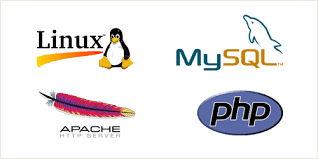
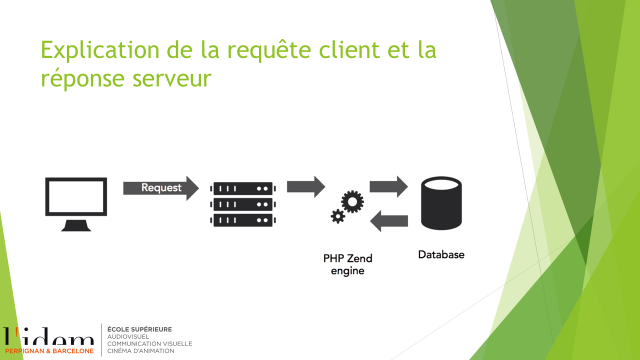
**Memo recap semaine 1**

* **Les applications web sont conçues avec une architecture multi-tier.**
* **Client: le navigateur Web (Firefox, Opera, Safari, Chrome)**
  + Reproduit du html/css et exécute du code javascript. Ainsi que des plug-ins (JAVA, Flash Player ou Silver Light)
* **Web, Business et Data:**
  + Web: HTTP server (Apache) reçoit la requête client et retourne une réponse.
  + Business: application server (PHP, python, Ruby, C#)
  + Data: server base de données (MySQL, SQL Server, SQLlite,Oracle…).
* **LAMP Stack**

****

**Comment marche une requête http ?**

****

L’utilisateur fait une requête HTTP (The Hypertext Transfer Protocol) via un navigateur généralement(ça peut être une autre application aussi) à une serveur pour qu’une page précise lui soit délivrée.

Si la page est statique –Que le html et du css,voir du javascript- la page lui sera délivrée. Dans le cas contraire apache renvoie au moteur PHP pour qu’il génére la page. Ce moteur peut ou non faire des requête en base de données.

**Conséquence du Schéma HTTP request (Le schéma précédent)**

* PHP est essentiellement un langage serveur et par conséquent les navigateurs ne le comprennent pas.
* Double cliquer sur un fichier php ne marchera donc pas car il n’y a pas de requête serveur effectuée.
* Assurez vous que votre serveur Apache fonctionne sinon nous aurons le même résultat/problème que précédemment.
* Toujours accéder au fichier à travers une URL. Quand on travail en local parfois on accède au fichier via link local (« C:\wamp\...\cheminDuFichier.php ») ; dans ce cas on fait lire le navigateur directement et ne comprendra pas le code php.

1. **Configuration de PHP**

Nous avons vu que php peut se configure avec le fichier php.ini sur le système. Chaque fois qu’il change tout les services doivent redémarrer pour être pris en compte.

Pour démarrer il est important de configurer

|  |  |  |
| --- | --- | --- |
| display\_errors | On | On |
| error\_reporting | 32767 | 32767 |

Il faut également configurer le fuseau horaire si ce n’est pas fait dans php.ini pour savoir quel php.ini est chargé (oui des fois il peut y en avoir plusieurs sur le système). Lisez la ligne

|  |  |
| --- | --- |
| Loaded Configuration File | C:\wamp\bin\apache\apache2.4.9\bin\php.ini |

[Date]

; Defines the default timezone used by the date functions

; http://php.net/date.timezone

date.timezone = Europe/Paris

Dans le cas où vous ne pouvez pas définir le fuseau horaire car vous n’avez pas accès au fichier php.ini.

Vous pouvez créer un fichier .htaccess dans le dossier racine du projet et faire les configurations nécessaires.

# Save this file as .htaccess (with a leading dot) in the server  
# document root folder on an Apache server.  
# Configuration directives that set a value are preceded by php\_value.  
# Directives that turn a setting on or off are preceded by php\_flag.  
  
**php\_value** error\_reporting 32767  
**php\_flag** display\_errors on  
  
# Time zone setting for US West Coast  
**php\_value** date.timezone America/Los\_Angeles

Et dans le cas ou le fichier .htaccess n’est pas supporté par la configuration initial il est possible de définir des options dans le code lui-même.

ini\_set('error\_reporting', *E\_ALL*);  
ini\_set('display\_errors', '1');  
  
date\_default\_timezone\_set('America/Los\_Angeles');  
  
phpinfo();

Sachez que parfois (pour les serveurs partagés) que certaines options sont non modifiables.

1. **Les variables**

* Information stockés temporairement en mémoire vive.
* On peut les concevoir comme des boites dans lesquelles ont stocks des informations.
* On utilise des variables tout les jours sans se rendre compte:
  + - * Nom, qui varie selon la personne.
      * Total a payer dans une addition.
      * La variable reste la même mais les valeurs changent
* La règle: en php les variables sont définies par le signe $ et le nom de la variable donc « $nomVariable »

**Bonnes pratiques pour définir les variables**

* Utilisation camelCase ou l'underscore.
* Choisir des nom qui ont un sens. Ni trop long ni trop court.
* Les variables ne sont vu que par les développeurs.
* N’utilisez pas les noms de variables réserver à des variables super globales.
* N’utilisez pas « $this » qui est réservé pour la programmation orienté objet.
* Ne commencez pas par un chiffre.
* Les variables sont sensibles au majuscule.

**Les types de variables**

* String - Chaînes de caractères
* Int - (Integer) Nombres entiers
* float - Nombres décimaux
* Bool - (boolean) booléen
* NULL - Néant
* Array() - des tableaux

**Exemples :**

**<?php**//String  
$Prenom = 'Jean';  
$Nom = 'Dubois';  
// (int)  
$numeroGagnant = 33;  
  
//float  
$decimal = 4.3;  
  
//On peut définir les tableau / array de différentes formes  
//array() avec crochet  
$tab = ['Marcel','Jean','Maude'];  
//array associatif avec crochet  
$assoc = ['Nom'=>'Dubois','Prenom'=>'Jean'];  
$multi = [  
 ['Texte'=>'mots'],  
 ['Chat'=>'Chien'],  
 ['Volvo','BMW'],  
];  
  
//avec le mot explicit array  
$voitures=**array**("Volvo","BMW","Toyota");  
  
$multi2 = **array**(  
 **array**('Texte','mots'),  
 **array**('Chat','Chien'),  
 **array**('Volvo','BMW'), //on peut laisser la virgule sur la dernière ligne contrairement à d'autres langage de programmation  
);  
  
//on peut finalement créer un tableau de la façon suivante  
$tab3[0] = 'Je suis une valeur du tableau tab3';  
  
*/\*\* AFFICHER LES VARIABLES \*\*/*// echo $Prenom //afiche une variable  
  
var\_dump($multi); //affiche une variable et son type et détails les tableau  
//print\_r($tab); //affiche également le type sous une forme différente.  
//die('stop'); //fonction pour "tuer le script"  
  
*/\*\* ACCEDER A UNE VALEUR DANS UN TABLEAU \*\*/*//La valeur dans $tab[0] est 'Marcel'  
//var\_dump($tab[0]);

Exemple 1 bis :

Ici on ne spécifie pas l’index mais ils se créent tous seuls.

**<?php**//$characters = array('Arthur Dent', 'Ford Prefect', 'Zaphod Beeblebrox');  
$characters = ['Arthur Dent', 'Ford Prefect', 'Zaphod Beeblebrox'];  
//echo $characters;  
$characters[] = 'Marvin';  
$characters[] = 'Slartibartfast';  
  
  
//print\_r($characters);  
  
**echo** $characters[1];

Pour les chaînes de caractères on utilise soit les guimets soit les apostrophes mais ce qui peut avoir des implications différentes.

Exemple 2 :

**<?php**//$livre = 'L\'aube du troisième millénaire';  
//$livre = "L'aube du troisième millénaire";  
//$livre = '"L\'aube du troisième millénaire"';  
$livre = "\"L'aube du troisième millénaire\"";  
  
  
//echo 'I love $livre';  
**echo** "I love $livre";

Exemple 2 bis :

**<?php**$descriptions = [  
 'Terre' => 'sans danger',   
];  
  
$descriptions['Zod'] = 'Commandant force spaciales';  
  
//echo $descriptions['Zod'];  
  
//print\_r($descriptions);  
  
**echo** "La terre est connue pour être **{**$descriptions['Earth']**}**";

Pour concaténer les variables :

Exemple 3

**<?php**$firstName = 'Austin';  
$lastName = 'Powers';  
$title = '"L\'aube du nouveau millénaire"';  
$author = 'Douglas Adams';  
$answer = 42;  
  
//$fullName = $firstName . $lastName;  
*/\*\* Si on veut un espace entre les variables \*/*//$fullName = $firstName . ' ' . $lastName;  
*/\*\* ou alors simplement \*/*$fullName = "$firstName $lastName";  
$book = "$title by $author";  
  
//echo $fullName . '<br>';  
//echo $book;  
  
$newLines = "<br>"; //Nouvelle ligne html qui est lu par le navigateur  
$newLines = "\r\n\r\n"; //Est lu par le code source mais pas le nagivateur... elle sont bien là mais le navigateur ne les reconnait pas  
$message = "Name: $fullName $newLines";  
$message .= "Book: $book $newLines";  
$message .= "Answer: $answer";  
**echo** $message;

Heredoc est une méthode qui nous permet d’insérer un ligne de caractère longue sans à avoir à échapper les caractères spéciaux :

Exemple 4 :

**<?php**header('Content-Type: text/css;charset=utf8'); //permet d'afficher les accent et autres caractères dans le navigateur  
  
$title = "L'aube du troisième millenaire";  
$author = 'Douglas Adams';  
$android = 'Marvin';  
$intelligence = 'intélligence incroyable';  
  
//heredoc est sensible à tout les caractères, on a pas besoin d'échapper les guimets ou autres caractères spéciaux  
$heredoc = <<< EOT  
Dans "$title" de $author, $android "L'android paranoiaque" rouspète parce qu'il fait des calculs trop difficile, meme s'il a une "$intelligence"  
 Même tab est reconnu  
EOT;  
  
**echo** $heredoc;

Nous avons vu que nous pouvons inclure et imbriquer des fichiers php entre eux.

Grâce à la déclaration « include » :

**<?php include**("footer.php"); **?>**

On peut le faire aussi grace à la déclaration « require »

**<?php require** ('footer.php'); **?>**

La différence est que « require » exige que le fichier exige sinon il y a une erreur qui s’affiche

Les deux variantes suivantes sont semblables aux deux précédentes mais n’inclus el fichier qu’une seule fois même si les déclarations sont répétées.

**<?php require\_once**('footer.php'); **?>**

**<?php include\_once**('footer.php'); **?>**

1. **Les opérateurs de calculs**

Exemple 5:

**<?php**// addition opérateur  
$num1 = 20;  
$result = 10;  
  
  
  
//$result = $num1 + $result;  
// est la même chose que  
//$result += $num1;  
  
//$result = $num1 \* $result;  
// est la même chose que  
// $result \*= $num1;  
  
//$result = $num1 / $result;  
// est la même chose que  
// $result /= $num1;  
  
//$result = $num1 - $result;  
// est la même chose que  
// $result -= $num1;  
  
var\_dump($result);

1. **Les opérateurs logiques**

() *paranthèses*

== *égal à*!= *différent de*<= *inférieur ou égal*>= *supérieur ou égal*< *inférieur à*> *supérieur à*=== *strictement supérieur à*, *vériefie egalement le type de la donnée*!== *strictement différent de*, *vériefie egalement le type de la donnée*

&& *et (cumul les condition sur un même if)*

|| *et (exclusif, c’est une condition ou l’autre)*

and *equivalent de &&*

or *equivalent de ||*

xor *exclusif – oui ça existe aussi en php*

1. **Structure d’une déclaration if /else if/ else**

Exemple 6 :

$name = '';

**if**($name == 'Alfonso')**{  
 echo** 'Ce sera le cours sur le PHP';  
**}else if**($name =='Paul')**{  
 echo** 'Ce sera le cours sur du Javascript';  
**}else{  
 echo** 'Il n\'y aura pas cours';  
**}**

Exemple 7:

**<?php**$name = 'Alfonso';  
$day = 'Jeudi';  
$tache = '';  
  
**if** ($name == 'Alfonso' && $day == 'Jeudi') **{  
 echo** 'Le vendredi il n\'y aura pas cours les djeuns';  
**} elseif** ($name == 'Paul' || $tache == 'rien' ) **{  
 echo** "Alfonso ne viendra pas c'est Paul qui fait cours";  
**} else {  
 echo** 'Ça doit être le weekend';  
**}**

Les if peuvent être écris d’une autre façon (notez les deux point à la fin de chaque déclaration, le endif à la fin et l’absence de crochet :

Exemple 8:

**<?php**$name = 'Alfonso';  
$day = 'Jeudi';  
$tache = '';  
  
**if** ($name == 'Alfonso' && $day == 'Jeudi') **:  
 echo** 'Le vendredi il n\'y aura pas cours les djeuns';  
**elseif** ($name == 'Paul' || $tache == 'rien' ) **:  
 echo** "Alfonso ne viendra pas c'est Paul qui fait cours";  
**else :  
 echo** 'Ça doit être le weekend';  
**endif ;**

1. **Exemple d’une déclaration switch :**

Équivalent d’une déclaration if /else if/ else mais plus pratique si on a multiple cas de figure pour une même variable/condition – Utile quand on fait du templating html:

Exemple 9 :

**<?php**$name = 'Brull';  
  
**switch** ($name) **{  
 case** 'Manuel':  
 **echo** 'J\'adore le SQL';  
 **break**;  
 **case** 'Aurore':  
 **case** 'Isaac':  
 **case** 'Carolina':  
 **echo** "2 semaines de PHP";  
 **break**;  
 **default**:  
 **echo** 'Pas de résultats';  
**}**

1. **Opérateur ternaire :**

Équivalent d’une déclaration if /else mais elles sont un raccourcies.

Exemple 10 :

**<?php**$reponse = 14;  
  
$resultat = ($reponse == 42) ? "La reponse est $reponse" : 'En cours de calcul';  
  
**echo** $resultat;

Exemple 11 :

**<?php**$cout\_unitaire = **false**;  
  
//opérateur qui marche comme in if simple  
$prix\_entier = $cout\_unitaire ?: 25;  
  
**echo** $prix\_entier;

1. **Isset:**

Isset() vérifie qu’une variable est définie et non null.

Exemple 12 :

**<?php**//$prix = NULL;  
  
**if** (**isset**($prix) && $prix==**true**) **{  
  
} else {** $prix = 25;  
**}  
  
echo** $prix;

1. **Incrémentateur:**

Exemple 13 :

**<?php**$number = 5;  
$number++; //augmente le nombre de 1

// $number--; diminue le nombre de 1  
**echo** $number . '<br>';  
++$number;  
**echo** $number . '<br>';  
$result = $number++ \* 2;  
**echo** '$result is ' . $result . '<br>';  
**echo** $number . '<br>';  
$result = ++$number \* 2;  
**echo** '$result is ' . $result . '<br>';  
**echo** $number . '<br>';

1. **Les boules (loop):**

Pratique pour effectuer des taches répétitives. Il y a 4 déclarations de loop : DO, WHILE, FOR et FOREACH

1. DO : Quoi qu’il arrive le code dans la boucle sera exécutée si et seulement si la condition est rempli (vraie) alors la boucle sera réitérée.

**<?php**$i = 15;  
  
**do {** $i++;  
 **echo** $i . '<br>';  
**} while** ($i < 10);

1. WHILE : Ici la boucle est lancée seulement si la condition est vraie.

**<?php**$i = 15;  
  
**while** ($i < 10) **{** // compteur  
 **echo** $i . '<br>';  
 $i++;  
**}**

1. FOR: Boucle qui sert à rétérer un nombre déjà connu d’itération :

**<?php**$characters = ['Michael Jordan', 'Kobe Bryant', 'Lebron James', 'Allen Iverson', 'Shaquille Oneal'];  
$num\_items = count($characters); //pour avoir le nombre d’itération  
//for ($i = 0; $i < $num\_items; $i += 2) {  
**for** ($i = $num\_items - 1; $i >= 0; $i--) **{  
 echo** "<li>$characters[$i]</li>";  
**}  
?>**

1. FOREACH: Boucle pour un nombre défini d’itération mais on ne sait pas le nombre. La subtilité celui-ci c’est qu’il peut parcourir des tableau et certains objets naturellement. Il prend comme référence l’index directement :

**<?php**$characters = ['Michael Jordan', 'Kobe Bryant', 'Lebron James', 'Allen Iverson', 'Shaquille Oneal'];

**foreach** ($characters **as** $name) **{  
 echo** "<li>$name</li>"; //affichera les nom  
**}**

Le foreach peut également afficher les index alias les clés dans un array associatif ou numérique.. au choix :

**<?php**$cel = [  
 'footballeur' => 'Messi',  
 'politique' => 'Obama',  
 'chanteur' => 'Bruno Mars'  
];

**foreach** ($cel **as** $key => $value) **{  
 echo** "<p>$value est $key.</p>"; //affichera les clés et les noms  
**}**

les mot clés **continue** et **break. Continue** est utilisées pour passer à l’itération suivante sans finir le reste des opérations s’il y en a. Le mot clé **break** casse littéralement la boucle.

**<?php**$i = 0;  
  
**while** ($i < 10) **{** $i++;  
 **echo** $i . '<br>';  
 **if** ($i == 6) **{  
 break**;  
 **}** // if ($i % 2) {  
// continue;  
// }  
**}**

Concernant les boucles d’autres syntaxe sont possibles. Pour FOR, WHILE et FOREACH. Ceci n’est pas applicable au DO.

**<?php**$characters = ['Michael Jordan', 'Kobe Bryant', 'Lebron James', 'Allen Iverson', 'Shaquille Oneal'];  
$num\_items = count($characters);  
**for** ($i = $num\_items - 1; $i >= 0; $i--) :  
 **echo** "<li>$characters[$i]</li>";  
**endfor**;  
**?>**

**<?php**$characters = ['Michael Jordan', 'Kobe Bryant', 'Lebron James', 'Allen Iverson', 'Shaquille Oneal'];  
**foreach** ($characters **as** $name) :  
 **echo** "<li>$name</li>";  
**endforeach**;  
**?>**

**<?php**// initialize counter  
$i = 0;  
  
**while** ($i < 10) :  
 // increment counter  
 $i++;  
 **echo** $i . '<br>';  
**endwhile**;

1. **Les fonctions:**

Une fonction est une expression ou relation mathématique qui lie deux ou plusieurs paramètres. En programmation c’est essentiellement la même chose sauf que la définition est orientée différemment. Une fonction est un bloque organisé et de code réutilisable qui performe une ou plusieurs opération.

A - En PHP elles sont définie par le nom et les parenthèses à la fin du nom.

**<?php**$version = phpversion();

Il existe deux types de fonctions natives à PHP – elles sont définies au sein du code PHP et on peut les appeler directement. Comme phpversion() précédemment vu.

L’autre type de fonction sont celle que nous définissons nous même pour effectuer les tâches que précisément nous voulons. Par exemple :

**<?php**

//modulo emulateur  
**function** EmulateurModulo($a,$b)**{** $r = $a/$b;  
 //la fonction floor prend l'entier  
 //et exclus les décimaux  
 $round = floor($r);  
 $res = $r-$round;  
 **return** $res\*$b;  
**}**

Le mot clé **return** indique que la fonction retourne un résultat. Toutes les fonctions ne retournent pas forcément un résultat. Elle peuvent contrôler des valeurs ou faire d’autres opérations.

B- On peut passer des paramètres aux fonctions. Soit par valeur soit par référence

**<?php**//On passe par référence on ne passe  
//pas la valeur au sein de la fonction  
// mais son addresse dans la mémoire vive  
$var = 3;  
  
**function** testRef(&$var)**{** $var\*=2;  
**}**testRef($var);//converti $var en 6  
var\_dump($var);//6  
//en temps normal nous avons ceci, voyez l’exemple 2  
$var2 = 4;  
  
**function** testFunc($var)**{  
 return** $var\*=5;  
**}**$res = testFunc($var2); // 20  
var\_dump($res);//20  
var\_dump($var2);//4  
  
//modulo emulateur  
**function** EmulateurModulo($a,$b)**{** $r = $a/$b;  
 //la fonction floor prend l'entier  
 //et exclus les décimaux  
 $round = floor($r);  
 $res = $r-$round;  
 **return** $res\*$b;  
**}**$resultat = EmulateurModulo(5, 3);  
var\_dump($resultat);

Certaines fonctions natives passe les variables par références. Exemple

**<?php**$class = [  
 ['first' => 'Mathieu', 'last' => 'Vie'],  
 ['first' => 'Lerognon', 'last' => 'Isaac'],  
 ['first' => 'Thomas', 'last' => 'Lerouge'],  
 ['first' => 'Lucas', 'last' => 'Hugo'],  
 ['first' => 'Damien', 'last' => 'Merle'],  
 ['first' => 'Maxime', 'last' => 'Paturel'],  
 ['first' => 'Aurore', 'last' => 'Regalado'],  
 ['first' => 'Marc', 'last' => 'Alsina'],  
 ['first' => 'Julien', 'last' => 'Alsina'],  
];  
  
usort($class, **function**($a, $b) **{  
 if** ([$a['last'], $a['first']] < [$b['last'], $b['first']]) **{  
 return** -1;  
 **} elseif** ([$a['last'], $a['first']] > [$b['last'], $b['first']]) **{  
 return** 1;  
 **} else {  
 return** 0;  
 **}  
}**);

Il n’y a pas de return pour cette fonction anonyme qui est appelé en callback. Mais le tableau est néanmoins trié.

C- Il est important d’avoir conscience que les dates peuvent prendre des formats différent.

**<?php**$noel = strtotime('Dec 25, 2016');  
//Unix time stamp  
//Nombre de secondes qui se sont écoulées  
//depuis 1 janiver 1970  
  
**echo** date('l, F j, Y', $noel);

//regardez la doc en http://php.net/manual/fr/function.date.php