college CD

Server-side technologies (TCS)

Class 7

The superglobals

$_SERVER

This superglobal is a special one as it actually is an array containing different predefined indexes which can be used to retrieve informations about the server.

Syntax :

$_SERVER["PREDEFINED_INDEX"]

Indexes (selection) :

PHP_SELF
Returns the name of the script currently being executed.

SERVER_NAME
Returns the name of the server on which the script is being executed.

SERVER_SOFTWARE
Returns the server's ID string.

SERVER_PROTOCOL
Returns the name and the revision of the information protocol.

REQUEST_METHOD
Returns the request method used to access a page *(GET, HEAD, POST, PUT)*.

HTTP_REFERER
Returns the URL of the page from which the current page was called.

HTTP_USER_AGENT
Returns the User_Agent of the current request, if available.

REMOTE_ADDR
Returns the user's IP address.

REMOTE_PORT
Returns the user's computer port used to make a request to the web server

SCRIPT_NAME
Returns the path of the current script.

Handling forms

Writing in a file using a form with « method="post" »

We already have seen how we can write in a file using different PHP functions. But you may want to have the users be able to change the contents of files. In this circumstances, you could use a simple form to do so.

index.php

```
<DOCTYPE html>
<html lang="fr">
<head>
<title>Writing into a file using forms</title>
</head>
<body>

<form action="greeting.php" method="post">
First name: <input type="text" name="first_name"><br>
Last name: <input type="text" name="last_name"><br>
<button type="submit">Submit</button>
</form>

</body>
</html>
```

Explanations :
When the user fills up the form's fields and presses the submit button, the data is sent for processing to another PHP file (greeting.php)

greeting.php

```
<html>
<body>

Welcome
<?php echo $_POST["first_name"] . " " . $_POST["last_name"]; ?>
// Result : Welcome John Smith

</body>
</html>
```

Explanations :
Since the result wanted is to show a greeting to screen, echo is used to show the content of the form's ***$_POST*** predefined variables (***superglobals***) that was posted and identified using the ***name=***" " attribute.

Writing in a file using a form with « method="get" »

The same result can be achieved using method="get", simply changing « post » for « get » everywhere they show. In both cases, these predefined variables create an array where the name attributes are associated to values.

index.php

```
<DOCTYPE html>
<html lang="fr">
<head>
<title>Writing into a file using forms</title>
</head>
<body>

<form action="greeting.php" method="get">
First name: <input type="text" name="first_name"><br>
Last name: <input type="text" name="last_name"><br>
<button type="submit">Submit</button>
</form>

</body>
</html>
```

greeting.php

```
<html>
<body>

Welcome
<?php echo $_GET["first_name"] . " " . $_GET["last_name"]; ?>

// Result : Welcome John Smith

</body>
</html>
```

The difference between « post » and « get »

$_POST's variables are passed to the script using an *http request* and, therefore, are invisible (embedded in the HTTP request). This method is well adapted to sensitive data, it allows an unlimited amount of data to be sent and supports many advanced options unavailable with the *get* method.

$_GET's variables are passed to the script using the URL parameters. The information is actually added to the URL and, therefore, are visible to everyone. This method limited to transfers of approximately 2000 characters. This method can be used for non-sentitive data, and the fact that the URL can be bookmarked may sometimes be useful.

Warning : Although this is working, never use forms without proper validation in order to protect yourself from spammers and hackers.

Basic form validation

```
<body>
<?php
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h1>My form</h1>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
        <label>Name:</label>
        <input type="text" name="name" required>
        <label>E-mail:</label>
        <input type="email" name="email" required >
        <label>Website:</label>
        <input type="url" name="website">
        <label>Comment:</label>
        <textarea name="comment" rows="5" cols="40"></textarea>
        <label>Gender:</label>
        <input type="radio" name="gender" value="Female">Female
        <input type="radio" name="gender" value="Male">Male
        <input type="radio" name="gender" value="Other">Other
        <button type="submit" name="submit">Submit!</button>
</form>

<?php
echo "<h2>User input:</h2>";
echo $name . "<br>";
echo $email . "<br>";
echo $website . "<br>";
echo $comment . "<br>";
echo $gender . "<br>";
?>
</body>
```

Variables declaration
Set to empty values.

Checks if the form has been submitted
If true, it needs to be validated. If false, displays a blank form.

Cleans the strings
Removes spaces, backslashes...

htmlspecialchars function
Avoid attacks conducted by adding HTML or JavaScript codes in the form.

Predefined variable (superglobal)
Returns the filename of the script being currently executed, sending collected data to itself.

To output in external file :
Replace $_SERVER["PHP_SELF"] by *yourfile.php*.

To erase here and to be replaced by the following in external file for external output (*yourfile.php*):

```
<?php
echo "<h2>User input:</h2>";
echo $_POST["name"] . "<br />";
echo $_POST["email"] . "<br />";
echo $_POST["website"] . "<br />";
echo $_POST["comment"] . "<br />";
echo $_POST["gender"] . "<br />";
?>
```

Programing your own conditions and error messages

In the preceding example, required field were specified in the input tags. Browsers have there own way to telling users a field is required. Although, you can specify you own condition and message using PHP in the first condition using the following structure.

```
<body>
<?php
$nameError = $emailError = $genderError = $websiteError = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
        if (empty($_POST["name"])) {
                $nameError = "Name is required";
        } else {
                $name = test_input($_POST["name"]);
                // existing command
 }
?>
</body>
```

To display the error messages:

```
...
<label>Name:</label>
<input type="text" name="name" required>
<span style="color:red">* <?php echo $nameErr;?></span>
...
```

Validating name input and programing an error message

```
<?php
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
        $nameError = "Please use letters and white spaces only";
}
```

preg_match()
This function inspects a string for pattern. It returns *true* if a given pattern is found and *false* if not.

Validating email input and programing an error message

```
<?php
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $emailError = "Please use a valid email format";
}
```

filter_var()
Many PHP filter functions and constants are available. A selected list of those will be supplied to you. This one verifies that the email format is respected.

Validating url input and programing an error message

```
<?php
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
        $websiteError = "Please provide a valid URL";
}
```

Summing it up

```
<?php
$nameError = $emailError = $genderError = $websiteError = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
        if (empty($_POST["name"])) {
                $nameError = "Name is required";
        } else {
                $name = test_input($_POST["name"]);
                if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
                        $nameError = "Please use letters and white spaces only";
                }
        }
...
}
...
```

filter_var()

The filter_var() function validates and sanitizes data coming from insecure sources such as users inputs. I filters one variable using a specific filter. To do so, you need to specify the variable to validate and the type of validation to be used.

```
<?php
$string = "<strong>Hello World!</strong>";
$finalString = filter_var($string, FILTER_SANITIZE_STRING);
echo $finalString;
?>
```

Explanation :
Using the constant FILTER_SANITIZE_STRING on a specific variable deletes all HTML tags from the associated string, leaving only the text.

Constants used with filter_var() (selection) :

FILTER_VALIDATE_EMAIL	Validates an e-mail address
FILTER_VALIDATE_FLOAT	Validates a float
FILTER_VALIDATE_INT	Validates an integer
FILTER_VALIDATE_IP	Validates an IP address
FILTER_VALIDATE_URL	Validates a URL
FILTER_SANITIZE_EMAIL	Removes all illegal characters from an e-mail address
FILTER_SANITIZE_ENCODED	Removes/Encodes special characters
FILTER_SANITIZE_MAGIC_QUOTES	Apply addslashes()
FILTER_SANITIZE_NUMBER_FLOAT	Remove all characters, except digits, +- and optionally .,eE
FILTER_SANITIZE_NUMBER_INT	Removes all characters except digits and + -
FILTER_SANITIZE_SPECIAL_CHARS	Removes special characters
FILTER_SANITIZE_STRING	Removes tags/special characters from a string
FILTER_SANITIZE_URL	Removes all illegal character from s URL

Assignment 5 : Using form to update an external file

In a HTML document, use PHP to create a form that will allow users to input their full name, city, and comment.

Upon submit, the data will be added (append) to existing data in another PHP page.

Finally, the external page will show its results underneath the form, most recent first, in the first page, like it would happen in a blog, for instance.