Curs TWSS

Validarea și manipularea erorilor în PHP și Javascript

Validarea erorilor

- Scop:
  - crearea si prelucrarea formularelor web într-un mod cât mai prietenos posibil
- Validarea input-ului utilizatorilor cu Javascript:
  - Datele validate client side trebuie revalidate server side

SQL Injection

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
```

$user: *fredsmith*
$password: *mypass*

SELECT * FROM users WHERE user='fredsmith' AND pass='mypass'

$user: *admin' #*

SELECT * FROM users WHERE user='admin' #' AND pass=''

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "DELETE FROM users WHERE user='$user' AND pass='$pass'";
```

$user: *anything' OR 1=1 #*

DELETE FROM users WHERE user='anything' OR 1=1 #' AND pass=''

Sanitizarea input-ului primit de la utilizatori

```
<?php
function mysql_fix_string($string)
{
 if (get_magic_quotes_gpc())
   $string = stripslashes($string);
 return mysql_real_escape_string($string);
}
?>
```

```
<?php
$user = mysql_fix_string($_POST['user']);
$pass = mysql_fix_string($_POST['pass']);
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
?>
```

Utilizarea substituenților

```
PREPARE statement FROM "INSERT INTO classics VALUES(?,?,?,?,?)";
SET @author = "Emily Brontë",
@title = "Wuthering Heights",
@category = "Classic Fiction",
@year = "1847",
@isbn = "9780553212587";
EXECUTE statement USING @author,@title,@category,@year,@isbn;
DEALLOCATE PREPARE statement;
```

Utilizarea substituenților

```
<?php
require 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server)
  die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database) or die("Unable to select database: " .
mysql_error());

$query = 'PREPARE statement FROM "INSERT INTO classics VALUES(?,?,?,?,?)"';
mysql_query($query);

$query = 'SET @author = "Emily Brontë",' .
'@title = "Wuthering Heights",' .
'@category = "Classic Fiction",' .
'@year = "1847",' .
'@isbn = "9780553212587"';
mysql_query($query);

$query = 'EXECUTE statement USING @author,@title,@category,@year,@isbn';
mysql_query($query);

$query = 'DEALLOCATE PREPARE statement';
mysql_query($query);
?>
```

PHP - Input-uri

- Sanitizarea input-urilor:
  - mysql_real_escape_string()
  - stripslashes()
  - htmlentities()
  - strip_tags()

```
<?php
function sanitizeString($var)
{
  $var = stripslashes($var);
  $var = htmlentities($var);
  $var = strip_tags($var);
  return $var;
}
function sanitizeMySQL($var)
{
  $var = mysql_real_escape_string($var);
  $var = sanitizeString($var);
  return $var;
}
?>
```

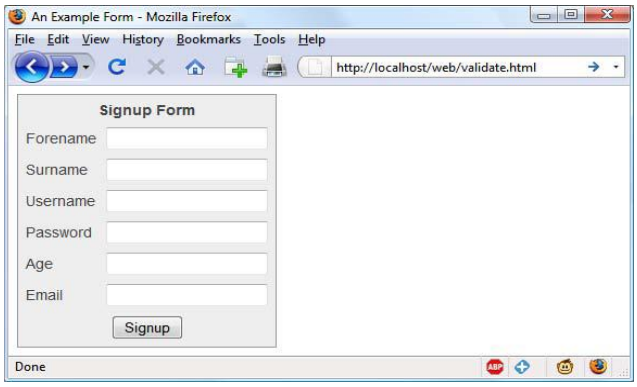
XSS - Cross Site Scripting

Apelul funcției htmlentities()
<script src='http://x.com/hack.js'> </script><script>hack();</script>
<script src='http://x.com/hack.js'> </script><script>hack();</script>

PHP - Expresii regulare

/g – global
/i – case insensitive
/m – multi line string
/cats/g - match both occurrences of the word cats in the sentence *I like cats and cats like me*.
/dogs/gi - match both occurrences of the word dogs (Dogs and dogs) in the sentence *Dogs like other dogs*.

PHP – Validarea erorilor



Validarea unui formular de înregistrare
(prenume, nume, user, parola, vârstă, email)

```
<form method="post" action="adduser.php" onSubmit="return validate(this)">
Forename <input type="text" maxlength="32" name="forename" />
Surname  <input type="text" maxlength="32" name="surname" />
Username <input type="text" maxlength="16" name="username" />
Password <input type="text" maxlength="12" name="password" />
Age      <input type="text" maxlength="3" name="age" />
Email    <input type="text" maxlength="64" name="email" />
        <input type="submit" value="Signup">
</form>
```

```
<script>
function validate(form) {
  fail = validateForename(form.forename.value)
  fail += validateSurname(form.surname.value)
  fail += validateUsername(form.username.value)
  fail += validatePassword(form.password.value)
  fail += validateAge(form.age.value)
  fail += validateEmail(form.email.value)
  if (fail == "")
    return true
  else
    { alert(fail); return false }
}
</script>
```

```
<script>
function validateForename(field) {
  if (field == "")
    return "No Forename was entered.\n"
  return ""
}
</script>
```

```
<script>
function validateSurname(field) {
  if (field == "")
    return "No Surname was entered.\n"
  return "" }
</script>
```

```
<script>
function validateUsername(field) {
  if (field == "")
    return "No Username was entered.\n"
  else if (field.length < 5)
    return "Usernames must be at least 5 characters.\n"
  else if (/[^a-zA-Z0-9_-]/.test(field))
    return "Only a-z, A-Z, 0-9, - and _ allowed in Usernames.\n"
  return ""
}
</script>
```

```
<script>
function validatePassword(field) {
  if (field == "")
    return "No Password was entered.\n"
  else if (field.length < 6)
    return "Passwords must be at least 6 characters.\n"
  else if (!/[a-z]/.test(field) || ! /[A-Z]/.test(field) ||
!/[0-9]/.test(field))
    return "Passwords require one each of a-z, A-Z and 0-9.\n"
  return ""
}
</script>
```

```
<script>
function validateAge(field) {
  if (isNaN(field))
    return "No Age was entered.\n"
  else if (field < 18 || field > 110)
    return "Age must be between 18 and 110.\n“
  return ""
}
</script>
```

```
<script>
function validateEmail(field) {
  if (field == "")
    return "No Email was entered.\n"
  else if (!((field.indexOf(".") > 0) && (field.indexOf("@") > 0)) || /[^a-zA-Z0-9.@_-
]/.test(field))
    return "The Email address is invalid.\n"
  return ""
}
</script>
```

/g – global
/i – case insensitive
/m – multi line string
/cats/g - match both occurrences of the word cats in the sentence *I like cats and cats like me*.
/dogs/gi - match both occurrences of the word dogs (Dogs and dogs) in the sentence *Dogs like other dogs*.

- Fiecare expresie regulară trebuie încapsulată între caracterele "/"
- Caracterele încapsulate în "/" sunt caractere speciale numite metacaractere
-

Căutarea expresiei *Le Guin* într-un text
/Le *Guin/ - 0 sau mai multe spații
/Le +Guin/ - cel puțin un spațiu
The difficulty of classifying Le Guin's works

. – poate potrivi orice caracter (în afară de enter)
/<.*>/ - caută tag-uri HTML
<>, ,
/<.+>/ - caută tag-uri HTML
, , , , , <a href="test.ro">, Introduction
/5\.0*/
5.0, 5.00, 5., 5.000
/1(,000)+ /
1,000 1,000,000 1,000,000,000 1,000,000,000,000 ...
/gr[ae]y/
gray, grey
/[0-9]/, /\d/
Potrivește o singură cifră
/Yahoo[^!]/ - negare
Yahoo?
/<[^>]+>/

- **/** - Begins and ends the regular expression
- **.** - Matches any single character except the newline
- ***element**** - Matches *element* zero or more times
- ***element+*** - Matches *element* one or more times
- ***element?*** - Matches *element* zero or one time
- **[*characters*]** - Matches a character out of those contained within the brackets
- **[^*characters*]** - Matches a single character that is not contained within the brackets
- **(*regex*)** - Treats the *regex* as a group for counting or a following *, +, or ?
- ***left*|*right*** - Matches either *left* or *right*

- ***l-r*** - Matches a range of characters between *l* and *r* (only within brackets)
- **^** - Requires match to be at the string's start
- **$** - Requires match to be at the string's end
- **\b** - Matches a word boundary
- **\B** - Matches where there is not a word boundary
- **\d** - Matches a single digit
- **\D** - Matches a single nondigit
- **\n** - Matches a newline character
- **\s** - Matches a whitespace character
- **\S** - Matches a nonwhitespace character
- **\t** - Matches a tab character
- **\w** - Matches a word character (a-z, A-Z, 0-9, and _)
- **\W** - Matches a nonword character (anything but a-z, A-Z, 0-9, and _)
- ***x*** - *x* (useful if *x* is a metacharacter, but you really want *x*)
- ***{n}*** - Matches exactly *n* times
- ***{n,}*** - Matches *n* times or more
- ***{min,max}*** - Matches at least *min* and at most *max* times
- **r** - The first r in *The quick brown*
- **rec[ei][ei]ve** - Either of *receive* or *recieve* (but also *receeve* or *reciive*)
- **rec[ei]{2}ve** - Either of *receive* or *recieve* (but also *receeve* or *reciive*)
- **rec(ei)|(ie)ve** - Either of *receive* or *recieve* (but not *receeve* or *reciive*)
- **cat** - The word cat in *I like cats and dogs*
- **cat|dog** - Either of the words cat or dog in *I like cats and dogs*
- **\.** - . (the \ is necessary because . is a metacharacter)
- **a-f** - Any of the characters *a, b, c, d, e* or *f*
- **cats$** - Only the final cats in *My cats are friendly cats*
- **^my** - Only the first my in *my cats are my pets*
- **\d{2,3}** - Any two or three digit number (*00* through *999*)
- **[\w]+** - Any word of one or more characters
- **[\w]{5}** - Any five-letter word

Javascript
document.write(/cats/i.test("Cats are fun. I like cats."))
document.write("Cats are fun. I like cats.".replace(/cats/gi,"dogs"))

PHP
$n = preg_match("/cats/i", "Cats are fun. I like cats.");
$n = preg_match("/cats/i", "Cats are fun. I like cats.", $match); echo "$n Matches: $match[0]"; // 1 Matches: Cats
$n = preg_match_all("/cats/i", "Cats are fun. I like cats.", $match); echo "$n Matches: "; for ($j=0 ; $j < $n ; ++$j) echo $match[0][$j]." ";
echo preg_replace("/cats/i", "dogs", "Cats are fun. I like cats.");

Validarea unui formular de înregistrare (prenume, nume, user, parola, vârstă, email)

```
<form method="post" action="adduser.php" onSubmit="return validate(this)">
Forename <input type="text" maxlength="32" name="forename" />
Surname  <input type="text" maxlength="32" name="surname" />
Username <input type="text" maxlength="16" name="username" />
Password <input type="text" maxlength="12" name="password" />
Age      <input type="text" maxlength="3" name="age" />
Email    <input type="text" maxlength="64" name="email" />
        <input type="submit" value="Signup">
</form>
```

```
<?php
$forename = $surname = $username = $password = $age = $email = "";
if (isset($_POST['forename']))
  $forename = fix_string($_POST['forename']);
if (isset($_POST['surname']))
  $surname = fix_string($_POST['surname']);
if (isset($_POST['username']))
  $username = fix_string($_POST['username']);
if (isset($_POST['password']))
  $password = fix_string($_POST['password']);
if (isset($_POST['age']))
  $age = fix_string($_POST['age']);
if (isset($_POST['email']))
  $email = fix_string($_POST['email']);
$fail = validate_forename($forename);
$fail .= validate_surname($surname);
$fail .= validate_username($username);
$fail .= validate_password($password);
$fail .= validate_age($age);
$fail .= validate_email($email);
if ($fail == "") {
  echo "Form data successfully validated";
  exit;
}
?>
```

```
<?php
function validate_forename($field) {
  if ($field == "")
    return "No Forename was entered<br />";
  return "";
}
?>
```

```
<?php
function validate_surname($field) {
  if ($field == "")
    return "No Surname was entered<br />";
  return ""; }
?>
```

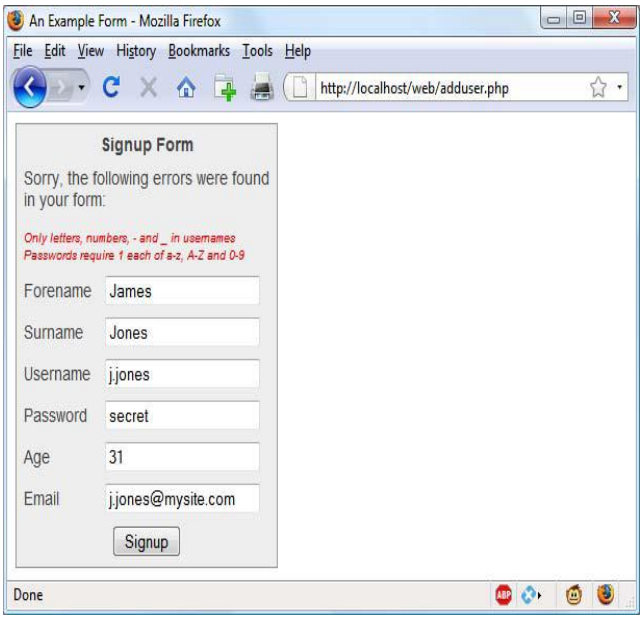
```php
<?php
function validate_username($field) {
  if ($field == "")
    return "No Username was entered<br />";
  else if (strlen($field) < 5)
    return "Usernames must be at least 5 characters<br/>";
  else if (preg_match("/[^a-zA-Z0-9_-]/", $field))
    return "Only letters, numbers, - and _ in usernames";
  return "";
}
?>
```

```php
<?php
function validate_password($field) {
  if ($field == "")
    return "No Password was entered<br />";
  else if (strlen($field) < 6)
    return "Passwords must be at least 6 characters<br/>";
  else if ( !preg_match("/[a-z]/", $field) ||
!preg_match("/[A-Z]/", $field) ||
!preg_match("/[0-9]/", $field))
    return "Passwords require 1 each of a-z, A-Z and 0-9";
  return "";
}
?>
```

```php
<?php
function validate_age($field) {
  if ($field == "")
    return "No Age was entered<br />";
  else if ($field < 18 || $field > 110)
    return "Age must be between 18 and 110<br />";
  return "";
}
?>
```

```php
<?php
function validate_email($field) {
  if ($field == "")
    return "No Email was entered<br />";
  else if (!((strpos($field, ".") > 0) &&
(strpos($field, "@") > 0)) ||
preg_match("/[^a-zA-Z0-9.@_-]/", $field))
    return "The Email address is invalid<br />";
  return "";
}
?>
```

```php
<?php
function fix_string($string) {
  if (get_magic_quotes_gpc())
    $string = stripslashes($string);
  return htmlentities ($string);
}
?>
```


An Example Form - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/web/adduser.php
Signup Form
Sorry, the following errors were found in your form:
Only letters, numbers, - and _ in usernames
Passwords require 1 each of a-z, A-Z and 0-9
Forename James
Surname Jones
Username j.jones
Password secret
Age 31
Email j.jones@mysite.com
Signup
Done

Trimitere mail-uri în PHP

Nume	Valoare implicită	Descriere
SMTP	Localhost	Nume sau adresa IP a serverului SMTP
SMTP_PORT	25	Port SMTP
sendmail_from	NULL	Specifică adresa expeditorului e-mail-ului (Windows)
sendmail_path	NULL	Specifică locația programului sendmail (Linux)

mail(to,subject,message,headers,parameters)	
to	Destinatar (*)
subject	Subiectul mail-ului (*)
message	Textul e-mail-ului (*)
headers	Specifică headere adiționale (From, Cc, Bcc)
parameters	Specifică parametrii adiționali

```
<?php
$to = "someone@example.com";
$subject = "Test mail";
$message = "Hello! This is a simple email message.";
$from = "someonelse@example.com";
$headers = "From: $from";
mail($to,$subject,$message,$headers);
echo "Mail Sent."; ?>
```

PHP - Securitatea trimiterii e-mail-urilor

```
<html>
<body>
<?php
if (isset($_REQUEST['email']))
{
 $email = $_REQUEST['email'] ;
 $subject = $_REQUEST['subject'] ;
 $message = $_REQUEST['message'] ;
 mail("s@example.com", "Subject: $subject", $message, "From: $email" );
 echo "Thank you for using our mail form"; }
else
{
 echo "<form method='post' action='mailform.php'>
 Email: <input name='email' type='text' /><br />
 Subject: <input name='subject' type='text' /><br />
 Message:<br />
 <textarea name='message' rows='15' cols='40'>
 </textarea><br />
 <input type='submit' />
 </form>"; }
?>
</body>
</html>
```

Tratarea erorilor în PHP

ERORI:

- **Notices** – erori triviale, non-critice; în mod implicit, aceste erori nu sunt afișate utilizatorului
- **Warnings** – erori mai complexe; în mod implicit sunt afișate utilizatorului, dar nu opresc executarea script-ului
- **Fatal errors** – erori critice; în mod implicit sunt afișate utilizatorului și cauzează terminarea imediată a script-ului

Exemplu
<?php $string = 'a string'; explode($string); ?>
Warning: Wrong parameter count for explode() in C:\wamp\www\test2\part12\eg1.php on line 8
E_WARNING – non-fatal error

Exemplu
<?php callMeJoe(); ?>
Fatal error: Call to undefined function callMeJoe() in C:\wamp\www\test2\part12\eg2.php on line 5
E_ERROR – fatal error

Exemplu

<?php
// report only fatal errors
error_reporting(E_ERROR);
$string = 'string';
explode($string);
?>

No output

Exemplu

<?php
// report no fatal errors
error_reporting(~E_ERROR);
callMeJoe();
?>

No output

Folosirea funcției die()

```
<?php
$file=fopen("welcome.txt","r");
?>
```

```
Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2
```

```
<?php
if(!file_exists("welcome.txt"))
{
die("File not found");
}
else
{
$file=fopen("welcome.txt","r");
}
?>
```

```
File not found
```

Folosirea unor funcții personalizate de tratare a erorilor

```
error_function(error_level,error_message,error_file,error_line,error_context)
```

- **error_level** – specifică tipul de eroare (*):
  - 2 (E_WARNING)
  - 8 (E_NOTICE)
  - 256 (E_USER_ERROR) = E_ERROR setat în urma apelului funcției trigger_error()
  - 512 (E_USER_WARNING) = E_WARNING setat în urma apelului funcției trigger_error()
  - 1024 (E_USER_NOTICE) = E_NOTICE setat în urma apelului funcției trigger_error()
  - 4096 (E_RECOVERABLE_ERROR) = E_ERROR setat în urma apelului funcției set_error_handler()
  - 8191 (E_ALL)
- **error_message** – specifică mesajul de eroare afișat utilizatorului (*)
- **error_file** – specifică numele fișierului în care a apărut eroarea
- **error_line** – specifică numărul liniei în care a apărut eroarea
- **error_context** – specifică un șir în care sunt descrise valorile variabilelor programului în momentul apariției erorii

```
function customError($errno, $errstr)
{
  echo "<b>Error:</b> [$errno] $errstr<br />";
  echo "Ending Script";
  die();
}
```

```
set_error_handler("customError");
```

```
echo($test);
```

```
Custom error: [8] Undefined variable: test
```

```
<?php
$test=2;
if ($test>1)
{
 trigger_error("Value must be 1 or below");
}
?>
```

```
Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6
```

```
<?php
function customError($errno, $errstr)
{
 echo "<b>Error:</b> [$errno] $errstr<br />";
 echo "Ending Script";
 die();
}
set_error_handler("customError",E_USER_WARNING);
$test=2;
if ($test>1)
{
 trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

```
Error: [512] Value must be 1 or below
Ending Script
```

PHP – logarea erorilor

```
<?php
function customError($errno, $errstr)
{
 echo "<b>Error:</b> [$errno] $errstr<br />";
 echo "Webmaster has been notified";
 error_log("Error: [$errno] $errstr",1,
"someone@example.com","From: webmaster@example.com");
}
set_error_handler("customError",E_USER_WARNING);
$test=2;
if ($test>1)
{
 trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

```
Error: [512] Value must be 1 or below
Webmaster has been notified
```

Tratarea excepțiilor în PHP

```
<?php
function checkNum($number)
{
 if($number>1)
 {
 throw new Exception("Value must be 1 or below");
 }
 return true;
}
checkNum(2);
?>
```

```
Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6
```

- Try: o funcție care tratează o excepție va fi într-un block TRY. Dacă excepția nu a fost declanșată, execuția codului va continua în mod normal
- Throw: declanșează excepția; fiecărui THROW îi corespunde cel puțin un CATCH
- Catch: creează un obiect care conține informația despre excepție

```
<?php
function checkNum($number)
{
 if($number>1)
   throw new Exception("Value must be 1 or below");
 return true;
}
try
{
 checkNum(2);
 echo 'If you see this, the number is 1 or below';
}
catch(Exception $e)
 echo 'Message: ' .$e->getMessage();
?>
```

```
Message: Value must be 1 or below
```

Crearea unei excepții personalizate

```
<?php
class customException extends Exception
{
  public function errorMessage()
  {
  $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile().': <b>'.$this-
>getMessage().'</b> is not a valid E-Mail address';
  return $errorMsg;
  }
}
$email = "someone@example...com";
try
{
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
    throw new customException($email);
}
catch (customException $e)
  echo $e->errorMessage();
?>
```

Excepții multiple

```
<?php
class customException extends Exception
{
  public function errorMessage()
  {
  $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile().': <b>'.$this-
>getMessage().'</b> is not a valid E-Mail address';
  return $errorMsg;
  }
}
$email = "someone@example.com";
try
{
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
    throw new customException($email);
  if(strpos($email, "example") !== FALSE)
    throw new Exception("$email is an example e-mail");
}
catch (customException $e)
  echo $e->errorMessage();
catch(Exception $e)
  echo $e->getMessage();
}
?>
```

Filtre în PHP

- Filtrele PHP sunt utilizate pentru validarea și filtrarea datelor
- Testarea, validarea și filtrarea input-ului utilizatorului este o parte importantă a fiecărei aplicații web
- Tipuri de date care ar trebui filtrate:
    - Datele introduse în formulare
    - Cookie-uri
    - Variabilele server-ului
    - Rezultatele aduse în urma interogării bazei de date

Funcții și filtre	
filter_var	Filtrează o singură variabilă folosind un filtru specificat
filter_var_array ()	Filtrează mai multe variabile folosind unul sau mai multe filtre
filter_input ()	Preia valoarea unui input și o filtrează
filter_input_array ()	Preia valoarile din mai multe input-uri și le filtrează folosind unul sau mai multe filtre
filter_has_var ()	Verifică dacă există o variabilă de un anumit tip
filter_id ()	Returnează id-ul asociat filtrului specificat

Funcții și filtre	
FILTER_CALLBACK	Apelează funcția de filtrare definită de utilizator
FILTER_SANITIZE_STRING	Elimina tag-urile și opțional codifică sau elimină caracterele speciale
FILTER_SANITIZE_STRIPPED	= FILTER_SANITIZE_STRING
FILTER_SANITIZE_ENCODED	Codifică șir-urile de caractere ca URL și opțional codifică sau elimină caracterele speciale
FILTER_SANITIZE_SPECIAL_CHARS	Anulează efectul caracterelor ‘”<>& și a caracterelor ASCII cu codul mai mic de 32
FILTER_SANITIZE_EMAIL	Șterge toate caracterele în afară de litere, cifre și !#$%&'*+-/=?^_`{|}~@.[]
FILTER_SANITIZE_URL	Șterge toate caracterele în afară de litere, cifre și $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=
FILTER_SANITIZE_NUMBER_INT	Șterge toate caracterele în afară de cifre și +-
FILTER_SANITIZE_NUMBER_FLOAT	Șterge toate caracterele în afară de cifre, +- și opțional .,eE
FILTER_SANITIZE_MAGIC_QUOTES	Apelează funcția addslashes()

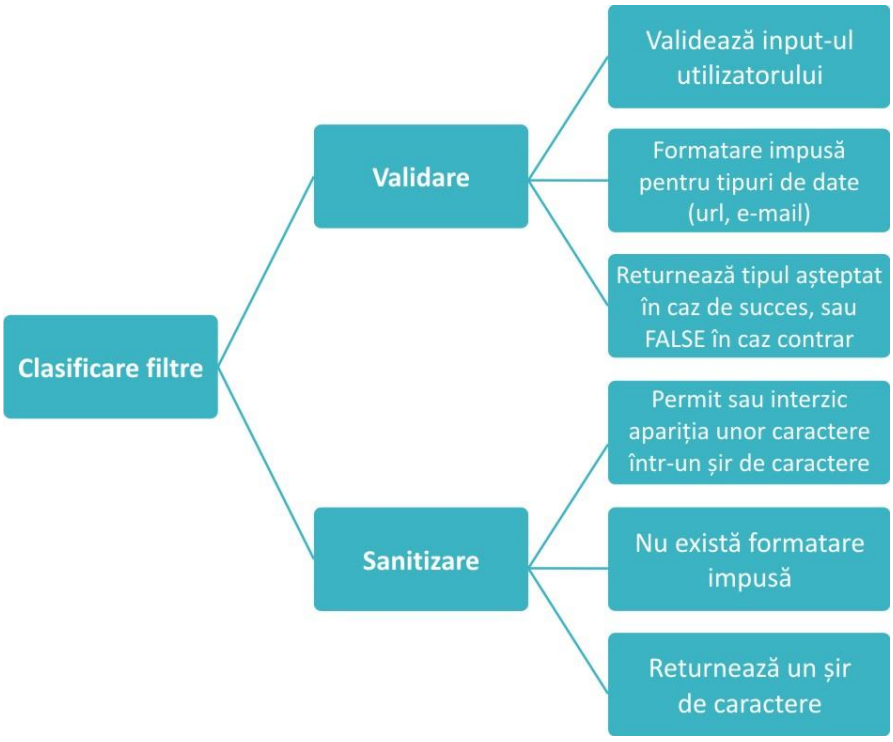
Funcții și filtre	
FILTER_UNSAFE_RAW	Opțional, codifică sau elimină caracterele speciale
FILTER_VALIDATE_INT	Validează valoarea ca întreg, opțional dintr-un interval specificat
FILTER_VALIDATE_BOOLEAN	Returnează TRUE pentru “1”, “true”, “on”, “yes”, FALSE pentru “0”, “false”, “off”, “no”, “” și NULL în celelalte cazuri
FILTER_VALIDATE_FLOAT	Validează valoarea ca număr real

FILTER_VALIDATE_REGEXP	Validează expresia regulară
FILTER_VALIDATE_URL	Validează URL, opțional se pot solicita anumite componente
FILTER_VALIDATE_EMAIL	Validează e-mail
FILTER_VALIDATE_IP	Validează adresa IP, opțional doar IPv4, IPv6 sau reale

Filtrarea unui întreg

```
<?php
$int = 123;
if(!filter_var($int, FILTER_VALIDATE_INT))
{
echo("Integer is not valid");
}
else
{
echo("Integer is valid");
}
?>
```

$int = "123abc"



Opțiuni și flag-uri pentru filtre

```
<?php
$var=300;
$int_options = array(
 "options"=>array
 (
  "min_range"=>0,
  "max_range"=>256
 )
);
if(!filter_var($var, FILTER_VALIDATE_INT, $int_options))
 echo("Integer is not valid");
else
 echo("Integer is valid");
?>
```

Sanitizare input

```
<?php
if(!filter_has_var(INPUT_POST, "url"))
 echo("Input type does not exist");
else
 {
 $url = filter_input(INPUT_POST,
 "url", FILTER_SANITIZE_URL);
 }
?>
```

"http://www.W3ååSchøøools.com/"	http://www.W3Schools.com/

Filtrare multiplă

```
<?php
$filters = array
(
 "name" => array
  (
  "filter"=>FILTER_SANITIZE_STRING
  ),
 "age" => array
  (
  "filter"=>FILTER_VALIDATE_INT,
  "options"=>array
   (
   "min_range"=>1,
   "max_range"=>120
   )
  ),
 "email"=> FILTER_VALIDATE_EMAIL,
);
$result = filter_input_array(INPUT_GET, $filters);
if (!$result["age"])
 echo("Age must be a number between 1 and 120.<br />");
elseif(!$result["email"])
```

```
  echo("E-Mail is not valid.<br />");
else
  echo("User input is valid");
?>
```

Filtre personalizate

```
<?php
function convertSpace($string)
{
  return str_replace("_", " ", $string);
}
$string = "Peter_is_a_great_guy!";
echo filter_var($string,FILTER_CALLBACK,array("options"=>"convertSpace"));
?>
```

```
Peter is a great guy!
```

Cuprins