# Séquence 2 : Interaction entre l'homme et la machine sur le web

## Activité 16 : Formulaire d'une page web

Tout d'abord, il faut vous connecter sur les machines avec l'identifiant du réseau local adm_s2i.

## 1. Version PHP

Comme déjà évoqué dans la partie consacrée au modèle client-serveur, un serveur Web (aussi appelé serveur HTTP) permet de répondre à une requête HTTP effectuée par un client (très souvent un navigateur Web). Nous allons travailler avec le serveur Web qui est installé sur votre ordinateur. Nous allons donc avoir une configuration un peu particulière puisque le client et le serveur vont se trouver sur la même machine. Cette configuration est classique lorsque l'on désire effectuer de simples tests. Nous aurons donc 2 logiciels sur le même ordinateur : le client (navigateur Web) et le serveur (serveur Web), ces 2 logiciels vont communiquer en utilisant le protocole HTTP. Le serveur Web que nous allons utiliser se nomme "Apache", c'est un des serveur Web le plus utilisé au monde. Apache a déjà été installé et configuré sur votre ordinateur à travers le logiciel wampserver, disponible sur le bureau.

Nous allons commencer par un cas très simple où le serveur va renvoyer au client une simple page HTML statique (ne pas hésiter à consulter la partie consacrée au modèle client-serveur pour plus de précision sur ce terme "statique"). Le logiciel a été configuré pour qu'il envoie vers le client une page HTML située dans un répertoire nommé "www_votre_nom" au sein du disque C:/wamp64/www, ce répertoire "www_votre_nom" devant se trouver à l'emplacement C:/wamp64/www.

### 1.1 À faire vous-même 1

Créez à l'emplacement C:/wamp64/www un répertoire nommé "www_votre_nom".

### 1.2 À faire vous-même 2

Créez un fichier "index.html", placez ce fichier "index.html" dans le répertoire "www_votre_nom" tout nouvellement créé.

### 1.3 À faire vous-même 3

Copiez le code HTML ci-dessous à l'aide de Notepad dans le fichier "index.html" que vous venez de créer.

```html
<!doctype html>
<html lang="fr">
        <head>
                <meta charset="utf-8">
                <title>Utilisation d'Apache</title>
        </head>
        <body>
                <p>Le serveur Apache fonctionne parfaitement</p>
        </body>
</html>
```

## 1.4 À faire vous-même 4

Lancer wampserver à l'aide du raccourci sur le bureau. Normalement un  doit s'afficher en bas à droite.

Ouvrez le navigateur web internet explorer (pas un autre sinon cela ne marchera pas !) et tapez dans la barre d'adresse "localhost".

Vous devriez voir votre page Web s'afficher.

Une petite explication s'impose à propos du "localhost" : comme nous l'avons déjà dit, notre serveur et notre client se trouvent sur la même machine, avec le "localhost", on indique au navigateur que le serveur Web se trouve sur le même ordinateur que lui (on parle de machine locale). Dans un cas normal, la barre d'adresse devrait être renseignée avec l'adresse du serveur Web.

Pour l'instant, comme déjà dit plus haut, notre site est statique : la page reste identique, quelles que soient les actions des visiteurs. Pour avoir un site dynamique, nous allons exécuter, côté serveur, un programme qui va créer de toute pièce une page HTML, cette page HTML sera ensuite envoyée au client par l'intermédiaire du serveur Web. Il existe différents langages de programmation qui permettent de générer des pages HTML à la volée : Python, Java, Ruby... Dans notre cas, nous allons utiliser le PHP (PHP est un acronyme récursif puisqu'il signifie : PHP Hypertext Preprocessor). Le PHP est un langage très utilisé même si dans le monde professionnel Java, Python, Ruby,... sont préférés au PHP. Il est très important de bien comprendre les processus mis en oeuvre :

- le client (le navigateur Web) envoie une requête HTTP vers un serveur Web
- en fonction de la requête reçue le serveur "fabrique" une page HTML grâce à l'exécution d'un programme écrit en PHP (ou en Python, Java...)
- le serveur Web envoie la page nouvellement créée au client
- une fois reçue, la page HTML est affichée dans le navigateur Web

Le langage PHP est déjà installé sur l'ordinateur par l'intermédiaire de wampserver que vous utilisez actuellement, nous allons donc pouvoir écrire notre premier programme en PHP. Notez qu'il n'est pas question ici d'apprendre à programmer en PHP, mais juste d'avoir un premier contact avec ce langage (et avec cette notion de Web dynamique)

## 1.5 À faire vous-même 5

Après avoir supprimé le fichier "index.html" préalablement créé dans le répertoire "www_votre_nom", créez un fichier "index.php", toujours dans le répertoire "www_votre_nom".

## 1.6 À faire vous-même 6

Copiez le code PHP ci-dessous dans le fichier "index.php" que vous venez de créer

```php
<?php
$heure = date("H:i");
echo '<h1>Bienvenue sur mon site</h1>
      <p>Il est '.$heure.'</p>';
?>
```

## 1.7 À faire vous-même 7

Ouvrez internet explorer et tapez dans la barre d'adresse "localhost".

Vous devriez avoir une page HTML qui vous donne l'heure, si vous actualisez la page, vous pourrez remarquer que bien évidemment l'heure évolue. Nous avons donc bien une page dynamique : le serveur PHP crée la page Web au moment où elle est demandée. À chaque fois que vous actualisez la page, la page HTML est générée de nouveau.

Vous aurez sans doute remarqué que l'extension ".html" a été remplacée par ".php". Au moment de la requête, le programme contenu dans ce fichier a été exécuté et la page HTML a été générée. Dans les 2 cas, le fichier se nomme "index", pourquoi ? Par défaut, le serveur prend en compte un fichier nommé "index" ("index.php" ou "index.html" selon les cas). Si vous voulez nommer votre fichier autrement, il faudra modifier ce que vous avez saisi dans la barre d'adresse de votre navigateur : si par exemple vous voulez nommer votre fichier "toto.html" (ou "toto.php"), il faudra saisir dans la barre d'adresse du navigateur "localhost/toto.html" (ou "localhost/toto.php").

Quelques remarques sur le programme PHP ci-dessus :

- "$heure = date("H:i");", "$heure" est une variable (en PHP les variables commencent par un "$"), cette variable "contient" une chaîne de caractères qui correspond à l'heure courante
- l'instruction "echo" permet d'afficher la chaîne de caractères qui suit l'instruction. Dans notre cas, la chaîne de caractères est "<h1>Bienvenue sur mon site</h1> Il est '.$heure.' " ce qui correspond à du HTML à l'exception de "$heure" qui permet d'afficher le contenu de la variable "$heure". La page Web générée contiendra le code HTML et le contenu de la variable "$heure". Le point "." est, en PHP, l'opérateur de concaténation (alors que par exemple en Python, l'opérateur de concaténation est le "+")

Si un client effectue une requête à 18h23, le serveur enverra au client le code HTML ci-dessous :

```html
<h1>Bienvenue sur mon site</h1>
<p>Il est 18h23</p>
```

Nous allons maintenant nous intéresser à la gestion des formulaires.

## 1.8 À faire vous-même 8

Après avoir supprimé tout le contenu de votre dossier "www_votre_nom", créez dans ce même dossier un fichier "index.html" et un fichier "trait_form.php"

## 1.9 À faire vous-même 9

Copiez le code HTML ci-dessous dans le fichier "index.html" que vous venez de créer :

```
<!doctype html>
<html lang="fr">
        <head>
                <meta charset="utf-8">
                <title>Le formulaire</title>
        </head>
        <body>
                <form action="trait_form.php" method="post">
                                <label>Nom</label> : <input type="text" nam
e="nom" />
                                <label>Prénom</label> : <input type="text"
name="prenom" />
                                <input type="submit" value="Envoyer" />
                </form>
        </body>
</html>
```

Copiez le code PHP ci-dessous dans le fichier "trait_form.php" que vous venez de créer :

```
<?php
    $n=$_POST['nom'];
    $p=$_POST['prenom'];
    echo "<p>Bonjour ".$p." ".$n.", j'espère que vous allez bien.</p>";
?>
```

## 1.10 À faire vous-même 10

Ouvrez internet explorer et tapez dans la barre d'adresse "localhost". Une fois la page Web affichée dans votre navigateur, remplissez le formulaire proposé et validez en cliquant sur le bouton "Envoyer"

Analysons le code HTML ci-dessus :

Nous utilisons la balise "form" afin de mettre en place un formulaire sur notre page :

```
<form action="trait_form.php" method="post">
```

Nous avons 2 attributs "action" et "method". L'attribut "action="trait_form.php"" indique que le client enverra une requête HTTP vers le serveur en cas de clic sur le bouton "Envoyer". Pour répondre à cette requête du client, le serveur devra exécuter le programme PHP contenu dans le fichier "trait_form.php". Le "method="post"" nous indique que la méthode utilisée pour effectuer cette requête HTTP est une méthode "POST" (cette notion de méthode a déjà été vu dans la partie consacrée au protocole HTTP)

Au niveau des 2 balises "input" permettant de saisir le nom et le prénom :

```
<label>Nom</label> : <input type="text" name="nom" />
<label>Prénom</label> : <input type="text" name="prenom" />
```

L'attribut "name" nous intéresse particulièrement dans ces 2 balises. Dans la première balise nous avons "name="nom"" et dans la deuxième balise, nous avons "name="prenom"". Nous aurons l'occasion de revenir sur le rôle de cet attribut "name" ci-dessous.

Intéressons-nous maintenant au code contenu dans le fichier "trait_form.php" :

La ligne

```
$n=$_POST['nom'];
```

permet d'attribuer à la variable "$n" la chaîne de caractères qui a été saisie par l'utilisateur dans le formulaire : balise "input" ayant l'attribut "name="nom"". Le "nom" de "$_POST['nom']" est directement lié au "nom" de l'attribut "name="nom"". Le principe est le même pour la variable "$p" : le "prenom" de "$_POST['prenom']" est directement lié au "prenom" de l'attribut "name="prenom"". Un ""$_POST['toto']"" contiendra la chaîne de caractères saisie par l'utilisateur dans le champ correspondant à une balise "input" possédant un attribut "name="toto"".

La dernière ligne :

```
echo "<p>Bonjour ".$p." ".$n.", j'espère que vous allez bien.</p>";
```

ne devrait pas vous poser de problème.

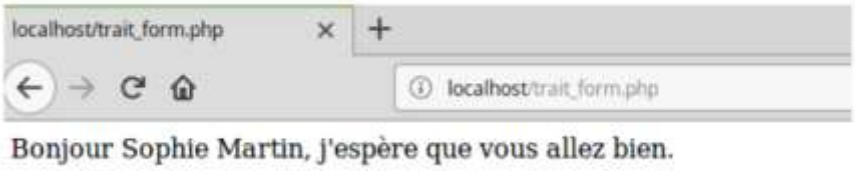
Nous avons vu qu'un clic sur le bouton "Envoyer" déclenche une requête HTTP vers le serveur et que les informations saisies dans le formulaire sont envoyées vers le serveur (puisqu'il est possible de récupérer ces informations au niveau du code PHP : une fois de plus, le code PHP est uniquement exécuté du côté serveur). Mais comment ces informations transitent-elles entre le client et le serveur ? La réponse est simple : cela dépend de la méthode utilisée par la requête HTTP.

Dans l'exemple ci-dessus, nous utilisons la méthode "POST" ("method="post"").

## 1.11 À faire vous-même 11

Ouvrez internet explorer et tapez dans la barre d'adresse "localhost". Une fois la page Web affichée dans votre navigateur, Saisissez "Sophie" pour le prénom et "Martin" pour le nom puis validez en cliquant sur

le bouton "Envoyer". Une fois que vous avez cliqué sur le bouton "Envoyer", observez attentivement la barre d'adresse de votre navigateur. Vous devriez obtenir quelque chose qui ressemble à cela :




Dans la barre d'adresse, rien de spécial à signaler, on constate que le fichier "trait_form.php" a bien été utilisé.

Il est possible d'utiliser une méthode HTTP "GET" à la place de la méthode "POST" :

## 1.12 À faire vous-même 12

Modifiez les fichiers "index.html" et "trait_form.php" comme suit :

index.html

```
<!doctype html>
<html lang="fr">
        <head>
                <meta charset="utf-8">
                <title>Le formulaire</title>
        </head>
        <body>
                <form action="trait_form.php" method="get">
                        <label>Nom</label> : <input type="t
ext" name="nom" />
                        <label>Prénom</label> : <input type
="text" name="prenom" />
                        <input type="submit" value="Envoye
r" />
                </form>
        </body>
</html>
```

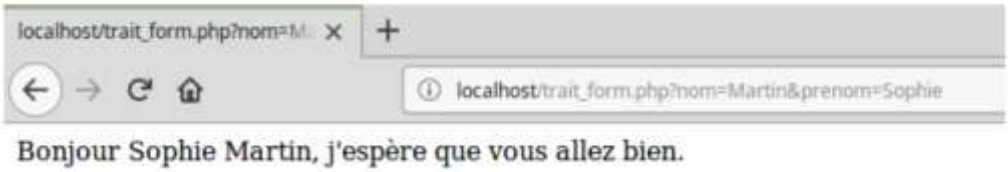
trait_form.php

```php
<?php
    $n=$_GET['nom'];
    $p=$_GET['prenom'];
    echo "<p>Bonjour ".$p." ".$n.", j'espère que vous allez bien.</
p>";
?>
```

Les différences sont minimes : dans le fichier "index.html" nous avons un "method="get"" à la place d'un "method="post"". Dans le fichier "trait_form.php" nous avons des "$_GET" à la place des "$_POST".

## 1.13 À faire vous-même 13

Ouvrez internet explorer et tapez dans la barre d'adresse "localhost". Une fois la page Web affichée dans votre navigateur, Saisissez "Sophie" pour le prénom et "Martin" pour le nom puis validez en cliquant sur le bouton "Envoyer". Une fois que vous avez cliqué sur le bouton "Envoyer", observez attentivement la barre d'adresse de votre navigateur. Vous devriez obtenir quelque chose qui ressemble à cela :




Vous avez du remarquer que cette fois-ci, les informations du formulaire sont transmises au serveur par l'intermédiaire de l'url : localhost/trait_form.php?nom=Martin&prenom=Sophie

---

Dans le cas de l'utilisation d'une méthode "POST" les données issues d'un formulaire sont envoyées au serveur sans être directement visibles, alors que dans le cas de l'utilisation d'une méthode "GET", les données sont visibles (et accessibles) puisqu'elles sont envoyées par l'intermédiaire de l'url.

Les données envoyées par l'intermédiaire d'une méthode "GET" peuvent être modifiées directement dans l'url :

## 1.14 À faire vous-même 14

Ouvrez votre navigateur Web et tapez dans la barre d'adresse "localhost". Une fois la page Web affichée dans votre navigateur, Saisissez "Sophie" pour le prénom et "Martin" pour le nom puis validez en cliquant sur le bouton "Envoyer". Une fois que le message "Bonjour Sophie Martin, j'espère que vous allez bien." apparait, modifier l'url : passez de "localhost/trait_form.php?nom=Martin&prenom=Sophie" à "localhost/trait_form.php?nom=Martin&prenom=Jean-Pierre", validez votre modification en appuyant sur la touche "Entrée".

Comme vous pouvez le constater, la page a bien été modifiée : "Bonjour Jean-Pierre Martin, j'espère que vous allez bien."

Même si dans notre cas cette opération de modification d'url est inoffensive, vous devez bien vous douter que dans des situations plus complexes, une telle modification pourrait entrainer des conséquences plus problématiques (piratage). Il faut donc éviter d'utiliser la méthode "GET" pour transmettre les données issues d'un formulaire vers un serveur.

Il est important de bien comprendre que la méthode "POST" n'offre pas non plus une sécurité absolue puisque toute personne ayant un bagage technique minimum sera capable de lire les données transmises à l'aide de la méthode "POST" en analysant la requête HTTP, même si ces données ne sont pas directement visibles dans l'url. Seule l'utilisation du protocole sécurisé HTTPS garantit un transfert sécurisé des données entre le client et le serveur (les données sont chiffrées et donc illisibles pour une personne ne possédant pas la clé de déchiffrement).

Bibliographie : Inspiré d'activités de David Roche