Concevez votre site web avec PHP et MySQL

30 hours Medium License

Fonctionnement d'un site écrit en PHP

Nous y sommes presque ! Bientôt, nous aurons la joie infinie de laisser balader nos doigts pour écrire de belles lignes de code. Mais avant cela, il est important de comprendre comment un site web (ou application web, les deux se disent) fonctionne. Voyons ensemble les bases du fonctionnement d'un site en définissant les termes **serveur**, **client**, **site statique**, **site dynamique**. Nous parlerons inévitablement de PHP, de MySQL ainsi que d'Apache.

Si tout cela est nouveau, pas d'inquiétude ! Nous y allons, pas à pas, ensemble. Prenez le temps de relire plusieurs fois si cela est nécessaire, pour vous imprégner pleinement de ce nouvel univers qui s'offre à vous.

Qu'est-ce qu'un site web écrit en PHP offre de plus qu'un site écrit en HTML/CSS ?

Dans l'un des cours précédents du parcours, vous avez vu qu'il est possible de créer un site web avec HTML et CSS.

Si ce n'est pas le cas, il est impératif d'aller le lire avant de continuer avec PHP : Rendez-vous sur le cours Apprenez à créer votre site web avec HTML5 et CSS3.

Il est tout à fait possible de créer un site web avec ces langages ! En revanche, ceux-ci seront uniquement **statiques**. Avec PHP, nous ajoutons une envergure **dynamique**.

Quelle est la différence entre un site web statique et un site web dynamique ?

Les sites web statiques

Il s'agit de sites réalisés uniquement avec les langages HTML et CSS. Cela peut convenir pour certains besoins, en revanche, ils ne permettent pas une mise à jour programmée (c'est à dire, sans intervention humaine) des informations présentées.

Ainsi, le développeur écrit le code pour la présentation (en HTML) et la mise en page (en CSS), et cela sera toujours montré de la même manière, avec le même contenu aux internautes. Ces langages ne sont utiles que lorsqu'il s'agit de page web qui n'ont pas besoin de mises à jour régulières.

Un site du type Le Parisien (qui demande un rafraichissement très régulier, jusqu'à plusieurs fois par minutes) ne peut pas n'être développé qu'en HTML/CSS, sinon des développeurs devraient être en mesure de modifier le code, puis le mettre en production sans rien détériorer et tout cela très rapidement.

Retenez donc qu'une page/site web statique ne permet que de présenter des informations qui n'ont pas besoin d'être mises à jour très régulièrement.

Les sites web dynamiques

Pour apporter du dynamisme, donc la faculté de faire en sorte que votre site puisse afficher du contenu changeant sans l'intervention d'un développeur pour modifier le code, le **HTML et le CSS ne suffisent plus**. Bonne nouvelle, PHP vous le permet.

En fonction de la provenance des informations à afficher, vous aurez besoin de coupler PHP à d'autres technologies. Voici une petite liste (non exhaustive) de qu'il est possible de faire en matière de récupération de données avec PHP :

- *depuis un fichier* : il est possible de lire un fichier et réafficher les informations récupérées, mais aussi de modifier ce fichier, le supprimer…
- *depuis un autre site web* : avec PHP, il est possible de récupérer le contenu d'une page web et réafficher des informations et/ou les modifier avant de les réafficher…
- *depuis une base de données* : c'est ce que nous ferons ensemble dans les cours OpenClassrooms. Nous développerons un peu plus loin dans ce même cours. Un peu de patience .

Comme vous pouvez déjà le pressentir, il devient évident que pour développer des sites web aux fonctionnalités excitantes, il faut passer par PHP ! Voici quelques fonctionnalités que vous serez en mesure de réaliser :

- ***offrir un espace membre*** : vos visiteurs peuvent s'inscrire sur votre site et avoir accès à des sections qui leur sont réservées ;
- ***présenter des actualités*** : vous pouvez automatiser l'écriture d'actualités, en offrant à vos visiteurs la possibilité d'en rédiger, de les commenter, etc. ;
- ***acheter des produits*** : donnez la possibilité aux utilisateurs de remplir un panier d'achat, et le payer en ligne ;
- ***envoyer des newsletters*** : vous pouvez envoyer un email à tous vos membres régulièrement pour leur présenter les nouveautés et ainsi les inciter à revenir sur votre site.

Bien entendu, ce ne sont là que des exemples. Il est possible d'aller encore plus loin, tout dépend des besoins business de votre site web. Sachez par exemple que le site d'OpenClassrooms est développé en partie en PHP avec le framework Symfony !

Vous n'aurez pas à choisir entre un site web dynamique et statique. Le HTML/CSS cohabite très bien avec le PHP ! Dans un même script il est possible d'avoir du PHP et du HTML. C'est le cas lorsque nous n'utilisons pas de moteur de templating.
Par ailleurs, il n'est pas impossible qu'il y ait dans le même site web une page statique et une autre page qui elle est dynamique.

Le triplet PHP, MySQL & Apache

Rentrons dans le vif du sujet dès maintenant : pour faire fonctionner un site dynamique, un ordinateur a besoin d'un certain nombre de logiciels.

Serveur web : Apache

C'est l'élément essentiel pour qu'un site web écrit en PHP fonctionne. Quand nous disons "élément", c'est en réalité un logiciel. Un serveur web est un logiciel installé sur un ordinateur. Il en existe plusieurs, fonctionnant sur tout type de système d'exploitation en fonction de ce que nous souhaitons "servir" comme site web. Dans le cas d'un site web écrit en PHP, les logiciels les plus courants sont Apache et Nginx. **Tout au long de ce cours, nous utiliserons Apache**.

Comme son nom l'indique, un serveur "sert" un site web. Ainsi, son métier est simplement de trouver le code qu'il faut exécuter en fonction de ce qu'on lui donne comme information au moment de la demande effectuée par un

navigateur.

Pour nos développements nous avons deux choix : avoir un serveur web distant (qui n'est donc pas installé sur notre ordinateur), soit installer un serveur web sur notre ordinateur. Nous choisirons la deuxième option, c'est à dire, l'installation d'un serveur web sur notre ordinateur. Pour ce faire, nous verrons cela au prochain chapitre.

Sachez que depuis la version 5.4 de PHP, un serveur web est directement intégré avec PHP. C'est très pratique pour les développeurs ! Ils n'ont plus qu'à installer PHP et hop, lancer le serveur avec la commande `$ php -S 127.0.0.1:8000` et le site web est accessible sur leur ordinateur.

PHP

PHP est non seulement un langage, mais il existe également un moteur (appelé "Zend Engine") qui se charge d'interpréter le langage en un langage intelligible pour les navigateurs. En effet, à la différence du code HTML, les navigateurs ne sont pas capables de comprendre le PHP sans interprétation préalable. Il faut donc que ce moteur soit utilisé par le serveur web afin de pouvoir générer la page que le serveur web aura à renvoyer au navigateur qui a effectué la demande d'affichage du site.

Tout comme le serveur web, nous aurons à l'installer sur notre ordinateur afin de pouvoir développer en toute quiétude. Encore un peu de patience, nous verrons également cela au prochain chapitre.

MySQL

Nous parlons de présenter des informations lorsqu'il s'agit d'afficher une page web, mais d'où proviennent ces informations ? Elles peuvent venir de n'importe où ! Ceci dit, il est très courant d'utiliser ce que l'on appelle une base de données pour stocker et puiser des informations de toute sorte (une liste d'utilisateurs, des produits, des articles de blogs…). Cette fois-ci, ce n'est pas le serveur web qui fait appel à la base de données, mais le code du site web lui-même.

Dans votre code PHP, vous aurez donc à écrire toutes les instructions pour se connecter à la base de données, puis y écrire des informations et/ou récupérer des informations sous forme de ***requêtes sql***.

Pour stocker des informations, il existe de nombreuses autres solutions. MySQL n'est pas le seul type de base de données : vous entendrez sans doute parler de PostgreSQL ou encore MariaDB. Ces solutions sont tout aussi valables en fonction des choix techniques que l'équipe technique / l'entreprise aura choisis.

Récapitulons comment le triplet Apache, PHP et MySQL interagit ensemble

Je vous propose donc de voir ensemble les étapes qui se déroulent depuis le moment où un utilisateur tape une adresse de site web dans son navigateur (exemple : http://openclassrooms.com), jusqu'à l'affichage de la page web sur le navigateur de cet utilisateur.

Étape 1 : L'utilisateur tape l'adresse du site web qu'il souhaite afficher

Nous l'avons tous fait au moins une fois (sinon vous ne seriez pas là). Mais oui, vous savez ! Taper dans la barre d'adresse quelque chose comme "*http://…*" ou "https://…". À ce moment là, vous demandez à votre navigateur

d'afficher une page d'un site internet en particulier.

Étape 2 : La demande est formalisée par le navigateur

Une fois l'adresse du site web saisie dans votre navigateur préféré, celui-ci se charge de formaliser votre demande d'affichage de page en **requête HTTP** (nous reviendrons sur ce terme un peu plus tard lorsque nous parlerons plus en détail du *protocole HTTP*). Une fois cette requête formalisée, le navigateur l'envoie sur internet, pour contacter le serveur web qui héberge le code du site web que vous souhaitez afficher.

Il y a d'autres étapes qui se passent pour que le contact avec le bon serveur web se fasse, mais nous avons choisi de les omettre pour simplifier un peu les explications.

Étape 3 : La requête est reçue par le serveur web

La requête est donc reçue par le serveur web hébergeant le site web. Grâce à une configuration écrite sur le serveur web, celui-ci est capable de comprendre la requête reçue pour savoir ce qu'il doit demander.

Étape 4 : Le serveur web exécute le code du site web demandé

Le serveur se charge de demander à exécuter le script qu'il faut pour que le site web soit en mesure de générer la page web.

Étape 5 : Le code du site web génère la page web demandée

Ainsi, c'est le serveur web qui fait appel à PHP pour générer la page web demandée. Il se peut que l'application web fasse appel à une base de données (comme MySQL par exemple), dans ce cas, c'est le code du site web qui se charge de s'y connecter, et faire ce qu'il y a besoin de faire (exemple : aller chercher des données ou en enregistrer par exemple).

Étape 6 : La page générée est renvoyée au navigateur par le serveur web

Une fois que la page est générée par PHP dans un langage intelligible par un navigateur (le HTML par exemple), le serveur web renvoie l'ensemble de la page (du texte formalisé) sous forme de **réponse HTTP** (nous reviendrons sur ce terme un peu plus tard lorsque nous parlerons plus en détail du *protocole HTTP*).

Étape 7 : Le résultat de la page est reçu par le navigateur

La dernière étape est donc pour le navigateur d'afficher ce qu'il a reçu. Et voilà, vous voyez la page web que vous aviez demandé.

J'aimerais attirer votre attention sur le fait que de nombreux acteurs entrent en jeu lorsque l'on souhaite afficher une page web : entre le **navigateur**, le **serveur web**, le **code de votre application web**, les échanges faits avec la **base de données**, et surtout l'échange d'informations qui se fait sur **Internet**… La rapidité d'affichage d'une page web peut être impactée par tous ces acteurs. Gardez-le en tête car la performance est une problématique à aborder au plus tôt !

Le protocole HTTP

Nous avons eu un avant goût de l'ensemble des communications établies lorsqu'il s'agit d'afficher une page web. Deux termes très importants ont été évoqués plus haut : **requête** et **réponse HTTP**.

Je vous propose donc de voir (ou revoir) le principe du protocole HTTP qui est à la base même de tous ces échanges.

Lorsque vous vous baladez sur internet, vous passez votre temps à taper ce type d'adresse "*http://mon-site-prefere.com/*". Ce préfixe "*http://*" n'est pas anodin !

En effet, vous demandez à votre navigateur d'effectuer une requête en suivant un protocole.

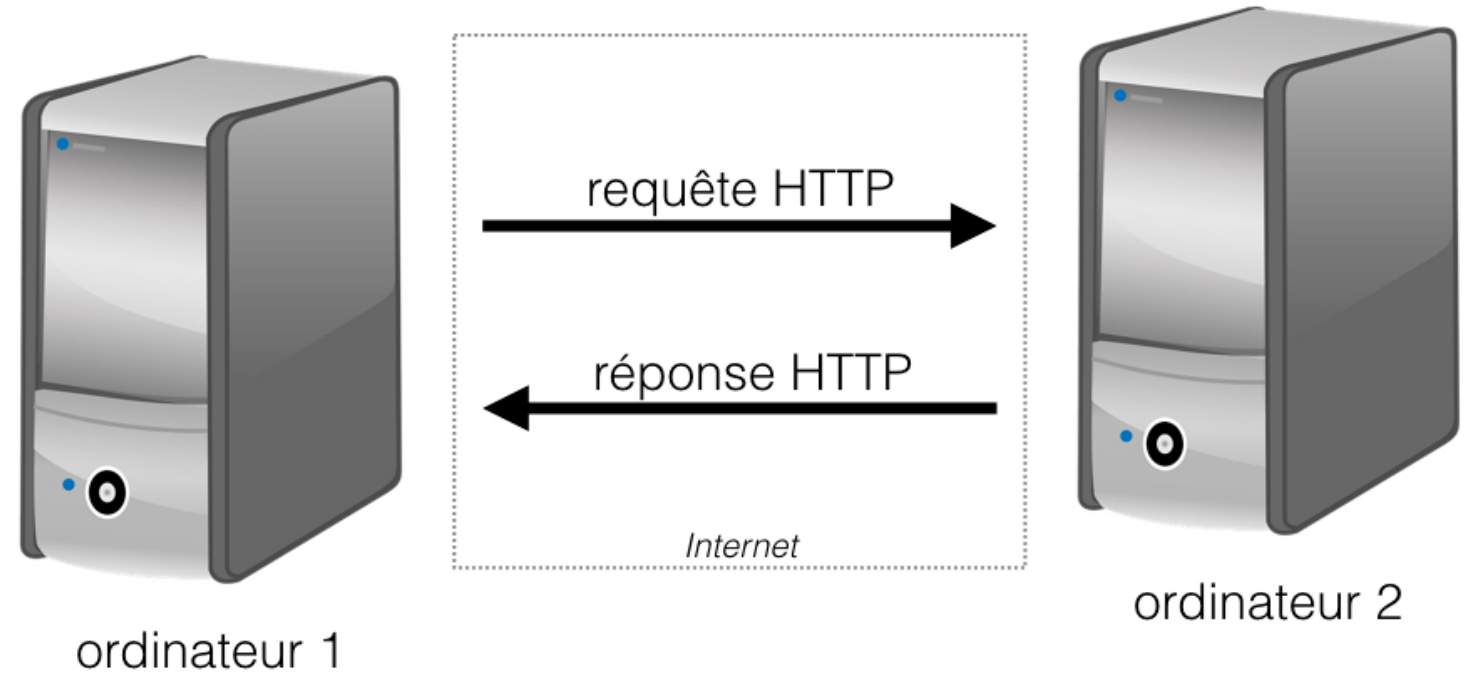
Définition de protocole d'après le Larousse : *Ensemble des règles établies en matière d'étiquette, d'honneurs, de préséances dans les cérémonies officielles*.

HTTP est un protocole permettant d'échanger des informations entre deux ordinateurs via Internet. Ce protocole est bien évidemment formalisé sous forme de document appelé "**RFC**" ("Request For Comment" en anglais, soit "Demande de commentaire").

Un consortium (ensemble de personnes prenant des décisions), le W3C, a écrit la RFC2616 afin d'écrire toutes les règles à suivre pour communiquer en HTTP. Cette RFC est désormais divisée en 7 RFC (de 7230 à 7237) pour en faciliter la maintenance et la lecture. Cela reste tout de même les documents les plus soporifiques que je connaisse  .

Vulgarisons un peu pour que nous nous raccrochions à quelque chose que nous connaissons bien : la communication par téléphone. La différence est que les interlocuteurs ne sont pas humains, mais des ordinateurs programmés pour communiquer ensemble sans l'intervention d'un humain. Pour pouvoir faire en sorte que leur communication se passe de manière "automatisée", il faut que les messages soient toujours formalisés de la même façon. C'est pour ça que l'on parle de **protocole**.

Les deux ordinateurs présentés ci-dessus peuvent être n'importe quels ordinateurs, pour l'instant, ce n'est pas important. Ce qui nous intéresse est la communication entre ces machines. La communication est toujours initiée par un ordinateur qui effectuera une requête HTTP ("*request*" en anglais) et une réponse HTTP ("*response*" en anglais) qui doit émaner de l'ordinateur à qui l'on fait la requête.



Communication via HTTP entre deux ordinateurs

HTTP n'est pas le seul protocole pour communiquer sur Internet. Il existe d'autres protocoles tels que AMQP, SSH ou encore FTP.

Ces deux types de messages sont formatés de telle manière que chacun des ordinateurs sait comment traiter ce message.

Décortiquons une requête HTTP

Le navigateur est en charge de générer la requête HTTP. Voici un exemple de requête HTTP lorsqu'un utilisateur tape l'adresse http://openclassrooms.com/cours dans la barre d'adresse du navigateur.

```
GET /cours HTTP/1.1
User-Agent: User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_2)
Host: openclassrooms.com
```

Exemple de requête HTTP

Prenons la première ligne de cette requête. Il faut distinguer trois informations :

1. `GET` ➞ Il s'agit de la méthode HTTP (pour connaitre la liste des méthodes HTTP possibles dans le protocole, rendez-vous sur la documentation officielle de la RFC) ;
2. `/cours` ➞ Il s'agit de la page que l'on souhaite récupérer ;
3. `HTTP/1.1` ➞ Il s'agit du protocole utilisé ainsi que la version qui a été utilisée pour générer cette requête (il existe plusieurs versions de HTTP - 1.0, 1.1 et 2.0 - *qui est encore expérimental*).

Les lignes qui suivent sont appelées en-têtes ("headers" en anglais) et servent à donner plus d'indications sur la manière dont on veut que la réponse soit formatée. Il peut en avoir bien plus, autant qu'on le souhaite.

Décortiquons maintenant une réponse HTTP

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2016 15:13:53 GMT
Server: Apache/2.2.14 (Win32)
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Exemple de réponse HTTP

A la différence de la requête, la réponse est formatée un peu différemment. Sur la première ligne nous avons :

1. `HTTP/1.1` ➞ le protocole utilisé et la version ;
2. `200` ➞ le code statut HTTP qui signifie que tout c'est bien passé - il en existe bien d'autres allant de 100 à 599 ;
3. `OK` ➞ le texte associé au code statut - en fonction du nombre, le texte changera.

Les lignes qui suivent (`Date` , `Server` , `Content-Length` , `Content-Type` , `Connection`) sont des en-têtes ("*headers*" en anglais) donnant des informations sur le contenu de la réponse HTTP.

Et nous avons enfin le contenu de la page, toujours précédé par un retour à la ligne, qui constitue la page que le navigateur aura à afficher. Dans l'exemple ci-dessus, nous aurons comme résultat ce qui suit :

Hello, World!

Résultat du corps de la réponse HTTP

Pour en savoir un peu plus sur les codes status HTTP, les resources, les requêtes… Je vous invite à faire un petit tour du côté du cours sur les API REST qui s'appuie sur le protocole HTTP.

Donc pour résumer, pour que deux ordinateurs communiquent entre eux via Internet, il faut simplement suivre des règles établies grâce au protocole HTTP.

À noter : Les règles établies par la RFC sont suivies (pas toujours à la lettre) par les personnes qui développent les navigateurs web et les serveurs web. **Il peut arriver que vous n'ayez pas exactement le même rendu d'un site web en fonction du navigateur utilisé** : en effet, les développeurs qui sont à l'origine de Google Chrome et Mozilla Firefox ne sont pas les mêmes ! La manière dont ils ont appliqués les règles imposés par la RFC varie. Il en est de même avec les serveurs web.
Gardez le en tête lorsque vous ferez vos tests après avoir fait vos développements .

Comme vous pouvez le voir, il ne s'agit que de texte échangé entre deux ordinateurs, souvent appelé **client** et **serveur**.

Qu'est-ce qu'un client ?

Lorsque vous voulez visiter un site web, vous devez taper son adresse dans votre navigateur web, que ce soit Mozilla Firefox, Google Chrome, Internet Explorer, Safari, Opera ou un autre. Nous avons vu qu'une communication devait être établie entre deux ordinateurs, et en particulier entre un **navigateur** et un **serveur web**.

Il faut savoir qu'Internet est un réseau composé d'ordinateur (d'ailleurs "web" est un mot anglais qui signifie "réseau"). Pour plus de simplicité, ces ordinateurs peuvent être classés en deux catégories : les clients et serveurs.

Attention : un ordinateur peut contenir un serveur ET un client. Ce même ordinateur peut contenir plusieurs clients (vous pouvez avoir plusieurs types de navigateurs, et plusieurs onglets ouverts).
Par ailleurs, un ordinateur, en particulier celui d'un développeur, peut avoir un ou plusieurs serveurs web installés.
Il ne faut donc pas confondre ordinateur, client et serveur.

Les clients

Il s'agit donc du logiciel capable d'émettre une requête HTTP. Celui dont nous parlons depuis le début de ce chapitre : **le navigateur**.

Il existe d'autre types de clients comme cURL qui est une interface en ligne de commande.

Chaque client représente une fenêtre du navigateur affichant une page web (donc trois onglets ouverts avec votre navigateur = trois clients). Pour plus de simplicité, vous verrez que la représentation d'un client est une image de ce type :

Représentation courante d'un client

Les serveurs

Nous en avons parlé précédemment : il s'agit d'un logiciel qui est en mesure prendre une requête et rendre une réponse. Le logiciel que nous allons utiliser s'appelle Apache. Nous verrons comment l'installer dans le chapitre qui suit.

Par abus de langage, l'on dit qu'un serveur est un ordinateur. Sachez qu'il peut y avoir sur un ordinateur plusieurs serveurs installés. Un ordinateur peut être dédié au fait d'être un serveur, c'est souvent le service que les entreprises d'hébergement de site offrent (comme OVH par exemple).

Pour plus de simplicité, vous verrez que la représentation d'un serveur est une image de ce type :

Représentation courante d'un serveur

La plupart du temps, un ordinateur dédié à une activité de serveur est dépourvu d'écran : il reste allumé et travail seul, sans intervention humaine, 24h/24, 7J/7. Il est possible de s'y connecter à distance, depuis un client pour monitorer l'activité de l'ordinateur par exemple.

Comment contacte-t-on un serveur ?

Eh bien vous le savez déjà ! Lorsque vous tapez dans la barre d'adresse "http://…", c'est comme si vous composiez un numéro de téléphone. Nous verrons dans le chapitre suivant comment contacter un serveur qui est installé sur son propre ordinateur.

Vous avez normalement maintenant une idée plus claire de ce qu'il se passe lorsque l'on souhaite afficher un site web développé en PHP.

Il est maintenant temps de préparer son ordinateur pour enfin se lancer dans le développement.

Votre rôle de développeur web PHP

Préparer son environnement de travail

The author

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Check out this course via

eBook Livre papier PDF Vidéo