Traitement coté serveur : initiation PHP

1. Introduction :

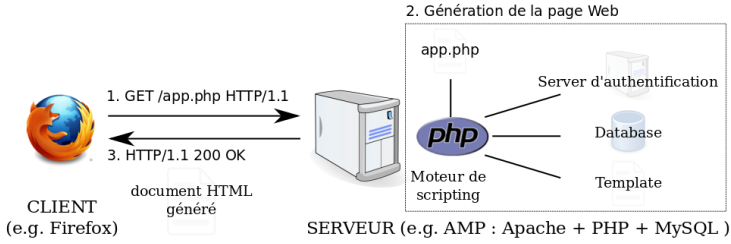
Jusqu'ici, nous n'avons étudié que des méthodes permettant de créer des pages web statiques ( pages affichant le même contenu à chaque visite). Nous allons maintenant nous intéresser à la conception de pages web dynamiques (pages dont le contenu change en fonction de l’utilisateur). Pour cela, il faut pouvoir afficher une page web qui sera fonction des interactions de l'utilisateur.

Ce traitement des actions de l'utilisateur peut s'effectuer :

* Coté client ( au niveau du navigateur) via JavaScript essentiellement
* Coté serveur via PHP, Java, Python (Django, Flask), Ruby, ASP.net, Nodejs…

Dans ce chapitre, nous allons étudier le traitement coté serveur :

* Le client (vous) envoie une requête http pour afficher une page web.
* Le serveur crée spécialement la page web pour le client (cela peut être en fonction des données envoyées par le client, en fonction de sa localisation, de la date ….)
* Puis le serveur envoie au client la page générée.



( source developpez.com, romain Le Breton)

**Remarques :**

* Bien que considéré par certains " un peu obsolète", le choix du langage s'est porté sur PHP car, ce langage nous a semblé plus simple à aborder. (Il s'agira d'une très modeste initiation essentiellement via des exemples).
* De plus, PHP (**PHP: Hypertext Preprocessor** bien que crée en 1994 ( par [Rasmus Lerdorf](https://fr.wikipedia.org/wiki/Rasmus_Lerdorf) ) est encore utilisé par environ 80% des sites web, et continue à être maintenu et amélioré.
* Cette année, il n'est pas demandé de maîtriser PHP, mais seulement de comprendre un code simple.

1. Installation serveur PHP

PHP est un langage coté serveur, il ne fonctionne donc que sur un serveur !

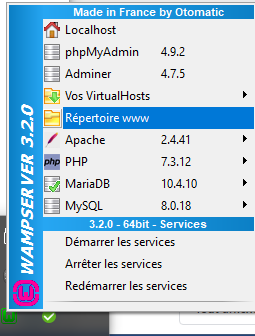
Si vous avez déjà publié un site web sur internet, vous avez déjà eu accès à un serveur web (Il suffirait de charger les pages PHP de test sur votre serveur via FTP). Mais si cela n'est pas le cas, pour faire quelques tests, il faut installer un serveur local. (WAMP, MAMP, XAMP…)

On vous propose de télécharger et installer XAMP : <https://www.apachefriends.org/fr/download.html> ou Wamp : <https://www.wampserver.com/>

Si besoin, tuto d'installation : <https://www.youtube.com/watch?v=3mP3wiz6vN4>

<https://espritweb.fr/installer-wamp-sur-windows-10-etape-par-etape-et-sans-erreur/>

1. Syntaxe PHP

Pour afficher une page PHP, il faut tout d'abord placer le fichier PHP sur votre serveur local. Avec WAMP, pour trouver où se trouve ce dossier, cliquer sur l'icône  située dans la barre systray ( en bas à droite), puis **Répertoire www**

* Copier le fichier **PHPexemple1.php** dans ce dossier. Ouvrir votre navigateur, puis rentrer l'URL correspondant à ce fichier :

http://localhost/PHPexemple1.php

**Remarques :**

Pour organiser un peu mieux votre serveur local, vous pouvez créer un dossier "**coursPHP**", puis placer les fichiers exemples (l'URL à rentrer devient

http://localhost/coursPHP/PHPexemple1.php)

* Actualiser la page. Pourquoi peut-on dire que c'est que c'est une page dynamique ?

* Ouvrir le fichier PHP avec votre éditeur (Sublime Text ou autre).

Comme vous pouvez le voir, un fichier PHP peut contenir du code HTML et bien sûr du code PHP. Examiner le code et répondre aux questions suivantes :

* Par quelles balises est délimité le code PHP ?
* Par quel symbole doit débuter toute variable PHP ?
* Quel est la fonction PHP permettant d'écrire du code HTML ?
* Quel symbole termine toute instruction PHP ?
* Quel symbole est utilisé pour placer une ligne de commentaire ?

1. "fonctionnalités" de bases

Après avoir copié le fichier **PHPexemple2.php** (ainsi que tous les autres fichiers exemples) dans le dossier cours PHP, l'ouvrir dans le navigateur et dans un éditeur puis répondre aux questions suivantes afin de trouver les syntaxes des boucles, tableaux… ( pour chaque réponse, vous copierez l'instruction correspondante issu du fichier exemple)

1. Tableau

Dans l'exemple, quelles instructions permettent de créer un tableau (au sens collection de valeur et non dessin de tableau HTML) ?

Fonction **Array()** → **$notes =array(12, 16, 19, 20, 17, 18);**

Ou écrire directement le tableau avec les éléments du tableaux entre crochets et séparés par des virgules (comme Python) → **$moyennes =[12, 16, 19, 20, 17, 18];**

Donner la fonction renvoyant la taille d'un tableau :

Pour le tableau **$moyennes**, quelle instruction permet d'accéder à son 4ème élément (en lecture ou en écriture d'ailleurs) ?

1. boucle for

La boucle FOR est utilisée pour répéter un nombre de fois défini.

La syntaxe pour la boucle for est la suivante :



Evolution de la variable de boucle

Entre chaque tour

Initialisation de

la variable de boucle

Condition de poursuite

de la boucle

En php, l'indentation n'est pas obligatoire pour rassembler les blocs (mais c'est conseillé).

* Entre quels symboles doit-on placer les expressions/instructions contenues dans la boucle for ?

1. Instruction conditionnelle

* Donner les mots clés permettant de coder des instructions conditionnelles :

* Comment sont délimitées les expressions à réaliser dans chacun des cas ?

1. Boucle while

Lorsque le nombre d'itération n'est pas connu à l'avance, on peut utiliser la boule while :

***while( condition )  
{  
bloc instruction(s);  
}***

**Rappel :** dans une boucle, il faut bien s'assurer que la condition sera fausse à un moment donné, sous peine de boucle infinie. (classiquement, on trouve avant la boucle l'initialisation de "la variable de condition de boucle". Et dans la boucle, on trouve au moins une instruction modifiant cette variable de boucle)

Exemple de code :

$depotInitial=10000;

$epargne=$depotInitial;

$n=0;

**while** ($epargne<2\*$depotInitial)

{

$n+=1;

$epargne=$epargne\*1.1; // intérêts de 10%

}

* Entourer l'initialisation de la variable de condition de boucle et l'instruction faisant évoluer cette variable dans la boucle
* Quelle information contient la variable $n après les passages dans la boucle while ?

1. Récupération des données transmises par GET ou POST
2. Tableau associatif

On peut transmettre des données à une page PHP via les méthodes GET ou POST. Dans les 2 cas, ces données sont stockées dans un type de variable appelé tableau associatif.

Tout comme un tableau classique, un tableau associatif permet de stocker dans une même variable plusieurs éléments (appelés valeurs). Mais ici, chaque élément du tableau associatif est identifié par une clé au lieu d'un numéro (indice).

Exemple :

$utilisateur=**array**('prenom'=>'Paul','nom'=>'Dupont','age'=>32);

Valeur associée

Variable

Tableau associatif

clé

Valeur associée

clé

Ou $utilisateur=['prenom'=>'Paul','nom'=>'Dupont','age'=>32];

Pour accéder à la valeur associée à la clé 'prenom' du tableau associatif $utilisateur, il suffit d'écrire :

$utilisateur['prenom']

Rq : L'accès peut se faire :

* en lecture : ex l'instruction **echo**("Bienvenue".$utilisateur['prenom']) écrira : Bienvenue Paul
* en écriture : $utilisateur['prenom']='Jacques' (on change la valeur associée à la clé, si la clé n'existe pas, elle est créée)

**Parcours de tableau associatif :**

Pour parcourir les clés et les valeurs d'un tableau associatif, il faut utiliser le mot clé **foreach** :

Exemple avec le tableau **$utilisateur**:

foreach($utilisateur as $key => $value )

{

echo (" une clé du tableau : $key ");

echo (" la valeur associée : ".$value."<br>" );

}

1. Récupération via GET

Rappel : Lors de l'envoi de données/paramètres via GET, ces données apparaissent dans l'URL.

Valeurs des paramètres

Exemple d'URL :

**http://localhost/coursPHP/bonjour.php?nom**=**Dupont**&**prenom**=**Marc**&**age**=**21**

Page PHP

? symbole de

séparation

Paramètres

Les valeurs sont alors stockées dans le tableau associatif **$\_GET**. Par exemple, ici, on récupère le prénom **Marc** par l'instruction : **$\_GET [prenom].**

* Ouvrir dans votre éditeur le fichier **bonjour.php.** Puis l'ouvrir dans votre navigateur en rentrant l'URL suivante (si besoin, l'adapter à votre arborescence)

[**http://localhost/coursPHP/bonjour.php?nom=Dupont&prenom=Marc&age=17**](http://localhost/coursPHP/bonjour.php?nom=Dupont&prenom=Marc&age=17)

* Faites plusieurs essais en modifiant les valeurs des paramètres GET transmis afin de comprendre le code PHP.
* Recharger la page, en donnant la valeur "**trente**" à **age** (chaine de caractère). Que se passe-t-il ?

**Conclusion :** PHP est un langage à typage faible, il convertit automatiquement les variables sous la forme qui lui parait adéquat …. quand il peut ! Il faut donc toujours vérifier la cohérence des données transmises avant traitement.

* Modifier le programme en ajoutant un test sur **age** grâce à la fonction **is\_int()** qui renvoie True si l'argument est un entier.

**Remarque :**

* Nous avons rentré ici "à la main" l'url avec les paramètres transmis en GET, mais l'url aurait pu être le résultat d'un page de login contenant un formulaire ( ex fichier login.html)

Exemple : Ouvrir dans le navigateur la page login : <http://localhost/coursPHP/login.html>

* Avec ce formulaire, dans le fichier PHP cible (bonjour.php), est-il encore nécessaire de tester si **$\_GET [age]** est bien un entier ? (justifier votre réponse)

1. Récupération via POST

Nous avons vu que si l'on veut éviter qu'un utilisateur "malveillant" puisse modifier les valeurs (directement dans l'url), il ne faut plus utiliser la méthode GET mais POST. La méthode de récupération des paramètres est exactement la même que précédemment, mais cette fois les données sont stockées dans le tableau associatif ayant pour nom :

**$\_POST .**

* Modifier les fichiers **login.html** et **bonjour.php** pour qu'ils fonctionnent avec la méthode transmission POST.