Bien démarrer en PHP

Sommaire

Les repères de lecture

Retour au chapitre
Définition
Objectif(s)
Espace Élèves
Vidéo / Audio
Point important / À retenir
Remarque(s)
Pour aller plus loin
Normes et lois
Quiz

Introduction

PHP repose sur une syntaxe précise et rigoureuse. Le premier travail à fournir est d'observer attentivement ces règles syntaxiques puis de les reproduire pas à pas. Au fur et à mesure de votre progression, ces règles deviendront des automatismes et nous pourrons ensuite continuer en nous appuyant sur ces bases.

I. Le PHP : un langage serveur

A. Qu'est-ce que PHP ?

PHP est un langage de script open source dont l'inventeur est Ramus Lerdorf.

Langage de script orienté serveur très proche du langage C, PHP permet de créer des applications ou des pages web dynamiques tout en étant plus simple et plus permissif que le JAVA ou autre langage Objet. C'est pour cette raison qu'il est très présent sur Internet, près de 80 % des sites internet utilisent le langage PHP.

PHP permet d'accomplir de nombreuses tâches et est très bien documenté (http://php.net), ce qui explique notamment l'intérêt qu'il suscite.

Le PHP propose de multiples fonctions utiles :
- accès à des fichiers et répertoires ;
- manipulation de chaînes de caractères ;
- des variables non typées à la volée ;
- à des pages web distantes et récupération de contenu ;
- analyse de flux ;
- calculs mathématiques ;
- formatage des dates et des heures ;
- aux bases de données.

Plusieurs versions de PHP existent qui ne permettent pas toujours un fonctionnement similaire du code. C'est la raison pour laquelle il est important de bien choisir la version installée ou de bien vérifier quelle version est déjà installée.

La dernière version est la version 7.0, mais vous pouvez commencer par la version 5.6 livrée dans la dernière version de WampServer.

B. Le modèle client/serveur

Nous avons vu que pour fonctionner, PHP nécessite d'être installé sur un serveur.

C'est pour cette raison qu'il est important de créer le code métier (calculs et règles de sécurité) côté serveur et donc en PHP plutôt qu'en JavaScript.

Dans notre cas, c'est WAMP qui s'est chargé de l'installer. Rappelons qu'un serveur est un ordinateur distant (ou local dans notre cas) qui héberge des fichiers et permet de les consulter en tant que pages web. Lorsque nous consultons une page web sur Internet, nous sollicitions de manière transparente le serveur qui héberge cette page. PHP est donc un langage interprété côté serveur. Il ne fonctionne pas directement dans le navigateur comme JavaScript, par exemple.

La relation client-serveur est détaillée dans le schéma ci-dessous.

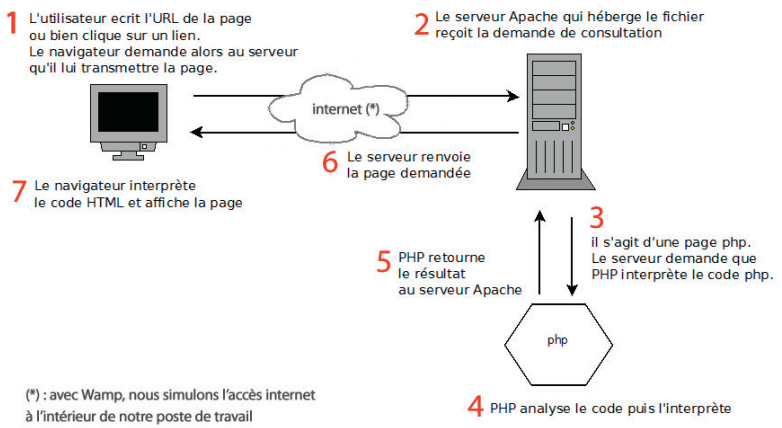

Fig. 1 La relation client-serveur

1. Explications détaillées du modèle client/serveur

a. Étape 1 : Requête du client vers le serveur

Le navigateur est appelé **le client**.

Il est utilisé par l'internaute (on dit l'utilisateur) pour afficher des pages web. Puisque ces pages possèdent une URL, l'utilisateur saisit cette URL dans la barre d'adresse du navigateur. Lorsque l'utilisateur valide l'URL, le navigateur envoie cette URL par Internet jusqu'au serveur (exemples : Apache, NGNIX, IIS) qui héberge la page. On dit qu'il effectue une **requête** (c'est-à-dire une demande).

Point technique complémentaire : c'est le protocole HTTP qui permet à la requête d'être transmise au serveur et à la réponse du serveur d'être transmise vers le client.

Cliquer sur un lien ou saisir une URL dans la barre d'adresse revient à effectuer la même requête.

b. Étape 2 : Requête du client vers le serveur Apache

Le serveur reçoit la requête de la part du client.

c. Étape 3 : Le serveur Apache sollicite l'interpréteur PHP

Le serveur constate que la page est un fichier PHP et demande donc au processus PHP d'analyser puis d'interpréter ce fichier (c'est-à-dire d'effectuer les instructions qui y sont écrites).

d. Étape 4 : L'interpréteur PHP interprète le code PHP

PHP commence par analyser la syntaxe du fichier. S'il constate une erreur, il retournera cette erreur. Si aucune erreur n'est constatée, PHP **interprète le code**. Cela signifie donc qu'il effectue les instructions contenues dans le fichier et qui ont été rédigées (ou codées) par le développeur. L'ensemble des instructions s'appelle le code ou le script.

e. Étape 5 : L'interpréteur retourne le code interprété au serveur Apache

Après avoir terminé l'interprétation du code, PHP retourne le résultat au serveur Apache.

f. Étape 6 : Le serveur Apache retourne le code interprété au client

Le serveur Apache envoie alors au client (le navigateur) le résultat fourni par PHP sous la forme d'une page HTML.

g. Étape 7 : Mise en forme HTML/CSS par le navigateur (rendu)

Le rôle du navigateur est d'afficher les pages web pour que l'utilisateur puisse les consulter correctement. Le navigateur va donc interpréter les éléments HTML et les instructions CSS contenus dans la page et effectuer les mises en forme correspondantes. La page est à ce moment consultable par l'utilisateur, ce qui marque la fin du processus client-serveur.

Nous constatons que ce processus, qui prend souvent moins de quelques secondes, permet de distinguer trois entités :
- le navigateur ;
- le serveur (qui contient l'interpréteur) ;
- le code PHP.

Ces trois entités fonctionnent entre elles dans un ordre précis qui est toujours le même. Il importe de bien le comprendre pour la suite de ce cours.

Par convention, le fichier index.php est le premier fichier qui est lu quand vous appelez le dossier. Par exemple, si vous avez nommé le fichier index.php dans le dossier essai, il suffira de se rendre à l'URL http://localhost/essai/ pour afficher notre code.

Pour écrire du PHP, il faut nécessairement un fichier de type texte que nous allons créer avec un éditeur de code. Le serveur Apache est paramétré (modifiable dans le fichier httpd.conf, mais non conseillé) pour reconnaître qu'un fichier est un fichier PHP s'il possède l'extension PHP.
Tous les fichiers PHP que nous serons amenés à créer posséderont donc l'extension PHP. Exemples : index.php, test.php, page.php, etc.

C. Les conventions de nommage du PHP

Respectez bien ces règles, en vous évitant des erreurs de fonctionnement, elles faciliteront votre apprentissage du PHP.

Concernant le nommage des variables, fonctions, méthodes, fichiers et dossiers :
- pas d'accents é, à, ô, etc. ;
- pas d'espace ;
- lettres et chiffres seulement ;
- les tirets ou les « underscores » sont permis ;
- nommage explicite (de préférence en anglais) ;
- pas d'abréviation.

Nous détaillerons d'autres conventions de nommage concernant les structures du PHP dans la suite du cours.

Concernant **le nom de fichier** : il est choisi librement. Il doit être le plus pertinent possible et respecter la convention de nommage établie ci-dessus. Pour un premier exemple de test nous pouvons créer un fichier que nous appellerons « index ».

Concernant **l'extension du fichier** : la plupart des fichiers possèdent une extension définie par un point suivi par des lettres. Par exemple, l'extension des fichiers Word est .doc, l'extension des fichiers HTML est .html. L'extension des fichiers PHP est .php.

Le nom complet de notre fichier sera donc : index.php. Les fichiers PHP étant des fichiers de type texte, nous avons besoin d'un simple logiciel d'édition de texte ou code.

Attention à l'encodage des fichiers, pour simplifier le portage de votre code entre Windows, Linux et Mac, préférez l'encodage UTF8.

Exemple non permis *:« présent Voiture.php »*

Exemple permis *:« presentation-voiture.php »*

II. Création d'un script PHP

A. Le fichier PHP

Lançons un logiciel d'édition de code tel que Atom, Sublime Text ou Notepad++.

Par défaut, ce type de logiciel propose :

– soit un fichier vide prêt à l'emploi dans lequel nous pouvons directement rédiger du code ;

– soit le fichier sur lequel nous travaillions la dernière fois. Dans ce cas, si nous voulons créer un nouveau fichier, il suffit dans la barre d'outils de cliquer sur « fichier » pour ouvrir le menu correspondant et ensuite de cliquer sur « Nouveau » pour obtenir un nouveau fichier.

Vous pouvez à l'avenir utilisez les « raccourcis clavier » pour gagner du temps. Le raccourci clavier pour ouvrir un nouveau fichier consiste à appuyer sur la touche **Control** ou **Cmd** du clavier selon le système d'exploitation puis la touche correspondant à la lettre *N* (à l'avenir nous dirons simplement **Control/Cmd + N).** Si vous n'êtes pas familier avec l'utilisation du clavier, exercez-vous car l'utilisation des raccourcis clavier est une aide appréciable dans le développement et représente un gain de temps réel.

Enregistrons immédiatement le nouveau fichier en le nommant donc *index.php*, le nom de fichier étant ici *index* et l'extension **.php**. Utilisons pour ce faire la commande « Enregistrer sous » du menu « Fichier ».

Enregistrons le fichier dans le répertoire *C:/wamp/www/test/* (ou celui dépendant de votre système d'exploitation).

Maintenant le fichier est prêt à être utilisé. Nous allons créer une simple ligne de code en PHP afin de vérifier son bon fonctionnement.

B. Les premières lignes de code

Le code PHP est interprété à condition d'être délimité par des balises PHP.

Les balises PHP agissent comme un signal d'ouverture et de fermeture. Elles indiquent au serveur que du PHP est présent et où commence le code et où il se termine.

La **balise PHP de début de code** s'écrit comme ceci : **< ?php**

Elle se compose :
- du chevron ouvrant : < disponible sur le clavier ;
- du point d'interrogation : ? ;
- Et des trois lettres : .php.

La balise PHP de fermeture de code s'écrit comme ceci : **?>**

Dans les dernières versions de PHP, il n'est plus nécessaire de fermer les balises en TOUTE FIN du fichier (si aucun HTML après), le serveur s'en charge cela permet d'éviter certaines erreurs d'interprétation (si le fichier contient uniquement du PHP).

Elle se compose :
- du point d'interrogation : ?
- du chevron fermant : > disponible sur le clavier (il faut utiliser la même touche que le chevron ouvrant en appuyant sur la touche *Shift* ou *Majuscule* du clavier).

Tout code PHP doit donc être écrit à l'intérieur des balises PHP **< ?php** et **?>** (sauf en fin de fichier).

Les balises PHP déterminent les blocs de code PHP. Lorsque le processeur PHP analyse un fichier, il cherche les balises d'ouverture et de fermeture, qui délimitent le code qu'il doit interpréter.

Le reste du code étant ignoré par l'interpréteur PHP, il est possible d'intégrer du HTML ou n'importe quelle ligne de texte dans un fichier PHP à partir du moment où ces lignes de texte sont placées **avant ou après les balises PHP**. Le code PHP devant, comme nous l'avons dit, être situé **à l'intérieur de ces balises**.

Exemple de contenu d'un fichier PHP :

```
1  <!doctype html>
2  <html lang="fr">
3      <head>
4          <meta charset="utf-8">
5          <title>Titre de la page</title>
6          <link rel="stylesheet" href="style.css">
7          <script src="script.js"></script>
8      </head>
9      <body>
10         <?php
11         // code php
12         ?>
13     </body>
14 </html>
```

Fig. 2 Exemple de fichier PHP contenant du HTML

Inclure du code HTML dans une chaîne de caractères n'est pas à utiliser en production, mais juste à des fins de test, sous peine de perdre en lisibilité et en maintenance du code.

a. Afficher des chaînes de caractères avec PHP

PHP est un langage de programmation qui permet d'effectuer de nombreuses actions. Dans le cadre de ce cours nous allons utiliser une fonction simple qui consiste à a cher des chaînes de caractères.

Les **chaînes de caractères** (*string* en anglais) sont littéralement une succession de caractères alphanumériques, c'est-à-dire du **texte**. Ce texte peut être court (un mot) ou long (plusieurs paragraphes).

Une chaîne de caractères peut également contenir du code HTML.

Notons enfin que les chaînes de caractères peuvent également contenir ou être exclusivement des nombres, mais dans ce cas ils ne sont plus considérés comme des nombres par PHP, et ils ne peuvent plus servir à effectuer des opérations mathématiques.

Pour afficher des chaînes de caractères en PHP, il faut utiliser une syntaxe précise que nous allons détailler ici en quatre étapes :

Étape 1 : En premier lieu, il est nécessaire d'utiliser une fonction d'affichage intitulée *echo*. La fonction *echo* est dédiée à l'affichage des chaînes de caractères.

Étape 2 : Ensuite, il est **obligatoire en PHP d'entourer les chaînes de caractères avec des** *single quotes* **ou** *double quotes* (signifiant respectivement apostrophes ou guillemets en anglais, et nous utiliserons ce terme car c'est celui qui est utilisé dans le monde du développement).

Étape 3 : Puis, à l'intérieur des balises PHP, nous utilisons *echo* pour procéder à l'affichage de notre phrase.

Étape 4 : Enfin, une ligne de code en PHP se termine par un point-virgule pour signaler la fin de ligne.

En résumé pour afficher la phrase Hello World! en PHP, le code complet est :

```
1  <?php
2  echo "Hello World!";
3  ?>
```

Fig.3 Exemple de Hello World

Une ligne de code est appelée **une instruction**. Après avoir créé cette ligne de code, nous souhaitons maintenant visualiser le résultat de ce code. Nous allons donc utiliser notre navigateur pour consulter la page que nous venons de créer.

C. Consultation des pages dans le navigateur

Rappelons-nous qu'une page web est en réalité générée par un fichier PHP. En installant Wamp, nous avons installé sur notre poste de travail un serveur web Apache qui nous permet de consulter en tant que pages web les fichiers que nous avons enregistrés dans un sous-répertoire de *www*. Ainsi le fichier *index.php* qui contient notre script PHP va pouvoir être consulté en tant que page web dans notre navigateur.

Comme vu précédemment, son URL sera http://localhost/test/index.php. Nous recommandons d'accéder à cette page en utilisant l'URL de base du serveur : http://localhost.

Cette URL est en effet la page d'accueil de Wamp qui recense l'ensemble des répertoires créés dans le répertoire www de Wamp. Il suffit de cliquer sur l'élément *localhost* du menu de Wamp qui est accessible via l'icône Wamp.

Depuis la page d'accueil de Wamp, nous pouvons cliquer sur le répertoire *test* nouvellement créé et afficher ainsi sous forme de liens cliquables les différents fichiers qui sont contenus dans ce répertoire, notamment le fichier *index.php*. Pour mémoire, le chapitre dédié à la présentation de Wamp détaille la méthode d'accès aux pages.

En cliquant sur *index.php*, nous ouvrons la page dans le navigateur et nous constatons que la phrase « Hello World! » est correctement affichée.

Nous avons donc produit notre première page web grâce à notre première ligne de code PHP.

```
1  <? php
2  echo 'la voiture est rouge';
```

Fig. 4

III. La syntaxe de base

A. Chaînes de caractères, analyse syntaxique et échappement

Suite à ce premier exemple, nous pouvons souhaiter afficher d'autres phrases. Par exemple nous voulons afficher ce texte simple : « PHP c'est sympa ! ».

Nous allons procéder de la même manière que précédemment mais dans ce cas, nous allons toutefois rencontrer un souci de syntaxe. En effet, le texte « PHP c'est sympa ! » contient une *single quote* entre la lettre « c » et le verbe « est ».

Hors nous avons vu qu'en PHP il était nécessaire de placer les chaînes de caractères entre *single ou double quotes*. Si nous ajoutons des *single quotes* en PHP et que la phrase contient elle-même une *single quote*, PHP va interpréter correctement la première *single quote* et il va considérer que la deuxième *single quote* (celle contenue dans la phrase) indique la fin de la chaîne de caractères. Ne comprenant pas que du texte soit encore présent après cette deuxième *single quote*, PHP va donc retourner une erreur.

Nous expliquerons les messages d'erreur par la suite. Gardons seulement à l'esprit que les erreurs ou les notices **doivent toujours être résolues.**

```
1  <?php
2  echo 'PHP c'est sympa'; // ce code génère une erreur
```

Fig. 5

B. L'analyse syntaxique ou *parsing*

En effet, la première chose que fait PHP est de vérifier que la syntaxe du code est correcte. Il **analyse donc la syntaxe** de tout le document (on dit qu'il **parse** le document) puis il exécute les instructions qui sont écrites.

En résumé, cela donne :
– PHP vérifie la syntaxe de tout le code ;
– PHP exécute l'instruction demandée.

Ici, la demande affiche la chaîne de caractères « PHP c'est sympa ! ». Ce principe de fonctionnement sera toujours le même tout au long de notre utilisation de PHP.

Dans notre exemple, la syntaxe n'est pas correcte, une erreur est retournée mais il existe cependant une solution simple. Il suffit de dire à PHP de ne pas tenir compte de la *quote* contenue dans la chaîne de caractères que nous souhaitons afficher.

Pour ce faire, nous allons utiliser le symbole antislash \ disponible sur le clavier (*Alt Gr* + touche \, la touche \ est celle du 8). Ce symbole *antislash* permet d'effectuer une action désignée sous le terme d'échappement, car elle permet qu'un caractère **échappe à l'analyse syntaxique**.

En mettant donc *l'antislash* devant la *quote* contenue dans la phrase, cette *quote* échappe à l'analyse syntaxique, on dit qu'elle est échappée. Ainsi, PHP ignorera la *quote* en question et la considérera comme faisant partie de la chaîne de caractères, ce qui est le but recherché.

La syntaxe correcte sera :

Dans un prochain cours, nous verrons que cette spécificité doit aussi être gérée pour les requêtes SQL.

```
1  <?php
2  echo 'PHP c\'est sympa';
```

Fig. 6

Il faut toujours échapper les *quotes* contenues dans les chaînes de caractères.

C. L'affichage avec le double quote

Il existe une autre manière d'entourer du texte en PHP qui repose sur l'utilisation des *double quotes* (guillemets doubles) comme ceci :

```php
<?php
echo "PHP c'est sympa";
```

Fig. 7

Les *double quotes* sont pratiques car elles nous dispensent d'échapper les *single quotes* d'une chaîne de caractères. En effet, l'analyse syntaxique ne va pas relever d'erreur puisqu'il n'y aura pas de conflit entre les *double quotes* et la *single quote* du texte, toute la phrase étant contenue dans *ces double quotes*.

D. Les commentaires en PHP

Les **commentaires** sont utilisés pour demander à PHP d'ignorer certaines lignes. Les lignes commentées ne sont donc pas analysées par PHP. Les commentaires sont très utilisés car ils nous permettent par exemple de rédiger une information complémentaire destinée à expliciter une ligne de code ou bien de noter à l'intention des développeurs des informations relatives au développement en cours.

Il existe trois façons de commenter du code :
- Commenter une ligne simple avec le *double slash* : //
- Commenter une ligne simple avec le *dièse* :#
- Commenter une ou plusieurs lignes avec *slash étoile* puis *étoile slash* : /* */

Notons pour conclure sur ce point que la plupart des éditeurs de code permettent de distinguer les commentaires du code PHP en les colorisant différemment.

```php
<?php
echo 'Salut'; // commenter une ligne
echo "Tu vas bien ?"; # commenter une autre ligne
echo 'PHP c\'est sympa'; /* commenter encore une autre ligne */

/*
Commenter
plusieurs
lignes
*/
```

Fig. 8

E. Résumé du cours

Ce chapitre nous a permis de voir les points suivants :
- comment utiliser les commentaires ;
- les différents types de commentaires ;
- l'affichage de texte avec la fonction écho ;
- l'échappement de caractères avec *antislash ;*
- l'utilisation des *singles* et *double quotes*.