**Recapitulatif semaine 2**

Cette semaine nous avons vu l’utilisation des constantes

**<?php***/\*\*  
 \* On peut définir des variables en tant   
 \* que constantes:  
 Elles sont similaires aux variables   
 \* sauf qu'elles ont vocation à rester  
 \* constantes  
 Peuvent être definie de deux façon  
 \* le deuxième exemple est à privilégier  
 Il est recommandé d'utiliser les   
 \* majuscules pour les définir  
 \*/*//const PHP\_MA\_CONSTANTE = 256;  
//define('PHP\_MA\_CONSTANTE',512);  
  
//echo PHP\_MA\_CONSTANTE;  
  
  
*/\*\*  
 \* PHP vient déjà avec une panoplie  
 \* de constantes qui ne seront pas forcément  
 \* utilisé si vous commencé à coder  
 \* Cependant les constantes magiques peuvent  
 \* être utile à n'importe quel stade  
 \* La syntaxe des constantes magiques  
 \* sont définies toujours avec deux underscore  
 \* au début  
 \* et deux underscore à la fin  
 \* Par exemple: \_\_FILE\_\_  
 \*  
 \* \_\_LINE\_\_ Le numéro de ligne actuel.  
 \* \_\_FILE\_\_ Le chemin d'accès absolu au fichier  
 \* \_\_DIR\_\_ Le dossier du fichier equivalent de dirname(\_\_FILE\_\_)  
 \* \_\_FUNCTION\_\_ Nom de la fonction  
 \* \_\_CLASS\_\_ Nom de la classe //on le verra plus tard  
 \* \_\_METHOD\_\_ Nom de la méthode //on le verra plus tard  
 \* \_\_NAMESPACE\_\_ Nom du namespace //on le verra plus tard  
 \*  
 \*/***function** maFonction()**{** // fonction indique le nom de la fonction utilisée  
 **echo** *\_\_FUNCTION\_\_*; //maFonction  
 **echo** '<br>';  
  
 **echo** *\_\_LINE\_\_*;// line 47  
**}**maFonction();  
  
**echo** *\_\_FILE\_\_*;  
**echo** '<br>';  
**echo** *\_\_DIR\_\_*;  
**echo** '<br>';  
**echo** dirname(*\_\_FILE\_\_*);

Nous avons vu dans une deuxième temps les constantes statiques, une fois la fonction exécutée la variable n’est pas effacé. Elle est gardée toujours en mémoire et peut être réutilisée si la fonction est appelée à nouveau.

**<?php**//static variable  
  
**function** test()  
**{  
 static** $count = 0;  
 **echo** $count;  
 $count++;  
**}**test();  
**echo** "<br>";  
test();

On a fait une explication de unset() qui est une fonction qui sert à détruire une variable. Si on veut faire un appel à cette variable elle sera considérée comme non définie.

**<?php***/\*\*  
 \* Unset explication de unset   
 \* le desctructeur de variables.  
 \*/*$var = 4;  
//La fonction unset sert à détruire  
//des variables  
**unset**($var);  
  
**if**(**isset**($var))**{  
 echo** 'variable definie';  
**}else{  
 echo** 'variable non definie';  
**}**

Alias de variables, attention ici la ligne  :

**echo** $$a."-"; sera lue comme $ . ‘b’ ;

**echo** $$$a."-"; sera lue comme $ . ‘$b’ ; qui est équivalent à comme $ . ‘c’

**<?php**$a="b";  
$b="c";  
$c="d";  
$d="e";  
$e="f";  
  
**echo** $a."-";  
**echo** $$a."-"; // $b  
**echo** $$$a."-"; // $$b ou $c  
**echo** $$$$a."-"; // $$$b ou $$c ou $d  
**echo** $$$$$a; // $$$$b ou $$$c ou $$d ou $e

Variables globales.

Elles sont accessibles dans un contexte global.

**<?php**$b = 2;  
  
*/\*\*  
 \* On déclare une variable globale dans une fonction si on veut y avoir accés  
 \* dans un contexte globale  
 \*/*//function Sum()  
//{  
// global $a;  
// $a = 'gerard';  
//}  
//  
//Sum();  
//echo $a;

Avec $\_GLOBALS

*/\*\*  
 \* On peut y avoir accés également à partir de la constante globale $\_GLOBALS  
 \* qui stocke toutes les constantes qui sont définie  
 \*/*$a = "alfonso";  
$test = 12;  
**function** test()  
**{** $GLOBALS['a']="fred";  
 **echo** $GLOBALS['test'];//12  
**}  
echo** '<br>';  
test();  
**echo** $a;  
//Conseil:Eviter d'utiliser des variables globales  
//On perd la trace du code qui a opéré le  
//changement

Les constantes globales sont générée par le système et on peut y avoir accès quand on le voudra :

*/\*\*  
 \* En fait PHP contient déjà plusieurs variables globales qui ont des informations importantes sur le système  
 \* Elles sont structurées en tableau associatifs on le reverra plus amplement un peu plus tard.  
 \*  
 \* $GLOBALS Toutes les variables sont définies dans le scope/étendue globale du script. Les variables sont nommées en tant que clés d'un tableau(array)  
 \* $\_SERVER Contient des information sur les headers(en tête), paths(chemin fichiers), et script de localisation. Les entrées sont créées par le serveur et il n'y a pas de garantie que tout les serveur créé ceci  
 \* $\_GET Variables passées au script via HTTP GET.  
 \* $\_POST Variables passées au script via HTTP POST.  
 \* $\_FILES objets/fichiers passés via HTTP POST.  
 \* $\_COOKIE Variables passées via HTTP cookies.  
 \* $\_SESSION Session variables disponible pour le script courrant.  
 \* $\_REQUEST Contient des informations du navigateur par défaut $\_GET, $\_POST, et $\_COOKIE.  
 \* $\_ENV Variables passées dans le script courrant via la methode environment. // On le verra plus tard  
 \*/*

ASTUCE conditionnelle :

*/\*\*\*  
 \* Explication de l'astuce conditionnelle  
 \*/*$var = "file";  
  
**if**( 0 === $var )**{  
 echo** "equals";  
**}else{  
 echo** "different";  
**}**

Dans l’exemple précédent nous voyons que lors de la condition on met en première position la constante comparative. Ça évite de faire l’erreur de syntaxe $var = 0 qui donne vrai alors que 0 = $var nous donne une erreur et c’est ce qu’on veut – avoir le warning.

Contraction conditionnelle

$strict = **false**;  
$val = 1;  
$val2 = **true**;  
  
var\_dump($strict ? $val === $val2 : $val == $val2);

Si strict = true alors la condition $val === $val2 sera vérifié / checkée

Inversement si strict = false alors la condition $val == $val2 sera vérifiée / checkée

Révision des insertion d’information de tableau dans une chaines de caractères :

**<?php**$book = [  
 'title' => "The Hitchhiker's Guide to the Galaxy",  
 'author' => 'Douglas Adams',  
 'description' => 'a comedy sci-fi adventure originally based on a BBC radio series'  
];  
$string = "**{**$book['title']**}** by **{**$book['author']**}**";

On utilize des crochets pour pouvoir afficher des valeurs de tableau sans devoir concatener avec d’autres guimets ou apostrophe/points.

**OPERATION SUR DES TABLEAUX**

Il existe une multitude de fonctions qui opèrent sur des tableaux. Nous avions vu que quelques exemple : array\_values()

**<?php**//array values -> reindex les clés  
$x = **array**(1, 2, 3, 4, 5);  
//avant  
var\_dump($x);  
**unset**($x[3]);  
$x = array\_values($x);  
//après  
//pratique pour pas avoir de trou dans le tableau  
var\_dump($x);  
**die**();

Output

**array** *(size=5)*

0 => int 1

1 => int 2

2 => int 3

3 => int 4

4 => int 5

**array** *(size=4)*

0 => int 1

1 => int 2

2 => int 3

3 => int 5

array\_push() ;

« pousse » les données que vous voulez dans un nouvelle index du tableau, array est le tableau cible et array2 les données qu’on désire pousser. Il se trouve ici qu’il s’agit d’un tableau.

//array push  
// permet d'insérer un tableau dans  
// l'un des index d'un autre tableau  
$array = **array**(1,2,3,4);  
$array2 = **array**(45, 46, 47,48);  
array\_push($array,$array2);  
  
var\_dump($array);

Output

**array** *(size=5)*

0 => int 1

1 => int 2

2 => int 3

3 => int 4

4 =>

**array** *(size=4)*

0 => int 45

1 => int 46

2 => int 47

3 => int 48

array\_combine()  combine deux tableau pour un donner un troisième. Avec les valeurs du premier tableau qui sera utilisé comme clés/index et les valeurs du deuxième qui seront utilisées comme valeurs.

//array\_combine  
// permet d'annexer un tableau en tant  
//que clé avec un autre qui servira comme valeur  
$array = **array**(2,4,6,5);  
$array2 = **array**(45, 46, 47,48);  
$c = array\_combine($array,$array2);  
  
var\_dump($c);

**array** *(size=4)*

2 => int 45

4 => int 46

6 => int 47

5 => int 48

//array\_merge  
//permet de fusionner deux tableau  
$array = **array**(2,4,6,5);  
$array2 = **array**(45, 46, 47,48);  
$c = array\_merge($array,$array2);  
  
var\_dump($c);

**array** *(size=8)*

0 => int 2

1 => int 4

2 => int 6

3 => int 5

4 => int 45

5 => int 46

6 => int 47

7 => int 48

$arr = ['prof'=>'Alfonso','eleve'=>'quelquun'];  
//array\_key\_exists-> recherche de clés  
var\_dump(array\_key\_exists('prof',$arr)); //vrai  
  
  
//in\_array équivalent de array\_search ->recherche de valeur in\_array  
// retourne vrai ou faux array\_search renvoie la clé  
var\_dump(in\_array('Alfonso',$arr));//vrai  
//renverra la clé  
$result = array\_search('Alfonso',$arr);  
var\_dump($result);

Challenge recherche de valeurs dans un array multidimensionnelle

**<?php**$a = **array**(  
 **array**('ordi'=>"Mac"),  
 "NT",  
 "Irix",  
 "Linux"  
);

// needle la valeur a chercher  
// haystack c'est le tableau  
  
**function** array\_searcher($needle, $haystack, $strict = **false**) **{  
 foreach** ($haystack **as** $item) **{  
 if** (($strict ? $item === $needle : $item == $needle)  
 || (is\_array($item) && array\_searcher($needle, $item, $strict))) **{  
 return true**;  
 **}  
 }  
 return false**;  
**}**$res = array\_searcher('Mac',$a,**true**);  
  
var\_dump($res);

Utilisation de lists().

List éclate un tableau avec le nom de variables qu’on lui passe:

**<?php***/\*\*  
 \* Le mot clé list transforme les array en variables réutilisables  
 \*/*$info = **array**('cafe', 'noir', 'cafeine');  
  
// Listing all the variables  
**list**($boisson, $couleur, $energie) = $info;  
**echo** "$boisson est $couleur et sa $energie est speciale.\n";

Ouput

Cafe est noir et sa cafeine est speciale

Update sur les conditions les mots clés AND , OR et XOR existe

**<?php**

$x = **true**;  
$y = **true**;  
  
//ça existe aussi de   
//if($x or $y){  
//if($x and $y){  
**if**($x **xor** $y)**{  
 echo** "true";  
**}else{  
 echo** "false";  
**}**

**LES OPERATEURS BINAIRES**

Les opérateurs binaires existe pour faire des opérations sur des binaires. Principalement utilisés pour gérer des problèmes accès mémoire en C++, en PHP ils ont une utilités si on a un peu d’imagination. Sachez qu’ils existent.

**<?php***/\*\*  
 \* Les opérateurs binaires  
 \* destinés à opérer sur la conversion binaires des variables  
 \* & les bits sont définis dans les deux variables $x et $y  
 \* | les bits sont définis soit dans les deux variables $x et $y ou l'une d'entre elle  
 \* ^ les bits sont définis dans l'un des deux $x ou $y pas dans les deux  
 \* ~ les bits qui sont définis dans $x ne le sont pas  
 \* <<  
 \* >>  
 \*/*//echo 12 ^ 11;  
//die();  
//Ça doit être des doubles pour marcher  
**const** *UTILISATEUR* = 1;  
**const** *PREMIUM* = 2;  
**const** *ADMIN* = 4;  
**const** *SUPER\_ADMIN* = 8;  
**const** *ANONYME* = 16;  
  
$state = *UTILISATEUR* | *SUPER\_ADMIN*;  
  
**echo** "Valeur: $state<br>";  
**echo** "Suis-je un super utilisateur? "  
.(bool)($state & *UTILISATEUR*)."<br>";  
**echo** "Suis-je un super premium? "  
.(bool)($state & *PREMIUM*)."<br>";  
**echo** "Suis-je un super super admin? "  
.(bool)($state & *ADMIN*)."<br>";  
**echo** "Suis-je un super admin? ".  
 (bool)($state & *SUPER\_ADMIN*)."<br>";  
**echo** "Suis-je un super anonyme? "  
 .(bool)($state & *ANONYME* )."<br>";  
**echo** "Suis-je un super utilisateur ou superadmin? "  
.($state == (*UTILISATEUR* | *SUPER\_ADMIN*))."<br>";  
  
// https://web.stanford.edu/class/cs101/bits-bytes.html  
  
*/\*\*  
$x=64;  
$y=4;  
echo $x >> $y;  
 \*\*/  
 /\*\*  
 \* Exemple pour 1 | 8  
 \*  
 \* 128 64 32 16 8 4 2 1  
 \* 0 0 0 0 0 0 0 1 = 1  
 \* 0 0 0 0 1 0 0 0 = 8  
 \* --------------------------------------  
 \*T 0 0 0 0 1 0 0 1 = 9  
 \*  
 \* Exemple pour 13 & 22  
 \* 128 64 32 16 8 4 2 1  
 \* 0 0 0 0 1 1 0 1 = 13  
 \* 0 0 0 1 0 1 1 0 = 22  
 \* --------------------------------------  
 \*T 0 0 0 0 0 1 0 0 = 4  
 \*  
 \* Exemple pour 12 ^ 11  
 \* 128 64 32 16 8 4 2 1  
 \* 0 0 0 0 1 1 0 0 = 12  
 \* 0 0 0 0 1 0 1 1 = 11  
 \* --------------------------------------  
 \*T 0 0 0 0 0 1 1 1 = 7  
 \*  
 \* Exemple avec 12 & ~ 10  
 \* 128 64 32 16 8 4 2 1  
 \* 0 0 0 0 1 1 0 0 = 12  
 \* 0 0 0 0 1 0 1 0 = 10  
 \* --------------------------------------  
 \*T 0 0 0 0 0 1 0 0 = 4*

*\*  
 \* Exemple avec 8 << 3  
 \* 128 64 32 16 8 4 2 1  
 \* 0 0 0 0 1 0 0 0 = 8  
 \* 8 est multiplié trois fois par 2  
 \*  
 \* 128 64 32 16 8 4 2 1  
 \* 0 1 0 0 0 0 0 0 = 64  
 \*  
 \* Exemple avec 64 >> 4  
 \* 128 64 32 16 8 4 2 1  
 \* 0 1 0 0 0 0 0 0 = 8  
 \* 8 est multiplié trois fois par 2  
 \*  
 \* 128 64 32 16 8 4 2 1  
 \* 0 0 0 0 0 1 0 0 = 2  
 \*  
 \*/*

**Fonction alias**

Comme pour les variables les fonctions peuvent aussi avoir des alias.

**<?php  
  
function** test()  
**{  
 echo** "Hello World";  
**}**$nomFonction = 'test';  
  
$nomFonction(); //Hello World

**Passer des variables à une fonction anonyme**

**<?php**$message = 'hello';  
$var = 256;  
  
// Notez l utilisation de use  
$example = **function** () **use** ($message,$var2) **{** var\_dump($message);  
 var\_dump($var2);  
**}**;  
**echo** $example();

**Les mots clés Callable et Closure**

Callable en prefixe d’une variable en arguments dans la définition d’une fonction exige que l’argument soit appelable c’est-à-dire qu’elle soit une fonction.

Closure de même mais que la fonction soit anonyme.

**<?php***/\*\*  
 \* Definir d'abord ce qu'est une fonction anonyme indépendante  
 \*/***function** func1(Closure $closure) **{** $closure();  
**}  
  
function** func2(**Callable** $callback) **{** $callback();  
**}**$function = **function**() **{  
 echo** 'Hello, World!';  
**}**;  
  
//les deux marches  
func1($function);  
func2($function); // Hello, World!  
  
**function** bye() **{  
 echo** 'Bye, World!';  
**}**;  
  
func1('bye'); //erreur  
func2('bye'); // Hello, World!

**Definition d’une function, affecter une valeur par défaut à une variable**

**<?php**/\*$var = array(8);  
  
function testFunction($var)  
{  
 $var2 = (int) $var;  
 return $var2;  
}  
  
$result = testFunction($var);\*/  
  
*/\*\* On peut affecter à une variable un valeur par défaut  
 \* lors de la définition d'une fonction  
 \*  
  
 \*\*/***function** valeurParDefaut($var2=**NULL**)**{  
 return** $var2;  
**}**$result = valeurParDefaut();  
  
var\_dump($result);  
**die**();  
  
*/\*\*  
 \* FORCER LES TYPE THE ARGUMENTS DANS LES FONCTION  
 \*  
 \*  
 \* Les types des arguments dans les fonctions peuvent être un nom de classe, la déclaration array, la déclaration callable  
 \*  
 \*/*

**Les fonctions func\_get\_arg() et func\_num\_args()**

Func\_get\_arg() recupère tout les arguments passés dans une fonction lors de son appel même si elle ne sont pas préalablement définies.

Func\_num\_args() compte le nombre d’arguments passés.

Par exemple ces deux fonction sont équivalente :

**<?php  
function** mean($numbers) **{** $sum = 0;  
 $size = count($numbers);  
 **for** ($i = 0; $i < $size; $i++) **{** $sum += $numbers[$i];  
 **}** $average = $sum / $size;  
 **return** $average;  
**}**// $moyenne est de 96.25  
$mean = mean(**array**(96, 93, 98, 98));  
var\_dump($mean);  
  
  
**function** mean2() **{** $sum = 0;  
 // nombre d'arguments passés  
 $size = func\_num\_args();  
  
 **for** ($i = 0; $i < $size; $i++) **{** //piocher les arguments  
 $sum += func\_get\_arg($i);  
 **}** $average = $sum / $size;  
 **return** $average;  
**}**$mean = mean2(96, 93, 98, 98);  
var\_dump($mean);

**Contrôles des types**

Les fonctions suivantes permettent de controller les types

**<?php**$var = **NULL**;  
**echo** '$var = '.$var.' <br/>';  
**echo** "isset ->".**isset**($var)."<br/>";  
**echo** "is\_numeric ->".is\_numeric($var)."<br/>";  
**echo** "is\_null ->".is\_null($var)."<br/>";  
**echo** "is\_string ->".is\_string($var)."<br/>";  
**echo** "is\_int ->".is\_int($var)."<br/>";  
**echo** "is\_float ->".is\_float($var)."<br/>";

**Emtpy() vs isset()**

**<?php***/\*\*  
 \* Extrait de* ***http://php.net/manual/en/types.comparisons.php*** *\*  
 \* Expression empty() isset()  
 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* $x = ""; TRUE TRUE  
 \* $x = null; TRUE FALSE  
 \* $x is undefined TRUE FALSE  
 \* $x = false; TRUE TRUE  
 \* $x = array(); TRUE TRUE  
 \*/*

**Quelques fonctions sur les chaînes de caractères**

**<?php***/\*\*  
 \* Chaînes de caractères  
 \* Fonction strlen obtient la longeur d'une CC  
 \*/*header("Content-Type: text/html; charset=UTF-8");  
  
$phrase = 'Salut tout le monde! Vous '  
 . 'êtes prêt pour un nouveau '  
 . 'challenge?';  
$longueur = strlen($phrase);  
//calcul la longeur de la chaîne  
**print**($longueur . ' caractères :' . $phrase);  
  
*/\*\*  
 \* str\_replace remplace une CC dans une autre  
 \*/*$ma\_variable = str\_replace('aimez', 'détestez', 'Vous aimez le chocolat?');  
**echo** $ma\_variable;  
  
*/\*\*  
 \*strtolower tranforme toutes les lettres en minuscules  
 \* strtoupper fait exactement l'inverse.  
 \*/*$phrase = 'ALORS COMMENT VA VOTRE JAVASCRIPT?';  
$phrase = strtolower($phrase);  
  
**echo** $phrase;  
  
*/\*\*  
 \* affiche "tout le"  
 \* deuxieme paramètre debut de chaine  
 \* troisieme parametre longueur  
 \*/***echo** substr("Salut tout le monde",6,7);

**$\_GET et $\_POST**

Get et post font parties des variables globales qui sont appelable à n’importe quel moment dans le code.

**<?php***/\*\*  
 \* En fait PHP contient déjà plusieurs variables globales qui ont des informations importantes sur le système  
 \* Elles sont structurées en tableau associatifs on le reverra plus amplement un peu plus tard.  
 \*  
 \* $GLOBALS Toutes les variables sont définies dans le scope/étendue globale du script. Les variables sont nommées en tant que clés d'un tableau(array)  
 \* $\_SERVER Contient des information sur les headers(en tête), paths(chemin fichiers), et script de localisation. Les entrées sont créées par le serveur et il n'y a pas de garantie que tout les serveur créé ceci  
 \* $\_GET Variables passées au script via HTTP GET.  
 \* $\_POST Variables passées au script via HTTP POST.  
 \* $\_FILES objets/fichiers passés via HTTP POST.  
 \* $\_COOKIE Variables passées via HTTP cookies.  
 \* $\_SESSION Session variables disponible pour le script courrant.  
 \* $\_REQUEST Contient des informations du navigateur par défaut $\_GET, $\_POST, et $\_COOKIE.  
 \* $\_ENV Variables passées dans le script courrant via la methode environment. // On le verra plus tard  
 \*/*

$\_GET est utilisé pour capturer des paramètre provenant de l’url

Exemple :

**<?php**  
$parametres = "?username=alfonso&telephone=0468922928&number=10";  
  
**?>**<html>  
 <head>  
  
 </head>  
 <body>  
 <a href="myspace.php**<?=**$parametres**?>**">Link to my space</a>  
 </body>  
</html>

Myspace.php

**<?php**//var\_dump($\_GET);  
$username = **isset**($\_GET['username'])?  
 $\_GET['username']:NULL;  
$telephone = **isset**($\_GET['telephone'])?  
 $\_GET['telephone']:NULL;  
//force le paramètre a être un chiffre  
$number = (int) $\_GET['number'];  
**?>**<html>  
 <head>  
 <title>My Space</title>  
 </head>  
 <body>  
 **<?php  
  
 if** (**isset**($username)   
 **AND isset**($telephone)   
 **AND isset**($number)):  
   
 **if**(is\_int($number)):  
 **for** ($i = 0 ; $i < $number ; $i++):  
 **echo** "<p>$i</p>";  
 **echo** "<p>Bienvenu $username</p>";  
 **echo** "<p>Votre numero de telephone est $telephone</p>";  
 **endfor**;  
 **endif**;  
 **else**:  
 **echo** 'username, tel and number are required';  
 **endif**;  
  
 **?>** </body>  
</html>

Un post récupère les données passés par les entêtes lors d’une requête http. Généralement passé via la soumission d’un formulaire :

**<?php** var\_dump($\_POST);  
   
 **if**(**isset**($\_POST))**{** $principle = $\_POST['principle'];  
 **}  
?>**<html>  
 <head>  
  
 </head>  
 <body>  
 <form method="post" action="form.php">  
 <pre>  
 Loan Amount <input type="text" name="principle">  
 Monthly Repayment <input type="text" name="monthly">  
 Number of Years <input type="text" name="years" value="25">  
 Interest Rate <input type="text" name="rate" value="6">  
 Describe your thoughts <textarea name="description"></textarea>  
 Insurance <input type="checkbox" name="option[]" value="45" \>  
 <input type="checkbox" name="option[]" value="69" \>  
 <select name="choix">  
 <option value="choix1">Choix 1</option>  
 <option value="choix2">Choix 2</option>  
 <option value="choix3">Choix 3</option>  
 <option value="choix4">Choix 4</option>  
 </select>  
 <input type="submit" />  
  
 </pre></form>  
 </body>  
</html>

**Les sessions**

Les sessions sont un moyen de passes des infos d’une page à une autre sans passer par les url ou des post mais elles sont souvent la conséquence des méthodes post et get.

Attention le server gère les sessions provenant de différent sites comme les mêmes. Il faut faire attention particulièrement quand on a plusieurs site sur un même serveur.

Fichier.php

**<?php**session\_start();  
  
**if**(**isset**($\_SESSION['username']))**{  
 echo** $\_SESSION['username'];  
**}**$\_SESSION['username']="apocalypse";

Fichier2.php

**<?php**session\_start();  
  
var\_dump($\_SESSION);

La session ‘username’ sera défini dans l’appel du Fichier2.php. Il faut également noter que les session sont généralement valable pendant 30 minutes dans le fichier php.ini. Deuxième remarque importante il faut placer session\_start() avant la manipulation de sessions sinon ça ne marchera tout simplement pas.

Exemples de session avec les étudiants :

**<?php**session\_start();  
header('Content-Type: text/html; charset=UTF8');  
  
**if**(!**isset**($\_SESSION['etudiants']))**{** //on défini la session en tant que tableau  
 //dans le cas ou elle n'existe pas  
 $\_SESSION['etudiants']=**array**();  
**}  
  
if**(!**empty**($\_POST))**{** $\_SESSION['etudiants'][]=$\_POST;  
**}  
  
if**(!**empty**($\_SESSION['etudiants']))**{** $students = $\_SESSION['etudiants'];  
**}  
?>** <html>  
 <head>  
 <title>Enregistrement Etudiants</title>  
 <link rel="stylesheet" type="text/css" href="styles.css">  
 </head>  
 <body>  
 <h1>Formulaire enregistrement étudiants</h1>  
 <h2>Liste d'étudiants</h2>  
 **<?php if** (!**empty**($students)): **?>** <table>  
 <thead>  
 <th>Prénom</th>  
 <th>Nom</th>  
 <th>Age</th>  
 <th>Email</th>  
 <th>Langue</th>  
 </thead>  
 <tbody>  
 **<?php foreach**($students **as** $s): **?>** <tr>  
 <td>**<?=**$s['prenom']**?>**</td>  
 <td>**<?=**$s['nom']**?>**</td>  
 <td>**<?=**$s['age']**?>**</td>  
 <td>**<?=**$s['email']**?>**</td>  
 <td>**<?=**$s['langue']**?>**</td>  
 </tr>  
 **<?php endforeach**; **?>** </tbody>  
 </table>  
 **<?php else**: **?>** <p>Il n'y a pas d'étudiants enregistrés</p>  
 **<?php endif**; **?>** <h1>Formulaire</h1>  
 <form method="post" action="students.php">  
 <pre>  
 prenom <input type="text" name="prenom" />  
 nom <input type="text" name="nom" />  
 age <input type="text" name="age" />  
 email <input type="text" name="email" />  
 langue <input type="text" name="langue" />  
 <input type="submit" name="envoyer" />  
 </pre>  
 </form>  
 </body>  
 </html>

**$\_COOKIES**

Les cookies servent à enregistrer des informations dans l’ordinateur de l’utilisateur pour être utilisé plus tard. Elle ont une durée de vie limitée définie comme suit :

**<?php**//date actuelle plus délai plus  
//delai d'expiration  
// date actuelle plus deux mois  
$expir=time() + 2\*30\*24\*3600;  
//défini le cookie  
setcookie("fond",12,$expir);  
  
**if**(**isset**($\_COOKIE['fond']))**{** var\_dump($\_COOKIE['fond']);  
**}**

Exemple d’utilisation des cookies, Color challenge :

**<?php  
if**(!**empty**($\_POST))**{** $fond=$\_POST['fond'];  
 $texte=$\_POST['texte'];  
 $expir=time() + 2\*30\*24\*3600;  
 setcookie("fond",$fond,$expir);  
 setcookie("texte",$texte,$expir);  
**}else if**(**isset**($\_COOKIE['fond']) && **isset**($\_COOKIE['texte']) )  
**{** $fond=$\_COOKIE['fond'];  
 $texte=$\_COOKIE['texte'];  
**}  
?>**<!DOCTYPE html>  
<html>  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  
 <title>Couleurs du site</title>  
 <style type="text/css" >  
 <!--  
 body**{**background-color: **<?php echo** $fond **?>** ; color: **<?php echo** $texte **?>** ;**}** legend**{**font-weight:bold;font-family:cursive;**}** label**{**font-weight:bold;font-style:italic;**}** -->  
 </style>  
</head>  
<body>  
<form method="post" action="color\_challenge.php">  
 <fieldset>  
 <legend>Choisissez vos couleurs</legend>  
 <label>Couleur de fond  
 <input type="text" name="fond" />  
 </label><br /><br />  
 <label>Couleur de texte  
 <input type="text" name="texte" />  
 </label><br />  
 <input type="submit" value="Envoyer" />&nbsp;&nbsp;  
 <input type="reset" value="Effacer" />  
 </fieldset>  
</form>  
<p>Essayer avec font #597cff et #cdfdff et couleur de fond #cdfdff</p>  
</body>  
</html>

Les cookies ne peuvent pas enregistrer des paramètres compliqués comme des objets et des tableaux contrairement aux sessions. Une astuce pour pouvoir le faire et de transformer ces tableaux ou objets en les transformant en chaînes de caractères avec l’aire de la fonction serialize(). Pour les lires ensuite il va falloir les reconvertir en tableau ou objet avec unseralize()

**<?php**//serialize transforme les array et objets  
// en chaînes de caract+ères  
$var = ['Alfonso','Paul'];  
$var = serialize($var);  
$expir=time() + 2\*30\*24\*3600;  
//défini le cookie  
setcookie("fond",$var,$expir);

// reconvertir les paramètres pour les utiliser  
**if**(**isset**($\_COOKIE['fond']))**{** var\_dump($\_COOKIE['fond']);  
 $var2 = unserialize($\_COOKIE['fond']);  
 var\_dump($var2);  
**}**

**Uploader un fichier.**

PHP va automatiquement enregistrer le fichier « uploadé » de façon temporaire. Si rien n’est fait avec ce fichier il sera détruit à la fin du code.

<form enctype="multipart/form-data" action="uploader.php" method="POST">  
 Choississez un fichier: <input name="uploadedfile" type="file" /><br />  
 <input type="submit" value="Upload File" />  
</form>

Le formulaire renverra à un service “uploader.php”

*/\*\*  
 \* Array element Contents  
 \* $\_FILES['file']['name'] nom  
 \* $\_FILES['file']['type'] contenu peut être:  
 \* application/pdf image/gif multipart/form-data text/xml  
 \* application/zip image/jpeg text/css video/mpeg  
 \* audio/mpeg image/png text/html video/mp4  
 \* audio/x-wav image/tiff text/plain video/quicktime  
 \*  
 \* $\_FILES['file']['size'] taille en bytes  
 \* $\_FILES['file']['tmp\_name'] nom temporaire du fichier  
 \* $\_FILES['file']['error'] Code erreur s'il y en a  
 \*/*$target\_path = "uploads/";  
  
$target\_path = $target\_path . basename( $\_FILES['uploadedfile']['name']);  
  
**if**(move\_uploaded\_file($\_FILES['uploadedfile']['tmp\_name'], $target\_path)) **{  
 echo** "le fichier ". basename( $\_FILES['uploadedfile']['name']).  
 " a été téléchargé";  
**} else {  
 echo** "Une erreur est survenu pendant le processus, réessayez plus tard!";  
**}**

Exemple plus élaboré dans lequel on contrôle le type et la taille :

**<?php**

// Testons si le fichier a bien été envoyé et s'il n'y a pas d'erreur  
**if** (**isset**($\_FILES['uploadedfile']) **AND** $\_FILES['uploadedfile']['error'] == 0)  
**{** // Testons si le fichier n'est pas trop gros  
 **if** ($\_FILES['uploadedfile']['size'] <= 1000000)  
 **{** // Testons si l'extension est autorisée  
 $infosfichier = pathinfo($\_FILES['uploadedfile']['name']);  
  
 $extension\_upload = $infosfichier['extension'];  
 $extensions\_autorisees = **array**('jpg', 'jpeg', 'gif', 'png');  
 **if** (in\_array($extension\_upload, $extensions\_autorisees))  
 **{** // On peut valider le fichier et le stocker définitivement  
 $newName = hash('sha1',$\_FILES['uploadedfile']['name']).'.'.$extension\_upload;  
 move\_uploaded\_file($\_FILES['uploadedfile']['tmp\_name'], 'uploads/' . basename($newName));  
 **echo** "Transfert du fichier complété !";  
 **}  
 }else{  
 echo** 'Erreur fichier trop gros';  
 **}  
}else{** $erreur = $\_FILES['uploadedfile']['error'];  
 **echo** "Le transfert du fichier a subis une erreur de code $erreur";

**Manipulation de fichiers**

Il est pratique d’opérer avec des fichiers dans le système que ce soit pour créer des fichier log pour établir des rapports d’erreurs etc… fichiers log.

Exemple 1 créer un fichier

**<?php** // testfile.php  
*/\*\* Correspond aux options d'écriture/lecture du fichier  
'r' Read from file start Open for reading only; place the file pointer at the beginning of the file.  
'r+' Read from file start and allow writing  
'w' Write from file start and truncate file  
'w+' Write from file start, truncate file, and allow reading if the file doesn’t already exist.  
Open for reading and writing; place the file pointer at the beginning of the file. Return  
FALSE if the file doesn’t already exist.  
Open for writing only; place the file pointer at the beginning of the file and truncate the  
file to zero length. If the file doesn’t exist, attempt to create it.  
Open for reading and writing; place the file pointer at the beginning of the file and  
truncate the file to zero length. If the file doesn’t exist, attempt to create it.  
'a' Append to file end Open for writing only; place the file pointer at the end of the file. If the file doesn’t exist,  
attempt to create it.  
'a+' Append to file end and allow reading  
Open for reading and writing; place the file pointer at the end of the file. If the file  
doesn’t exist, attempt to create it.  
 \*/*$fh = fopen("testfile.txt", 'w') **or die**("Failed to create file");  
$text = <<<\_WRITE  
Line 1  
Line 2  
Line 3  
\_WRITE;  
fwrite($fh, $text) **or die**("Could not write to file");  
fclose($fh);  
**echo** "File 'testfile.txt' written successfully";

Exemple 2:

Lire les fichiers

**<?php** // copyfile2.php  
$fh = fopen("testfile.txt", 'r') **or  
die**("File does not exist or you lack permission to open it");  
//filesize donne la taille du fichier  
//donc la longueur de lecture  
$text = fread($fh, filesize("testfile.txt"));  
fclose($fh);  
  
**echo** $text;

Exemple 3:

Copier les fichiers

**<?php**copy('testfile.txt', 'testfile2.txt') **or die**("Could not copy file");  
**echo** "File successfully copied to 'testfile2.txt'";

Exemple 4:

Bouger / renommer un fichier (le même processus) :

**<?php**//Generalement ces fonctions retourne un booléen  
  
**if** (!rename('testfile.txt', 'testfile3.new'))  
 **echo** "Could not rename file";  
**else echo** "File successfully renamed to 'testfile3.new'";

Exemple 5:

Effacer un fichier

**<?php**//Effacer les fichiers avec la fonction unlink  
**if** (!unlink('testfile2.new')) **echo** "Could not delete file";  
**else echo** "File 'testfile2.new' successfully deleted";

Exemple 6:

Mettre à jours un fichier

**<?php**//Modifier un fichier  
$fh = fopen("testfile.txt", 'r+') **or die**("Failed to open file");  
$text = fgets($fh);  
// zero indique le déplacement du pointeur  
//après la fin du fichier qui est donné par  
//SEEK\_END  
fseek($fh, 0, *SEEK\_END*);  
fwrite($fh, "$text") **or die**("Could not write to file");  
fclose($fh);  
**echo** "File 'testfile.txt' successfully updated";

Exemple 7 :

Le même que précédemment mais avec une logique de verrouillage. Une action utilisateur pourra mettre à jours le fichier mais pas en même temps ce qui pourrait provoquer des conflits.

**<?php**$fh = fopen("testfile.txt", 'r+') **or die**("Failed to open file");  
$text = fgets($fh);  
**if** (flock($fh, *LOCK\_EX*)) **{** fseek($fh, 0, *SEEK\_END*);  
 fwrite($fh, "$text") **or die**("Could not write to file");  
 flock($fh, *LOCK\_UN*);  
**}**fclose($fh);  
**echo** "File 'testfile.txt' successfully updated";

Exemple 8:

Etant donnée le fichier:

Testfile.txt qui contient:

jake@gmail.com 52163

On peut écrire le code suivant pour avoir une comparaison avec les données de ce fichiers

**<?php**$fh = fopen("testfile.txt", 'r') **or  
die**("File does not exist or you lack permission to open it");  
$line = fgets($fh);  
fclose($fh);  
//explode lit $line le separe chaque qu'il  
//y a un espace et transforme chaque morceau  
//en tableau  
$user = explode(" ",$line);  
  
**if**($user[1] === '52163')**{  
 echo** 'le password est correct';  
**}else{  
 echo** 'le password est incorrect';  
**}**var\_dump($user);  
**die**();

Exemple 9

Lire un fichier entier

**<?php  
echo** "<pre>"; // Enables display of line feeds  
**echo** file\_get\_contents("testfile.txt");  
**echo** "</pre>"; // Terminates pre tag  
  
//echo file\_get\_contents("http://oreilly.com");

Marche même avec les sites. Essayer de dé commenter la dernière ligne.

**Execution de lignes de commandes du système**

On peut executer des lignes du commande du système sous-jacent à PHP.

**<?php** // exec.php  
$cmd = "dir /s"; // Windows  
// $cmd = "ls"; // Linux, Unix & Mac  
exec(escapeshellcmd($cmd), $output, $status); //si status est zero alors tout est ok  
**if** ($status)**{  
echo** "Exec command failed";  
**} else {  
 echo** "<pre>";  
 **foreach** ($output **as** $line) **echo** htmlspecialchars("$line\n");  
 **echo** "</pre>";  
**}**

Affichera les détails des dossiers et fichiers système.

**Introduction aux expressions régulières**

**<?php***/\*\*  
 \* Les regex peuvent être présenté entre # ou entre /  
 \* /vladimir/ et #vladimir# sont la même chose  
 \*  
 \* L'option i signifie que les majuscule sont ignorées  
 \*  
 \*/***if** (preg\_match("#Vladimir#+", "Vladimir Kramnik")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
*/\*\*  
 \* Le point "." sert de joker pour n'importe quel caractère  
 \*/***if** (preg\_match("#Vladim.r#", "Vladimar Kramnik")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
*/\*\*  
 \* Le point match n'importe qu'elle caractère  
 \* L'astérisque '\*' match de 0 ou supérieur nombre de fois  
 \*/***if** (preg\_match("#<h.\*>#", "<h1>This is a title</h1>")):  
 **echo** 'trouve tag';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
*/\*\*  
 \* /<.+>/  
 \* Le point match n'importe qu'elle caractère  
 \* Le plus '+' match de 1 ou supérieur nombre de fois  
 \*/***if** (preg\_match("#<.+>#", "<>This is a title")):  
 **echo** 'trouve tag';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
*/\*\*  
 \* #a?#  
 \* 0 ou 1  
 \*/***if** (preg\_match("#a?#", "<h1>This is a title")):  
 **echo** 'trouve a';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
*/\*\*  
 \* Matcher les méthacaractères  
 \*  
 \* Le point match n'importe qu'elle caractère  
 \* Le plus match de 1 ou supérieur nombre de fois  
 \*/***if** (preg\_match("/8\.1/", "Windows 8.1")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;

Partie 2

**<?php***/\*\*  
 \* Les paranthèses signifient traiter ces caractères comme un groupe  
 \*/***if** (preg\_match("#1(,000)+#", "1,000,000,000,000")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
*/\*\*  
 \* conditionnel ou  
 \* avec le symbole pipe |  
 \*/***if** (preg\_match("#chateau|gateau#", "je prefere le gateaux")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
*/\*\*  
 \* Expression commençant par  
 \* ^  
 \* Expression finissant par  
 \* $  
 \*/***if** (preg\_match("#u$#", "je prefere le chateau")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
*/\*\*  
 \* Classes de la form [ae]  
 \* Soit a soit e  
 \*/***if** (preg\_match("#ch[aeiou]let#", "chalet")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
*/\*\*  
 \* Negation dans la classe  
 \* avec ^ également attention!!  
 \* [^at]  
 \*   
 \*/***if** (preg\_match("#[^t]{4}#", "Chateau")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
*/\*\*  
 \* Indiquer une fourchette/range  
 \* [0-9]  
 \* equivalent de /\d/  
 \*   
 \*/***if** (preg\_match("#^[A-Z]#", "Pas de chiffre")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
  
  
*/\*\*  
 \* entre crochet {} représente le nombre de fois qu'un pattern doit être répété  
 \*/***if** (preg\_match("#r{4}#", "Superrr")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;

Part 3

**<?php***/\*\*  
 \* List d'exo  
 \*/  
  
/\*\*  
 \* /^[A-Z]/ -->  
 \* Chaînes de caractère qui commence par une majuscule  
 \*/  
  
/\*\*  
 \* carte de crédit (version simplifiée)  
 \* 9999-9999-9999-9999  
 \*  
 \*/*$reg = "#[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4}#";  
**if** (preg\_match($reg, "9152-5214-5124-5124")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
  
*/\*\*  
 \* Typiquement nous avons  
 \* pour matcher un username /^[a-z0-9\_-]{3,16}$/  
 \*/***if** (preg\_match("/^[A-Za-z0-9\_-]{8,16}$/", "123456")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;  
  
*/\*\*  
 \* Email address  
 \*/***if** (preg\_match("#^[a-z0-9\_-]+@[a-z0-9\_-]+\.[a-z0-9\_-]{2,}$#", "afernandezocampo@gmail.com")):  
 **echo** 'trouve';  
**else**:  
 **echo** 'ne trouve pas';  
**endif**;