Chapitre 4
Formulaire d'une page Web

I. Création d'un formulaire

A faire **Activité 1 : Création d'un formulaire**

A. Insertion d'un formulaire

En HTML, pour insérer un formulaire, on se sert de la balise `<form>`.

```
<form method="post" action="traitement_formulaire.php">
      // Les éléments du formulaire sont à insérer ici
</form>
```

La balise `<form>` nécessite au moins deux attributs pour fonctionner.

- L'attribut `method` indique comment doivent être envoyées (au serveur) les données saisies par l'utilisateur. Cet attribut peur prendre deux valeurs : GET et POST.
- L'attribut `action` précise l'URL de la page qui recevra les données du formulaire et qui sera chargée de les traiter. Il s'agit généralement d'un fichier PHP.

B. Eléments d'un formulaire

La balise `<input>` permet d'insérer différents éléments dans un formulaire : des zones de texte, des boutons, des cases à cocher, etc. Chaque élément `input` aura obligatoirement les deux attributs suivants :

- un attribut `type` qui permet de définir le *type de données* à saisir ;
- un attribut `name` qui nomme chaque élément du formulaire et qui correspond au *nom du paramètre qui sera passé au serveur*.

Saisie de textes

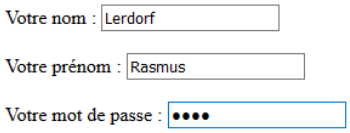
La balise `<input type="text">` permet de saisir une chaîne de caractères. En utilisant l'attribut `type="password"` fonctionne de la même façon mais remplacera les caractères saisis par des « • » pour masquer la saisie : c'est ce qui est utilisé pour saisir des mots de passe.
La balise `<textarea>` permet d'afficher une zone de saisie de texte plus grande.

```
<form method="post" action="traitement_formulaire.php">
    <label for="nom">Votre nom : </label>
    <input type="text" name="nom" id="nom"/><br><br>
    <label for="prenom">Votre prénom : </label>
    <input type="text" name="prenom" id="prenom" /><br><br>
    <label for="mdp">Votre mot de passe : </label>
    <input type="password" name="mdp" id="mdp" /><br><br>
</form>
```

Pour décrire à l'utilisateur ce qu'il doit rentrer dans chaque champ du formulaire, on utilise généralement l'élément `label` que l'on lie à un élément du formulaire grâce aux attributs `for` (pour le `label`) et `id` (pour l'élément du formulaire).

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS

Dans l'exemple ci-contre, la soumission du formulaire entraînerait le passage des paramètres `nom=Lerdorf`, `prenom=Rasmus` et `mdp=1234`



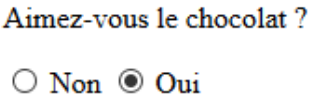
Boutons « radios »

Il s'agit d'un bouton permettant le choix entre plusieurs valeurs. On utilise pour cela la balise <input type="radio">.

```
<p>Aimez-vous le chocolat ?</p>
<input type="radio" name="choixChocolat" value="1" id="choix1">
<label for="choix1">Non</label>

<input type="radio" name="choixChocolat" value="2" id="choix2">
<label for="choix2">Oui</label><br><br>
```

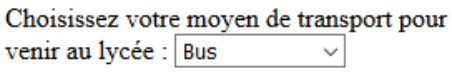
Les boutons appartenant au même groupe doivent avoir la même valeur pour l'attribut `name`. L'attribut `value` définie la valeur qui sera transmise au serveur. Dans l'exemple ci-contre, la soumission du formulaire entraînerait le passage du paramètre `choixChocolat=2`.



Liste

La paire d'élément `select` et `option` permettent de définir une liste de choix. C'est sur l'élément `select` que l'on définit l'attribut `name`. Les éléments `option`, imbriqués dans l'élément `select`, définissent les valeurs des chaînes affichées.

```
<label for="moyen_transport">Choisissez votre
moyen de transport pour venir au lycée : </label>
<select name="transport" id="moyen_transport">
      <option value="bus">Bus</option>
      <option value="voiture">Voiture</option>
      <option value="velo">Vélo</option>
      <option value="autre">Autre</option>
</select>
```

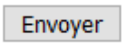


Dans l'exemple ci-dessus, la soumission du formulaire entraînerait le passage du paramètre `transport=bus`.

Le bouton de soumission du formulaire

Un formulaire se termine toujours par un bouton permettant d'envoyer les données du formulaire au serveur : la balise `<button type="submit">` fait ce travail.

```
<p>
<button type="submit">Envoyer</button>
</p>
```



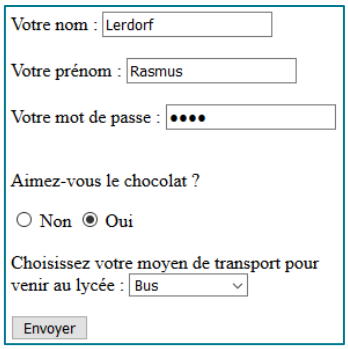
Après un clic sur ce bouton, les données saisies dans le formulaire vont être transmises au serveur par passage de paramètres via une requête HTTP.

II. Passage des paramètres

A faire **Activité 2 : Formulaires et passages de paramètres**

Au moment de la soumission du formulaire au serveur, voici ce qui se passe :

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS



Etape 1 : Pour chaque élément graphique dont la valeur est v_i et le nom (donné par l'attribut `name`) est n_i, le navigateur forme la chaîne de caractères $n_i=v_i$, puis il rassemble tous les éléments entre eux en les séparant par des caractères « & » : $n_1=v_1\&n_2=v_2\&...\&n_k=v_k$.

Dans l'exemple ci-contre, la chaîne construite par le navigateur est :
`nom=Lerdorf&prenom=Rasmus&mdp=1234&choixChocolat=2&transport=bus`

Etape 2 : Le navigateur contacte le serveur (requête) se situant à l'URL donné dans l'attribut `action` de l'élément `form`.

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

Etape 3 : Le navigateur envoie la chaîne de caractères $n_1=v_1\&n_2=v_2\&...\&n_k=v_k$ au serveur. Il y a deux façons de passer cette chaîne au serveur, selon la méthode utilisée.

A. La méthode GET

Avec la méthode GET, cette chaîne de caractères est ajoutée en fin d'URL avec un « ? ».

```
<form method="get" action="traitement_formulaire.php">
```

Les paramètres sont passés au fichier `traitement_formulaire.php` via l'URL :
`traitement_formulaire.php?nom=Lerdorf&prenom=Rasmus&mdp=1234&choixChocolat=2&transport=bus`

Remarques importantes : Comme la méthode GET affiche les paramètres dans l'URL, il est possible de sauvegarder directement l'URL. En revanche, cette méthode n'est pas adaptée pour le passage de données sensibles comme un mot de passe ou des coordonnées bancaires car celles-ci s'affichent à l'écran. De plus, comme la longueur d'une URL ne peut pas être trop longue (quelques milliers d'octets), il est impossible d'utiliser cette méthode pour transmettre des données trop longues (des longs champs de saisie de texte ou le dépôt d'un fichier sur le serveur par exemple).

B. La méthode POST

Avec la méthode POST, la chaîne de caractères est passée au serveur dans le corps de la requête HTTP et n'est pas visible dans l'URL. Sa longueur peut donc être aussi grande que l'on veut.

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

Extrait de la requête HTTP (tous les entêtes n'ont pas été écrits) :

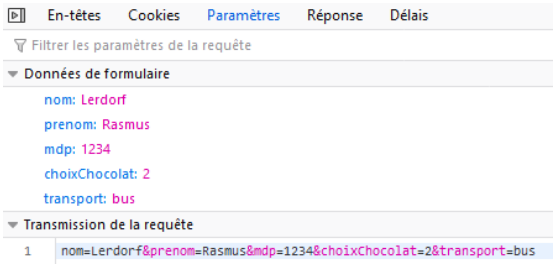
```
POST /traitement_formulaire.php HTTP/1.1
Host: info-mounier.fr
Content-Type: application/x-www-form-urlencoded
Content-Length: 64

nom=Lerdorf&prenom=Rasmus&mdp=1234&choixChocolat=2&transport=bus
```

Chaîne dans le corps de la requête

Remarque importante : La méthode POST est obligatoire pour tout requête effectuant des mises à jour côté serveur (par exemple, pour l'achat d'un billet de concert en ligne, le serveur doit faire une mise à jour puisque le billet n'est alors plus disponible pour les autres personnes). Pour les formulaires demandant un mot de passe, la méthode POST

ne rendra pas visible celui-ci dans la barre d'adresse. Attention, cette méthode n'est pas suffisante pour garantir la confidentialité des données sensibles car celles-ci sont « en clair » dans la requête HTTP. Pour cela, il est nécessaire d'utiliser le protocole HTTPS (et non HTTP) pour chiffrer les échanges entre le client et le serveur (programme de Terminale).



III.Traitement des données côté serveur

Etape 4 : Sur réception de la requête, le serveur Web passe le fichier de l'étape 2 et la chaine de caractères de l'étape 3, à l'interprète PHP (ou Python, …).

Etape 5 : L'interprète copie automatiquement tous les caractères du fichier sur sa sortie standard, jusqu'à rencontrer la balise spéciale `<?php`.

Etape 6 : Cette balise contient du code PHP que l'interprète est capable d'exécuter. Par exemple, l'instruction `echo` permet d'afficher des résultats sur la sortie standard.

Etape 7 : Après avoir rencontré les caractères `?>`, l'interprète reprend sa copie du reste du fichier HTML sur sa sortie standard.

Fichier `traitement_formulaire.php`

```
<?php
 $nom = $_POST["nom"];
 $prenom = $_POST["prenom"];
 $transport = $_POST["transport"];
?>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <title>Formulaire Test</title>
  </head>
  <body>
    <p>Bonjour <?php echo $prenom . " " .
$nom; ?>, bienvenue !</p>
    <p>Je sais que vous venez au lycée en
<?php echo $transport ?>.</p>
  </body>
</html>
```

Interprète PHP

Fichier HTML fabriqué par l'interprète PHP et renvoyé au navigateur

```
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <title>Formulaire Test</title>
  </head>
  <body>
    <p>Bonjour Rasmus Lerdorf,
    bienvenue !</p>
    <p>Je sais que vous venez au
    lycée en bus.</p>
  </body>
</html>
```

Etape 8 : Le serveur Web récupère la sortie standard de l'interprète PHP et la renvoie (le fichier HTML) au navigateur comme réponse à la requête.

Etape 9 : Le navigateur reçoit enfin le fichier HTML qu'il peut alors afficher. Ce fichier ne contient plus de code PHP.

Bonjour Rasmus Lerdorf, bienvenue !

Je sais que vous venez au lycée en bus.

A retenir

Le langage HTML permet de décrire des formulaires permettant à l'utilisateur de saisir des valeurs et de les soumettre à un serveur Web. Il existe deux méthodes de passages des paramètres en HTTP : GET et POST. La méthode GET stocke les paramètres dans l'URL elle-même, alors que la méthode POST stocke ces paramètres dans le corps de la requête.

Ressources :

- *Numérique et Sciences Informatiques*, T. BALABONSKI, S. CONCHON, J.-C. FILLIATRE, K. NGUYEN, Ellipses.