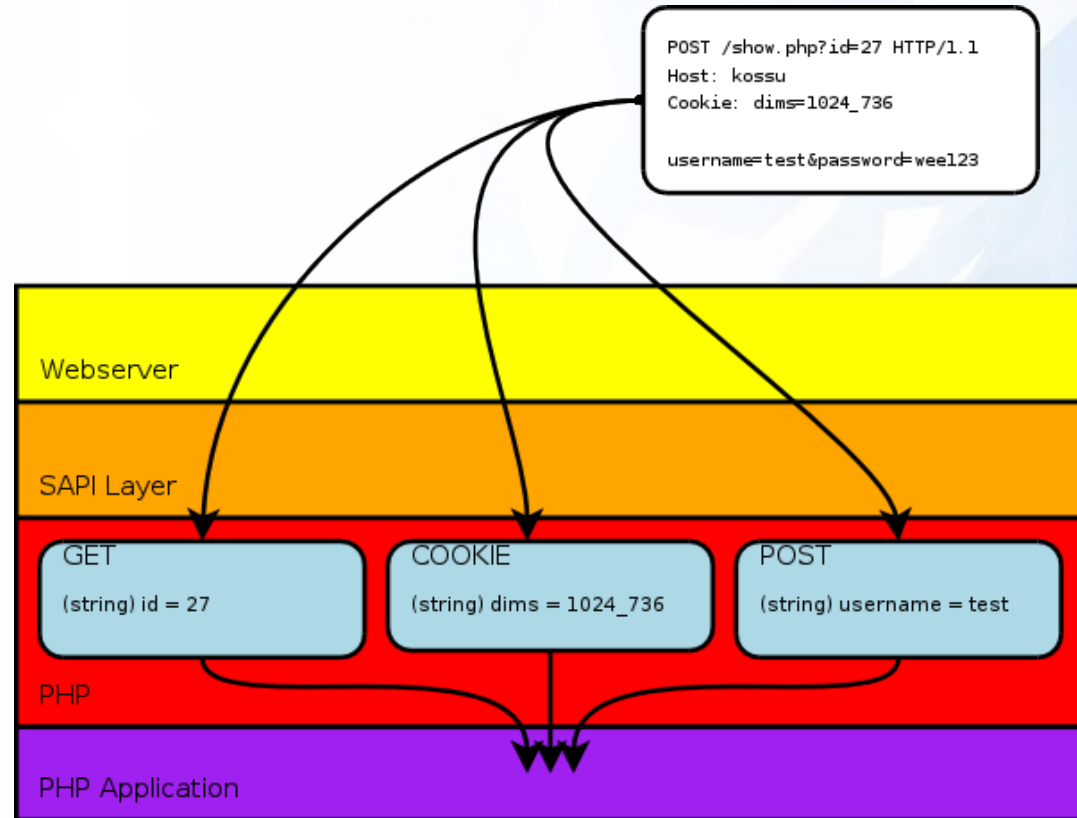
SAPI Input Filter

mod_security

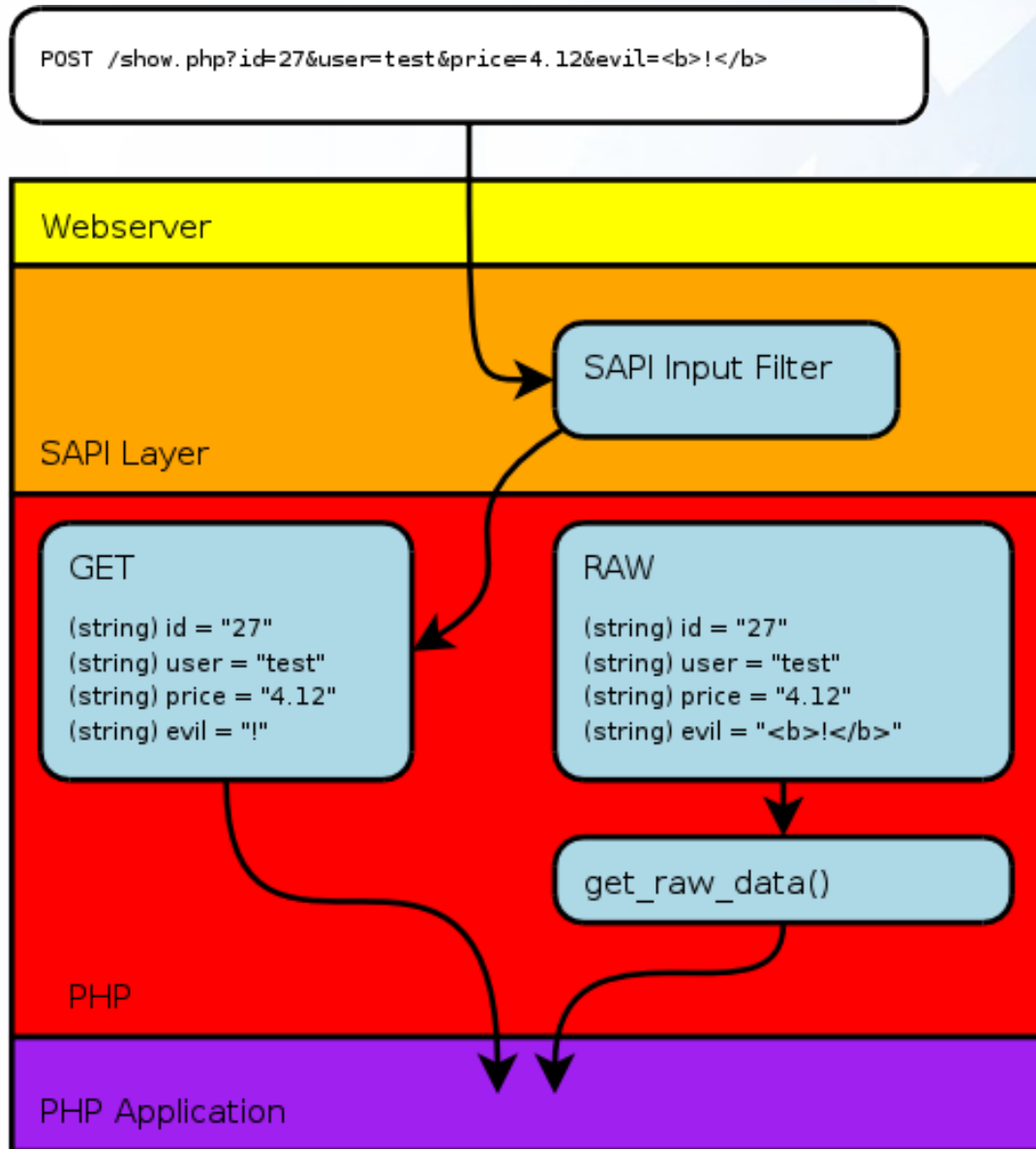
SAPI Input filter

- Sits between PHP and the webserver
- Is used while fetching data from users sources
- Can be used to filter data
- Prohibit data from entering PHP
- Written as a C extension to PHP
- Server wide filter

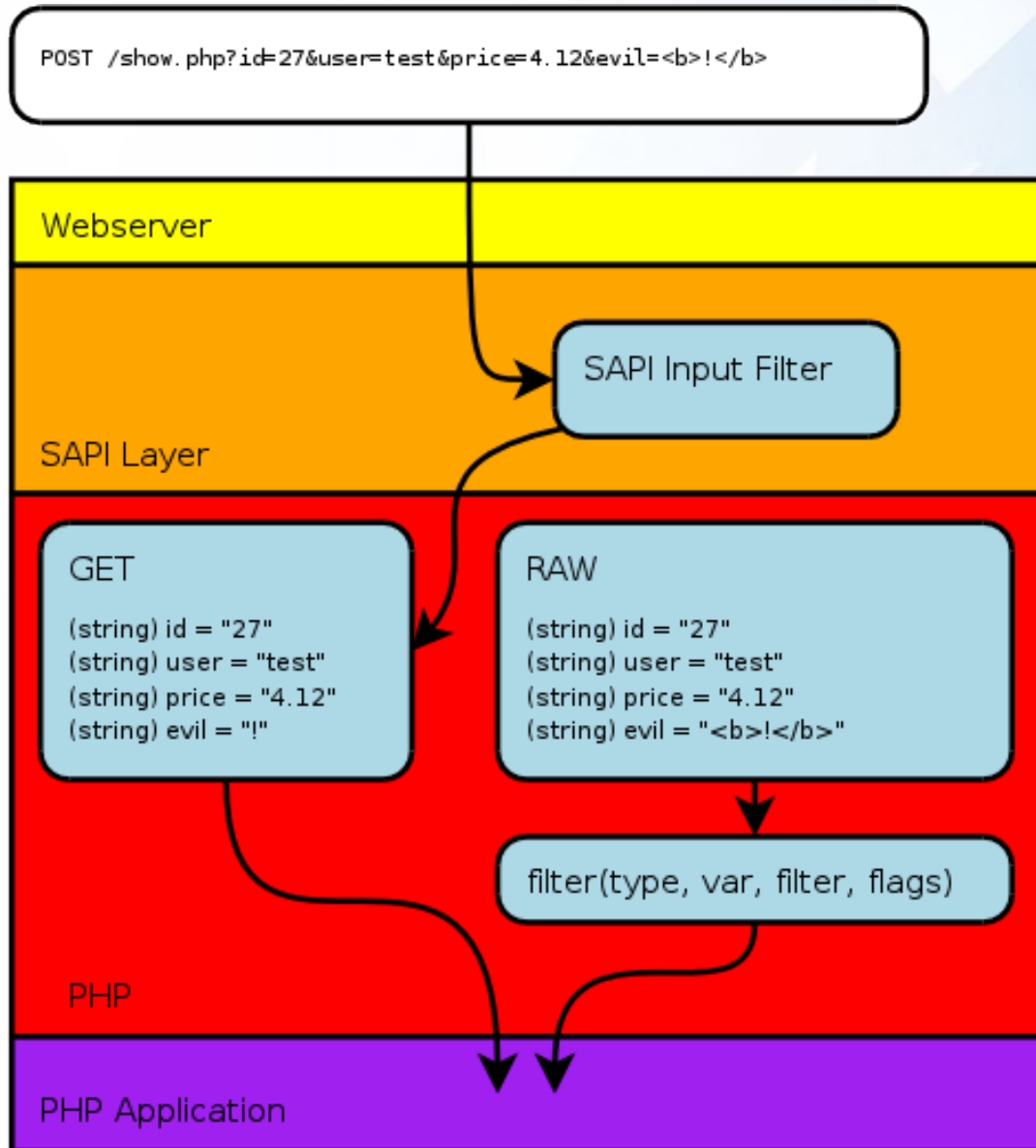
Current Situation



First Idea of an Input Filter



Second Idea of an Input Filter



- Comes with a filter extension
- Enabled by default
- Provides a default filter
- Provides two groups of accessing filters: sanitizing and logical
- Filters can have options to configure their behavior

Sanitizing filters try-out

```
<form action="" method="get">
<input type="text" name="data" maxlength="64" size="64"/>
<input type="submit"/>
</form>
<?php
if ( isset( $_GET['data'] ) ) {
    $filter = ini_get('filter.default');
    echo "The data filterered through '$filter' is:";
    var_dump( $_GET['data'] );
}
?>
```

```
<form action="" method="get">
Data: <input type="text" name="data" maxlength="64" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$options = FILTER_FLAG_STRIP_HIGH;

if (isset($_GET['data'])) {
    $data = input_get(
        INPUT_GET,           // source
        'data',             // parameter name
        FILTER_SANITIZE_STRING, // filter
        $options            // options
    );
    var_dump( $data );
}
?>
```

Getting Input Arrays

```
<form action="" method="get">
Data 1: <input type="text" name="data[]" maxlength="64" size="64"/><br/>
Data 2: <input type="text" name="data[]" maxlength="64" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$options = FILTER_FLAG_STRIP_HIGH | FILTER_FLAG_ARRAY;

if (isset($_GET['data'])) {
    $data = input_get(
        INPUT_GET,           // source
        'data',             // parameter name
        FILTER_SANITIZE_STRING, // filter
        $options            // options
    );
    var_dump( $data );
}
?>
```


Getting Input Arrays

```
<form action="" method="get">
Number: <input type="text" name="int" maxlength="64" size="64"/><br/>
String: <input type="text" name="string" maxlength="64" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$definition = array(
    'int' => array(
        'filter' => FILTER_VALIDATE_INT,
        'options' => array( "min_range" => 1, "max_range" => 10 )
    ),
    'string' => FILTER_SANITIZE_SPECIAL_CHARS
);

if (isset($_GET['int'])) {
    $data = input_get_args( INPUT_GET, $definition );
    var_dump( $data );
}
?>
```

Getting Input Arrays

```
<?php
$text = "This text is going to be changed\n";
$data = filter_data(
    $text,
    input_name_to_filter( 'special_chars' ),
    FILTER_FLAG_STRIP_HIGH
);
var_dump( $data );
?>
```

Sanitizing Filters

- Allows or disallows characters in a string
- Does not take care about data formats
- Does not transform types, but keeps strings

Sanitizing filters

Is basically "strip_tags", but supports a couple of flags:

```
<?php
$flags = array(
    'FILTER_FLAG_NO_ENCODE_QUOTES', 'FILTER_FLAG_ENCODE_LOW',
    'FILTER_FLAG_ENCODE_HIGH', 'FILTER_FLAG_STRIP_LOW',
    'FILTER_FLAG_STRIP_HIGH'
);
$filter = FILTER_SANITIZE_STRING;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Sanitizing filters

Is basically "url_encode", but supports a couple of flags:

```
<?php
$flags = array(
    'FILTER_FLAG_STRIP_LOW', 'FILTER_FLAG_STRIP_HIGH'
);
$filter = FILTER_SANITIZE_ENCODED;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Sanitizing filters

Is basically "html_special_chars", but supports a couple of flags:

```
<?php
$flags = array(
    'FILTER_FLAG_STRIP_LOW', 'FILTER_FLAG_STRIP_HIGH',
    'FILTER_FLAG_ENCODE_HIGH',
);
$filter = FILTER_SANITIZE_SPECIAL_CHARS;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Sanitizing filters

By default doesn't do anything, but supports a couple of flags:

```
<?php
$flags = array(
    'FILTER_FLAG_ENCODE_AMP', 'FILTER_FLAG_ENCODE_LOW',
    'FILTER_FLAG_ENCODE_HIGH', 'FILTER_FLAG_STRIP_LOW',
    'FILTER_FLAG_STRIP_HIGH'
);
$filter = FILTER_SANITIZE_STRING;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Sanitizing filters

This only strips out illegal characters, no validation is done!
The allowed characters are: a-z A-Z 0-9 " ! # \$ % & ' * + - / = ? ^
_ ` { | } ~ @ . []

```
<?php
$flags = array();
$filter = FILTER_SANITIZE_EMAIL;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```


Sanitizing filters

This only strips out illegal characters, no validation is done!
The allowed characters are: a-z A-Z 0-9 \$ - _ . + ! * ' () , { } | \ ^ ~
[] ` < > # % " ; / ? : @ & =

```
<?php
$flags = array();
$filter = FILTER_SANITIZE_URL;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Sanitizing filters

Strips all chars which are not 0-9, + and -.

```
<?php
$flags = array();
$filter = FILTER_SANITIZE_NUMBER_INT;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Sanitizing filters

Strips all chars which are not 0-9, + and -, but supports some flags as well.

```
<?php
$flags = array(
    'FILTER_FLAG_ALLOW_FRACTION', 'FILTER_FLAG_ALLOW_THOUSAND',
    'FILTER_FLAG_ALLOW_SCIENTIFIC'
);
$filter = FILTER_SANITIZE_NUMBER_FLOAT;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Sanitizing filters

Filter that applies magic quotes or actually "add_slashes"

```
<?php
$flags = array();
$filter = FILTER_SANITIZE_MAGIC_QUOTES;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Validating Filters

- Analyses the data in a logical way
- Understands data formats
- Can transform type

```
<a name='form' /><?php
$flags = array( 'FILTER_FLAG_ALLOW_OCTAL', 'FILTER_FLAG_ALLOW_HEX' );
$filter = FILTER_VALIDATE_INT;
?>
<form action="" method="get">
data: <input type="text" name="data" maxlength="64" size="64"/>
<table>
<?php foreach( $flags as $flagName ) {
echo "<tr><td>$flagName</td><td><input type='checkbox'
name='$flagName' /></td></tr>";
} ?>
</table>
Min: <input type="text" name="min" /><br/>Max: <input type="text" name="max" />
<input type="submit" />
</form>
<?php
if (!empty($_GET['min']))
    $options['min_range'] = (int) $_GET['min'];
if (!empty($_GET['max']))
    $options['max_range'] = (int) $_GET['max'];
if (isset($_GET['data'])) {
    $o = 0;
    foreach( $flags as $flagName )
        if ( isset( $_GET[$flagName] ) )
            $o |= constant( $flagName );
    $options['flags'] = $o;
    $data = input_get( INPUT_GET, 'data', $filter, $options );
    var_dump( $data );
}
?>
```

Logical filters

Returns true for '1', 'on', 'yes' and 'true', false otherwise.

```
<?php
$flags = array();
$filter = FILTER_VALIDATE_BOOLEAN;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

```
<?php
$flags = array();
$filter = FILTER_VALIDATE_FLOAT;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```



```
<form action="" method="get">
Data: <input type="text" name="data" maxlength="64" size="64"/><br/>
Regexp: <input type="text" name="regexp" maxlength="64" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$filter = FILTER_VALIDATE_REGEXP;
if (!empty($_GET['regexp']))
    $options['regexp'] = $_GET['regexp'];

if (isset($_GET['data'])) {
    $data = input_get( INPUT_GET, 'data', $filter, $options );
    var_dump( $data );
}
?>
```

```
<?php
$flags = array(
    'FILTER_FLAG_SCHEME_REQUIRED', 'FILTER_FLAG_HOST_REQUIRED',
    'FILTER_FLAG_PATH_REQUIRED', 'FILTER_FLAG_QUERY_REQUIRED'
);
$filter = FILTER_VALIDATE_URL;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

```
<?php
$flags = array();
$filter = FILTER_VALIDATE_EMAIL;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

```
<?php
$flags = array(
    'FILTER_FLAG_IPV4', 'FILTER_FLAG_IPV6', 'FILTER_FLAG_NO_RES_RANGE',
    'FILTER_FLAG_NO_PRIV_RANGE'
);
$filter = FILTER_VALIDATE_IP;
include 'presentations/slides/input-filter/render-form.php';
include 'presentations/slides/input-filter/check-data.php';
?>
```

Callback Filter

- Allows you to write your own callback filter
- Can not be used as default filter

```
<form action="" method="get">
Data: <input type="text" name="data" maxlength="64" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$filter = FILTER_CALLBACK;
$callback = array( 'Validate', 'My' );

if (isset($_GET['data'])) {
    $data = input_get( INPUT_GET, 'data', $filter, $callback );
    var_dump( $data );
}

class Validate {
    function My( $text ) {
        if ( $text == 'PHP' ) {
            $text = 'PHP Rocks!';
            return $text;
        }
        return false;
    }
}
?>
```



These slides: h
SQLite Input Filter: http://derickretnans.nl/sqlite_filter.php
Questions?: <mailto:dr@ez.no>