PHP Chapter 4 : Manage your data throughout your PHP web application

Table of Contents

Introduction : why ?

Use URL Variables to Pass Data: $_GET

Sending and receiving HTML form data: $_POST

PHP sessions : $_SESSION

Working with secured data (forms, URL, session)

Case Study

- Basic principle of a Web Application working with a Database : HTML forms to enter data
- Displaying data

Introduction : why ?

In many situations, or within your web application, sending data to another web page may be necessary, but also ensuring that data are available and accessible throughout the navigation within the web site.

Among différents possibilities, PHP offer :

From Client Side to Server:

- Attached variables to the URL
- Sending forms, usual form fields which are completed from the user, but also hidden input field to pass data from one form to another

Server side only, with sessions. Data are stored on the server, to be used across multiple pages of your site. At least a session key is stored on the Client Side, using a tiny cookie

PHP and Javascript can also handle Client Side cookies.

Javascript can now handle Local Storage, i.e. a type of web storage on the Client Side.

Use URL Variables to Pass Data: $_GET

Constructing URL Variables

For a given link to the URL web page : *myWebPage.php*
To pass a variable and its value, just pass the variable name / value pair : *myWebPage.php*?my_variable=value

Operators

? : the question mark is placed at the end of the URL, immediately followed by variable / value pair

& : the ampersand character allows to pass subsequent variable / value pairs.

= : the equals sign separates the variable from its value

myWebPage.php?my_variable1=value1&my_variable2=value2&my_variable3=value3

If possible, limit the length of the string consisting of variable / value pairs to 256 characters.

PHP $_GET

PHP makes URL variables available in the associative array $_GET, as a super global variable, which exists everywhere in your script, even in functions.
The array keys are the variable name :

$_GET['my_variable1'] array element has the value value1

$_GET['my_variable2'] array element has the value value2

Be careful, data passes via the URL may be transformed. Never trust received data, check them first!

Sending and receiving HTML form data: $_POST

History : PHP/FI "Personal Home Page/Forms Interpreter". This included Perl-like variables, **form** handling, and the ability to embed HTML.

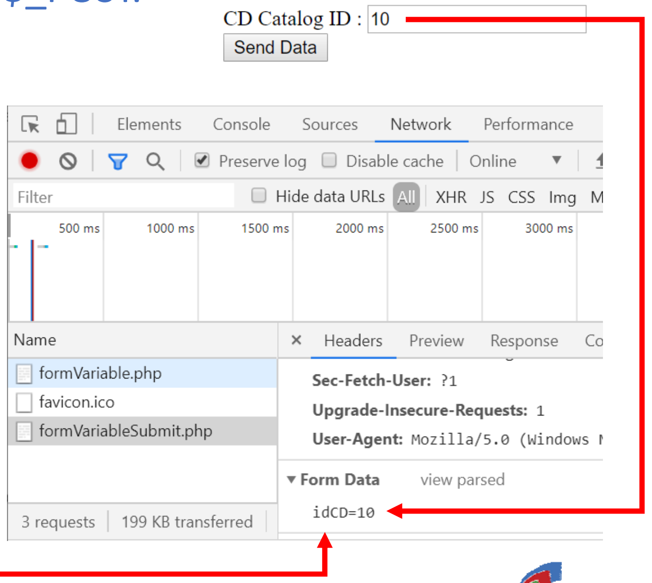
One of the most common tasks in PHP, a language largely designed for this purpose at first, is to be able to process responses from an HTML form.

The following HTML form, named *formVariable.php*, will submit one variable to the *formVariableSubmit.php* script.
The *post* method is used, data are sent as a text file (an http header and a text message), sent data can only be processed in PHP.
The *get* method adds data as URL variables, sent data can be processed with Javascript and PHP.
The value of the *name* attribute will be used in PHP as the index of an associative array $_POST.

Front-End Web Programming 1

REMINDER

```
<form action="formVariableSubmit.php" method="post">
CD Catalog ID : <input type="number" name="idCD"><br>
<button>Send Data</button>
</form>
```


CD Catalog ID : 10
Send Data

formVariable.php
favicon.ico
formVariableSubmit.php

Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows

▼ Form Data   view parsed
idCD=10

3 requests | 199 KB transferred

In PHP, **$_POST** is a reserved variable used to store all information submitted by the POST HTTP method.
It is a **super global** variable, it means that $_POST always exists in your script, even in functions.

Reminder, in a function, declared variables are local, and exist only in the scope of the function, not outside.

In your script, variable are global, accessible anywhere in your script, but to use them in a user-defined function, you need to declare it inside the function with the *global* keyword.

```php
<?php

var_dump($_POST);

?>
```

```
array (size=1)
  'idCD' => string '10' (length=2)
```

The value of the submitted form can be used in a SELECT query, for instance. Be aware that the following example does not take into account a possible security risk (SQL injection).

```php
include("connectDB.inc.php");

$query="SELECT * from catalog WHERE idCB=".$_POST['idCB'];
$result=mysqli_query($link,$query);
```

```php
<?php
$host="localhost";
$user="root";
$passwd="";
$db="ufaz_php_pw";

//@ to prevent a warning display
$link=@mysqli_connect($host,$user,$passwd,$db);

if(mysqli_connect_errno()){
    echo "Error No :".mysqli_connect_errno()." ,
        Msg : ".mysqli_connect_error()."<br>";
    exit();
}

//encoding
mysqli_set_charset($link,'utf8');

?>
```

To connect to the MySQL server, the same connection script, named *connectDB.inc.php*, can be reused for all our scripts, using the include() function.

PHP sessions : $_SESSION

The information is stored in files, on the server, for each session there is a corresponding file. Each session is designated by a name and an identifier. When the visitor accepts cookies, the session ID is stored in a cookie.

Start a PHP Session

This code is used to start a session. If a file exists on the server for this session, the session variables will be retrieved, if not, a new file will be created.

```php
<?php
session_start();
```

global variable: $_SESSION

Assign values to the super global session variable $_SESSION

```php
$_SESSION['variable'] = $valeur ;
```

Get PHP Session Variable Values

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()). var_dump() allows you th check your $_SESSION for debug.

```php
var_dump($_SESSION);
```

Working with secured data

Basic principle of a Web Application

Basic principle of a Web Application working with a Database : HTML forms to enter data

Displaying data

See lecture…