II - Création de formulaire

a) Utiliser un serveur local

Dans cette partie, nous aurons besoin d'un serveur. Nous allons nous contenter d'un serveur local installé sur la machine. L'ordinateur jouera alors à la fois le rôle de serveur (ce sera un serveur Apache, l'un des plus utilisés) et de client (par le navigateur).
Pour windows, UwAMP est très simple d'utilisation. Le plus pratique est d'utiliser la version portable : télécharger le fichier .zip et le décompresser.
Sous Linux, il existe un équivalent : LAMP.

Une fois le fichier .zip décompressé, se rendre dans le dossier UwAmp puis dans le sous-dossier www. Y créer un dossier nommé Initiation.

b) Première page dynamique

Télécharger le fichier index.html et le placer dans le dossier Initiation. Revenir à la racine du dossier UwAmp et cliquer sur UwAmp.exe qui va lancer le serveur local. On obtient l'affichage ci-contre :

Cliquer alors sur "www Site" puis, dans "Virtual Host", cliquer sur Initiation. Le serveur lance alors le fichier index.html.

Une autre façon de procéder consiste à lancer le fichier UwAmp.exe afin d'obtenir l'image ci-contre puis de cliquer sur le bouton Démarrer (ou Start).
Dans un navigateur, écrire l'URL `http://localhost/Initiation` où *localhost* est le nom de notre serveur local (il a toujours pour adresse IP 127.0.0.1). Le navigateur cherche alors le fichier index.html (ou index.php) présent dans le dossier Initiation. Si ce fichier n'existe pas, alors le serveur renvoie une erreur 404.

La page proposée est très simple (une seule balise <h1>), elle permet juste de créer un site statique. Pour le rendre dynamique, nous allons exécuter côté serveur un programme qui va créer une page HTML, propre à chaque client. Plusieurs langages peuvent être utilisés (Python, Java, Ruby, ...) mais dans notre cas, nous allons utiliser PHP (qui reste très utilisé y compris dans le monde professionnel).

Fermer la page web et arrêter le serveur : **appuyer sur le bouton "Arrêter" (ou "Stop") de UwAmp avant de fermer UwAmp** (ou laisser UwAmp ouvert pour savoir où en est votre serveur).

- Ouvrir Visual Studio Code (ou Notepad++, SublimeText, ...) et créer un fichier index.php dans le dossier Initiation. Y recopier le contenu du fichier index.html.
  Déplacer alors le fichier index.html (ne pas le supprimer car il servira plus tard).

- Dans le fichier index.php, remplacer le corps du document par :

```
<body>
   <h1>Premier formulaire</h1>
   <?php
   date_default_timezone_set('Europe/Paris');
   $heure=date("H:i:s");
   echo '<h2>Bienvenue sur mon site</h2>
   <p>Il est '.$heure.'.</p>';
   ?>
</body>
```

  Relancer le serveur puis dans le navigateur, saisir l'URL `http://localhost/Initiation` (qui chargera le fichier index.php).

  Une partie du code écrit dans ce fichier PHP est du code HTML, mais le code propre au PHP est situé entre les balises `<?php` et `?>`.

  - La première fonction (`date_default_timezone_set('Europe/Paris');`) n'a pas d'importance, elle sert juste à permettre une bonne utilisation de la fonction date.

- `$heure=date("H:i:s");` permet de définir la variable `$heure` (toutes les variables sont précédées d'un `$` en php) égale à la date au format heures, minutes, secondes (c'est donc l'heure et non la date qui sera déterminée dans notre exemple).
- `echo` est une instruction PHP fondamentale : elle permet d'afficher la chaîne de caractères qui suit l'instruction.
- La ligne suivante est du texte HTML (balise <h2>).
- Enfin, on écrit une phrase donnant l'heure. En PHP, la concaténation d'une chaîne de caractères avec une variable se faire à l'aide du point "." (en python, c'est le caractère "+").

Donc quand le client va effectuer la requête, le serveur va générer la page HTML avec l'heure qui aura été calculée au moment de la requête puis l'envoyer au client. Si on recharge la page, l'heure est actualisée (car une nouvelle requête est envoyée au serveur qui recalcule la page).

ATTENTION : En php, chaque ligne doit se terminer par un point-virgule (sauf si c'est après une accolade ouvrante ou si l'instruction est sur plusieurs lignes, comme pour l'instruction `echo` de l'exemple précédent).

c) Premier formulaire et méthode GET

Dans cette partie (et la suivante), nous allons travailler avec 2 fichiers :

- le fichier index.html qui sera la page d'accueil du site (donc replacer le fichier index.html dans le dossier Initiation). Ce fichier contiendra un formulaire de saisie ;
- un fichier php destiné à interpréter les saisies dans le formulaire et à générer la page HTML en fonction de ces saisies.

  Renommer le fichier index.php en traitForm.php.

1. **Conception du fichier index.html et donc du formulaire**

   Remplacer le corps du fichier (entre les balises <body>) par ce qui suit :

```
<body>
   <h1>Premier formulaire</h1>
   <form action="traitForm.php" method="get">
     Nom : <input type="text" name="nom" />
     Prénom : <input type="text" name="prenom" />
     <p>Vous pouvez ajouter un commentaire :
     <textarea name="commentaire" placeholder="Commentaire"></textarea></p>
     <button type="submit">Envoyer</button>
   </form>
</body>
```

   Remarque : Il est possible de mettre un peu plus en forme cette page, par exemple avec un fichier CSS. On pourrait aussi forcer les retours à la ligne en plaçant les lignes dans des balises de type bloc (<p>, , ...).

2. **Conception du fichier traitForm.php**

   Remplacer le code PHP du fichier traitForm.php par ce qui suit :

```
<?php
   date_default_timezone_set('Europe/Paris');
   $heure=date("H:i:s");
   $nom=$_GET["nom"];
   $prenom=$_GET["prenom"];
   $com=$_GET["commentaire"];
   echo '<h2>Bienvenue sur mon site</h2>
   <p>Bonjour '.$prenom.' '.$nom.', il est '.$heure.'.</p>';
   echo '<p>Voici ce que tu voulais dire:<br/>'.$com.'</p>';
?>
```

   Remarque : dans ce fichier php, il est également possible de définir un fichier CSS.
   J'ai mis deux fois l'instruction `echo` dans ce code, mais cela pouvait n'être fait qu'une seul fois.

3. Le serveur étant en fonctionnement (le redémarrer si ce n'est pas le cas), taper dans le navigateur `localhost/Initiation`, puis compléter le formulaire.
   On valide alors ce formulaire en cliquant sur le bouton "Envoyer". Le serveur génère alors le fichier HTML avec les données du formulaire et envoie sa réponse au navigateur qui affiche la page HTML.
   En faisant Ctrl+U, on se rend compte que la page envoyée est bien la page HTML contenant les données fournies à l'aide du formulaire.

4. Observons alors l'URL de cette page.
   Elle commence bien entendu par `http://localhost/Initiation/traitForm.php` car c'est le fichier appelé lors du clic sur le bouton "Envoyer" (le type `"submit"` permet bien d'envoyer au fichier php précisé lors de la déclaration du formulaire les réponses saisies).
   La suite a été évoquée dans la partie I-c sur les URL complètes : dans le formulaire, les valeurs de la propriété "name" deviennent des variables de l'URL et leurs valeurs sont les saisies.
   Cette URL peut être copiée et utilisée dans un autre navigateur, le résultat affiché sera le même.

   On peut modifier le fichier php, par exemple remplacer `$prenom` par `ucfirst($prenom)` et `$nom` par `strtoupper($nom)`. Puis recharger la page traitForm.php. `ucfirst` et `strtoupper` sont deux fonctions PHP (qui ne sont pas à retenir, c'est juste pour l'exemple) permettant de mettre en forme du texte.

5. Dans le formulaire (fichier index.html), ajouter ces deux lignes pour voir d'autres exemples d'utilisation de la balise <input> :

   ```
   <p>Mail : <input type="email" name="email" /></p>
   <p>Mot de passe : <input type="password" name="mdp" /></p>
   ```

   Il n'est pas utile de modifier le fichier php (mais vous pouvez le faire pour voir si vos réponses sont bien traitées et si vous êtes en mesure d'écrire un peu de code PHP).

   Recharger la page index.html, remplir les champs (on peut remarquer que le comportement des 2 zones input ajoutées est différent) puis appuyer sur le bouton "Envoyer".

   Qu'est-ce qui devient gênant dans l'URL obtenue ?

La méthode GET, bien que très pratique car elle permet d'obtenir une URL pointant directement vers la page attendue (comme pour la partie I-c) pose deux soucis :

– l'URL peut devenir très longue avec l'exemple précédent car le commentaire peut contenir un texte très important. Or la taille d'une URL est limitée donc cela peut devenir gênant...

– cette méthode n'est pas du tout sécurisée car tout apparait dans l'URL, y compris le mot de passe.

Une deuxième méthode est plus pertinente pour des formulaires plus important et ceux contenant un mot de passe, c'est la méthode POST.

d) Formulaire et méthode POST

Dans le fichier index.html, il suffit de modifier une ligne :
`<form action="traitForm.php" method="post">` (seule la valeur de l'attribut method a changé : "post" à la place de "get").

Du côté du fichier traitForm.php, on remplace alors tous les `"GET"` par des `"POST"`.

Effectuer le test en lançant le serveur, puis en écrivant `localhost/Initiation` dans la barre d'adresse du navigateur. Remplir le formulaire et l'envoyer.
Cette fois-ci l'URL sera `localhost/Initiation/traitForm.php` quelles que soient les valeurs complétées dans le formulaire. Dans les outils de développement du navigateur, dans l'onglet "Réseau" (ou Network), relancer la page pour vérifier la requête reçue par le serveur :

```
URL de la requête :http://localhost/Initiation/traitForm.php
Méthode de la requête :POST
Version :HTTP/1.1
Code d'état :200 OK
```

Dans la réponse du serveur, on obtient la même chose qu'avec la méthode GET (à condition de remplir les champs de la même façon bien sûr), sauf que cette fois-ci nous ne sommes pas limités en taille.
Même si les renseignements du formulaire ne figurent pas dans l'URL, on constate qu'elles transitent en clair entre le client et le serveur. Une personne ayant des droits suffisants sur une machine intermédiaire pourrait tout à fait intercepter le message et récupérer ces données. C'est pourquoi le protocole HTTP a été sécurisé pour devenir HTTPS et ainsi chiffrer ces données.

On peut ajouter d'autre types d'éléments dans un formulaire comme des cases à cocher ou des listes déroulantes.

Ajouter dans le formulaire ces éléments (pour avoir une liste déroulante) :

```
Classe : <select name="classe">
   <option value="P1">P1</option>
   <option value="P2">P2</option>
   <option value="P3">P3</option>
   <option value="P4">P4</option>
   <option value="P5">P5</option>
   <option value="P6">P6</option>
</select>
```

On peut aussi ajouter des cases à cocher (sous la forme de boutons radio) :

```
Régime : <input type="radio" name="regime" value="Int">Interne</input>
<input type="radio" name="regime" value="DP">Demi-pensionaire</input>
<input type="radio" name="regime" value="Ext">Externe</input>
```

Dans le cas des boutons radio, cliquer sur l'un d'eux le sélectionne et annule la précédente sélection (contrairement aux checkbox). Par contre cela ne le fait que pour les boutons radio possédant le même attribut `name`. Il serait possible de créer une autre suite de boutons radio avec une autre valeur de `name`.

Toujours pour ces boutons radio, on peut permettre de cliquer sur le texte pour faciliter la sélection. Voilà comment on peut faire, par exemple sur le premier bouton :

```
<input type="radio" name="regime" value="Int" id="btnInt">
  <label for="btnInt">Interne</label></input>
```

Il faut faire de même pour les autres.

Ajouter dans le fichier traitForm.php la prise en compte de ces éléments du formulaire.