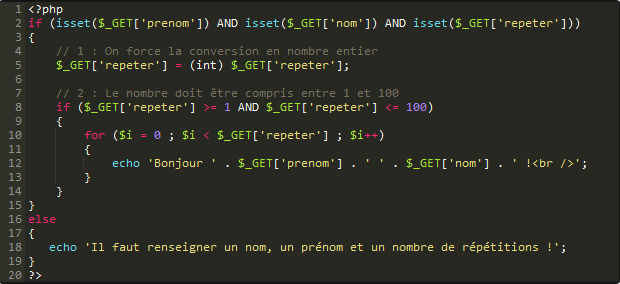
**Protecting GET statements**

On the receiving script, where we want an integer input for the keyword ‘repeter’ , we could write :

http://i.gyazo.com/b0e821127c23eb4858bfb5ff28f621aa.png

This will forcefully transform the input get into an integer (0 if the value type is illegal). In the end, we would have a code like this:



**array\_count\_values($array)**: counts the frequencies of each distinct value inside the array

**ord($str)** : return the ASCII value of a string.

**array\_map(my\_function(), $array)** :

**Forms**

Remember, in a form input, name= is the attribute that is going to become the key of the value=.

For input type “checkbox”, the value we are gonna send is on or off, depending whether it’s selected or not.

http://i.gyazo.com/67f0e0d0e2883210e83bc4009b265f13.png

Radio inputs are groups of input and you can only select one. Remember to all give them the same name, because they work as a group!

Sometimes you wanna transmit constant information each time a form is submitted, like for a counter or smthg. You’ll use the hidden input, which has a fixed value.

http://i.gyazo.com/79a5663492ba8b97f6c8309ebd500a36.png

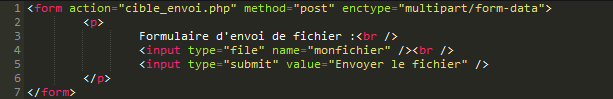
Beware of XSS exploitation! any user can do anything he wants with your forms (c.f realistic 1 with rock band where you just changed the value of the select to pump up the votes) or even put html code in it

That’s why you gotta take an extra precaution. Just like you used (int) to transform input into an integer, use htmlspecialchars to automatically transform any html tag into ascii:

http://i.gyazo.com/432b1ab537e8e910e2a9cbf7f8602d7c.png

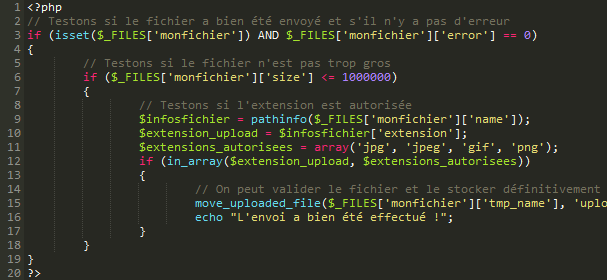
**Sending files**

You can create form that send files:



You are going to catch them with $\_FILES[‘input\_name’] function, and accept them with the move\_uploaded\_file() function.

Safe-proof it like this:



You test whether a file has been sent and that there has been no error, then that the file isn’t too big, and eventually you compare the file extension with an array of authorized filetypes to check whether everything is alright.

(remember to set the rights of the upload folder on your folder to 733)

Check out more: <http://openclassrooms.com/courses/upload-de-fichiers-par-formulaire>

**Sessions, cookies et variables superglobales**

Superglobal variables are accessible from anywhere within your script. Here is a list of some of them:

**$\_SERVER**: Are all the data sent by the server. One very useful is $\_SERVER[‘REMOTE\_ADDR’], which traces the IP of the client that is requesting to view the page.

**$\_ENV**: mostly used on LINUX servers, but let’s not get too technical here.

**$\_SESSION**: all the date stored manually when you initiate a session. Info stored in this supervariable survives through all the pages of the website.

**$\_COOKIE**: same here, but fills up when you declare cookies instead.

**$\_GET, $\_POST**: basic stuff, bro.

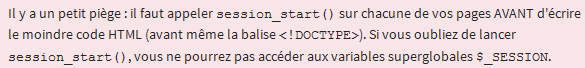
**$\_FILES**: contains the info of the files sent via the immediate previous form.

Sessions

Each session is surrounded by two commands:

session\_start();

session\_destroy();



So write it really on top of your script, especially before the ‘include(view.phtml)’

Of course, don’t do it on all the pages. Once a session is started, its very purpose is to stay alive through the client’s navigation.

One cool thing about sessions is that you can implement variable degrees of authorization: some users will have access to certain pages other users won’t.

http://i.gyazo.com/2cacbf10bbdc6dbd7af6e8e0d72d75c9.png

Cookies

Cookies are a bit different from sessions. First, the data is stored on the client’s computer, not on the server.

It can be used for targeted advertising by saving the user’s preferences.

You create a cookie with the function **setcookie(cookie\_name, cookie\_value, expiration\_date)**

The value is what you put inside the cookie, and the expiration date has to be a UNIX Timestamp. For example, if you want to set the expiration date of the cookie one year from now, you take the time now and multiply it by the number of seconds present in a year:

http://i.gyazo.com/1e488b7fba3db76ca5d725b5516763be.png

Now, let’s talk security. In basic 10 you used javascript in the URL bar to change the value of a cookie (password\_value) in order to enter a restricted area. You can actually prevent that easily by using *httponly*, like so:

http://i.gyazo.com/bd042fb64e8585566aca31088d2e0158.png

The last true parameter activates http only, rendering it inaccessible for javascript, who won’t be able to modify the cookie.

Again , set your cookies *before* you write any html code, on top of your script.

Once all that’s done, you access the cookie by calling its name in the supervariable $\_COOKIE:

http://i.gyazo.com/37fd9ff168715cd89d34665b22780625.png

Remember to put isset() before $\_COOKIE, because if no cookie has yet been created, the supervariable will be non-existent.

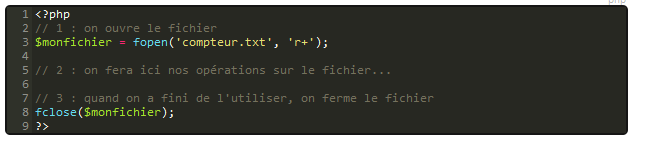
*Note: Any client can manipulate cookies. Ironically, you should be as afraid of cookies as clients are afraid of them!*

**Modifying a cookie**: Simply create a new cookie with the same name. Each time you want to want to modify a cookie, use the **setcookie(‘same name’, ‘new value’, nul, null, false, true)** function

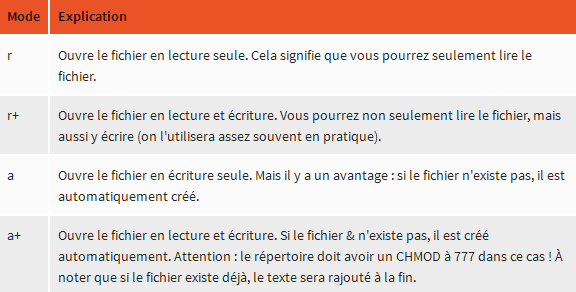
**Lire et écrire dans un fichier**

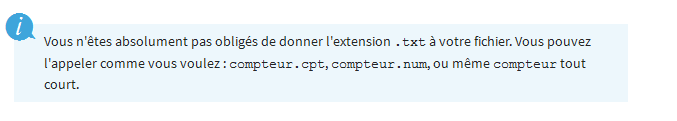
First, create an empty .txt file and load it up on your server. Set CHMOD auth to 777.

Your code is going to be inserted between the file\_open and file\_close functions



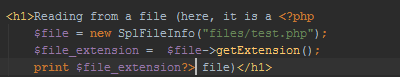
The r+ refers to the mode of opening. Here you can access the informations inside the file and add new ones inside. There are a few other commands like this:





*Note: getting the extension of a file and returning it:*

*SplFileInfo is an object which stores a lot of data about the object.*



Now that you have specified the authorizations and created the bulk of your function, how are you actually gonnna access and write into the file? You are gonna use the fget function: fgets gets only the first line. If you wanna read more, you have to do a loop.

Writing in a file

you use fput. Really cool, but it will put text only at the last line, and will overwrite the last entry you put unless there is a fclose and fopen between the two calls. You can insert all kind of shit: I inserted some php brackets+php code into a php file, and now, the php script displays that code FOREVER! Muhaha.