Chapitre 9
Transmettre des données via les formulaires

1. Description d'un formulaire

Vous avez certainement déjà saisi des informations pour vous connecter à un site web, pour dialoguer sur un blog, pour demander des compléments d'information sur un site marchand, etc.

Pour échanger ces informations, vous avez utilisé des formulaires, ils sont partout sur les sites web, c'est le meilleur moyen pour transmettre des données.

Vous verrez dans les paragraphes suivants que pour récupérer les informations saisies par le visiteur, on utilisera essentiellement la méthode `POST` (on abordera succinctement la méthode `GET`, vue au chapitre précédent).

Remarque

Le but de ce chapitre n'est pas de concevoir des formulaires, mais d'en récupérer les données transmises au serveur.

2. La méthode POST

Afin de comprendre la transmission des données via les formulaires, on va suivre un exemple à partir d'un petit formulaire de contact, qui permettra au visiteur de saisir quelques données le concernant et de poser une question.

Voici l'affichage que l'on souhaite obtenir :

Le formulaire est écrit en HTML, en voici le code source :

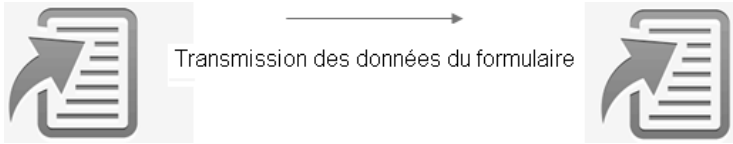
```
<h2>Formulaire de contact</h2>

<form action="ch11_traitement_du_formulaire.php" method="post">
    Nom<br /><input type "text" name="nom"><br /><br />
    Prenom<br /><input type "text" name="prenom"><br /><br />
    Question<br /><textarea rows="5" cols="40"
name="question"></textarea><br />
    <br />
    <input type="submit" name="valider" value="Valider">
</form>
```

On va s'attarder un peu sur la balise `<form>` du formulaire et ses deux attributs importants : **`action`** et **`method`** :

– `Action` : cet attribut contient tout simplement le nom de la page PHP qui sera appelée lorsque le visiteur aura cliqué sur le bouton **Valider**, et c'est dans cette page que l'on pourra récupérer les données saisies par le visiteur.

– `Method` : c'est le mode ou la méthode de récupération des données, vous en connaissez déjà une, `GET` (cf. chapitre Transmettre des données entre les pages) qui permet de récupérer les valeurs via l'URL, et la méthode `POST`, qui permet de récupérer les données transmises à la page décrite dans l'attribut `action`, via la variable superglobale `$_POST`.

Dès que le visiteur va cliquer sur le bouton **Valider,** les données seront transmises à la page PHP "ch11_traitement_du_formulaire.php" :



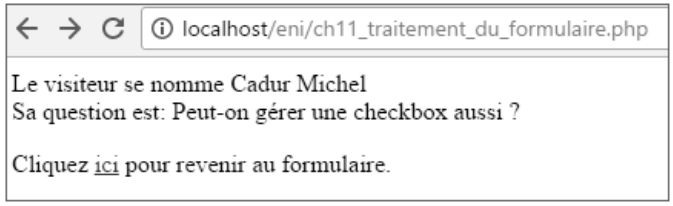
Bien entendu, vous pouvez nommer comme bon vous semble la page définie dans l'attribut `action`, par exemple "reception_données.php", ou "post.php", etc. Vous pouvez même utiliser le même nom que le fichier d'origine : "ch11ex01.php", cela veut dire que la gestion de la récupération des données sera effectuée dans le même fichier que celui qui décrit le formulaire en HTML.

Voici maintenant (ci-dessous) un exemple de fichier PHP qui permettra de gérer les données transmises dans le tableau associatif `$_POST` :

```
<?php
    if (isset($_POST['nom'])) {
           $nom=$_POST['nom'];
    }

    if (isset($_POST['prenom'])) {
           $prenom=$_POST['prenom'];
    }

    if (isset($_POST['question'])) {
           $question=$_POST['question'];
    }
```

```
    echo "Le visiteur se nomme ".$nom." ".$prenom."<br>";
    echo "Sa question est: ".$question."<br>";
?>

<p>Cliquez <a href="ch11_ex01.php">ici</a> pour revenir au formu-
laire.</p>
```

- Testez la page du formulaire, puis voyez le résultat, en supposant que vous ayez saisi les trois champs :



Ce n'est pas plus compliqué que ça ! N'oubliez jamais de tester l'existence des différentes données attendues (on le répète, on n'est jamais à l'abri d'incohérences).

Si vous modifiez le premier fichier en appliquant la méthode `GET`, vous devrez également modifier le second fichier en changeant tous les `$_POST` par `$_GET`.

Comme on l'a évoqué au chapitre précédent, la méthode `GET` permet de passer les données dans l'URL, voici ce que l'on obtiendrait :

La méthode `POST` permettra de gérer tous les types d'éléments du formulaire, c'est ce que l'on va voir ci-dessous.

3. $_POST avec les autres types d'éléments

Dans l'exemple ci-dessus, on a vu les zones de texte classiques et les grandes zones de texte (textarea). Il en existe bien d'autres que vous allez découvrir ci-après.

3.1 Type mot de passe

La zone de texte **mot de passe** est une zone de type texte qui permet de cacher la saisie du visiteur (tous les caractères sont remplacés par des astérisques (*)), voyez plutôt :

Pour obtenir cet affichage, il faudra écrire en HTML :

```
Mot de passe<br /><input type="password" name="motdp">
```

Et pour récupérer la valeur, il faudra écrire en PHP :

```
<?php
  echo $_POST['motdp'];
?>
```

Ceci affichera le mot de passe que vous avez saisi.

3.2 Type Liste déroulante à choix unique

Vous avez déjà tous eu à choisir une valeur dans une liste déroulante comme celle ci-dessous :

Le code HTML de cet exemple :

```
<h2>Formulaire de contact</h2>
    <form action="ch11_ex03_post.php" method="post">
           Quel sport pratiquez-vous?<br />
           <select name="choix_sport">
           <option value="Football">Football</option>
           <option value="Tennis">Tennis</option>
           <option value="Rugby">Rugby</option>
           <option value="Autre">Autre</option>
    </select>
</form>
```

Et pour récupérer la valeur sélectionnée, il faudra utiliser le tableau `$_POST` comme ceci :

```
<?php
  echo $_POST['choix_sport'];
?>
```

Ce code affiche **`Tennis`** si c'est votre choix. C'est la valeur qui est inscrite dans la balise **`value`** qui est transmise à la variable superglobale `$_POST`.

Si vous omettez de définir les attributs **`value`**, c'est la valeur entre les balises **`<option>`** qui sera transmise.