Chapitre 6
Gérer les formulaires et les liens

1. Vue d'ensemble

1.1 Introduction

Dans les sites web dynamiques, il est très souvent nécessaire d'interagir avec l'utilisateur.

En HTML, il existe principalement deux méthodes pour interagir avec un utilisateur :

– les liens (balise `<a>`) ;

– les formulaires (balise `<form>`).

Des scripts PHP peuvent être utilisés pour traiter le clic de l'utilisateur sur un lien ou la saisie de l'utilisateur dans un formulaire.

1.2 Les liens

Le lien est la technique de base qui permet à un utilisateur de naviguer entre les différentes pages d'un site.

Un lien HTML est défini entre les balises `<a>` et `</a>`.

Syntaxe simplifiée

```
<a
  [ href="url" ]
  [ id="identifiant_lien" ]
  [ target="cible" ]
```

```
>
...
</a>
```

Les attributs de la balise <a> sont les suivantes :

`href` URL (*Uniform Resource Locator*) relative ou absolue qui est appelée par le lien.

`id` Identifiant du lien. Si la page HTML contient plusieurs liens, l'identifiant permet de les différencier. En ce qui nous concerne, cet identifiant ne présente pas d'intérêt car il n'est pas récupéré dans le script de traitement du lien. Par contre, il peut être utilisé côté client, en JavaScript par exemple.

`target` Cible (par exemple une autre fenêtre) dans laquelle ouvrir l'URL cible. Par défaut, l'URL cible s'affiche dans la même fenêtre.

L'URL peut contenir des paramètres qui permettent de passer des informations d'une page à une autre.

Syntaxe

```
url_classique?nom=valeur[&...]
```

Le point d'interrogation (?) introduit la liste des paramètres de l'URL séparés par le caractère esperluette (&) ; chaque paramètre est constitué par un couple nom/valeur sous la forme `nom=valeur` :

```
www.monsite.com/info/accueil.php?prenom=Olivier
chercher.php?prenom=Olivier&nom=HEURTEL
```

Exemple

– Script `page1.php`

```
<?php
// Initialisation d'une variable.
$nom='Olivier';
?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">
  <head><meta charset="utf-8" /><title>Page 1</title></head>
  <body>
    <div>
    <!-- lien vers la page 2 en passant la valeur de $nom
         dans l'URL -->
    <a href="page2.php?nom=<?= $nom ?>">Page 2</a>
    </div>
  </body>
</html>
```

– Source de la page dans le navigateur

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">
  <head><meta charset="utf-8" /><title>Page 1</title></head>
  <body>
    <div>
    <!-- lien vers la page 2 en passant la valeur de $nom
         dans l'URL -->
    <a href="page2.php?nom=Olivier">Page 2</a>
    </div>
  </body>
</html>
```

Résultat

– Affichage de la page 1 :

– Résultat du clic sur le lien :

Pour l'instant, aucun nom n'est affiché dans la deuxième page. La variable `$nom` définie dans le script `page1.php` n'est pas disponible dans le script `page2.php` (voir le chapitre Introduction à PHP, section Les bases du langage PHP - Variables - Portée et durée de vie). De plus, notre script ne contient aucune instruction permettant de récupérer les données passées dans l'URL ; nous verrons comment procéder à la section Récupérer les données d'une URL ou d'un formulaire.

1.3 Les formulaires

1.3.1 Petit rappel sur les formulaires

Le formulaire est un outil de base indispensable pour les sites web dynamiques puisqu'il permet à l'utilisateur de saisir des informations et donc d'interagir avec le site.

Un formulaire HTML est défini entre les balises `<form>` et `</form>`.

Syntaxe simplifiée

```
<form
  [ action="url_de_traitement" ]
  [ method="GET"|"POST" ]
  [ id="identifiant_formulaire" ]
  [ target="cible" ]>

...
</form>
```

Les attributs de la balise `<form>` sont les suivants :

`action` URL (*Uniform Resource Locator*) relative ou absolue qui va traiter le formulaire (en ce qui nous concerne, un script PHP). Cet attribut est obligatoire pour se conformer à la recommandation XHTML stricte.

`method` Mode de transmission vers le serveur des informations saisies dans le formulaire.

`GET` (valeur par défaut) : les données du formulaire sont transmises dans l'URL.

`POST` : les données du formulaire sont transmises dans le corps de la requête.

`id` Identifiant du formulaire. Si la page HTML contient plusieurs formulaires, l'identifiant permet de les différencier. En ce qui nous concerne, cet identifiant ne présente pas d'intérêt car il n'est pas récupéré dans le script de traitement du formulaire. Par contre, il peut être utilisé côté client, en JavaScript par exemple.

`target` Cible (par exemple une autre fenêtre) dans laquelle ouvrir l'URL cible. Par défaut, l'URL cible s'affiche dans la même fenêtre.

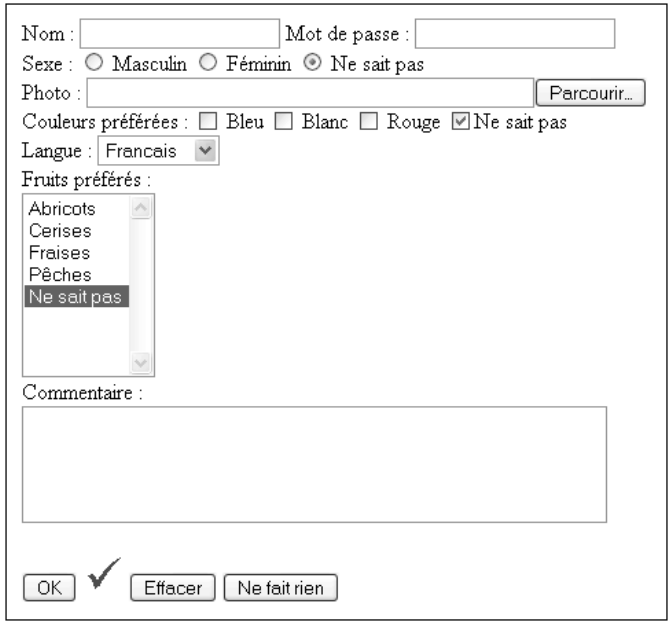
Entre les balises `<form>` et `</form>`, il est possible de placer des balises `<input>`, `<select>` ou `<textarea>` pour définir des zones de saisie.

Exemple (formulaire HTML complet)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">
  <head>
    <meta charset="utf-8" />
    <title>Saisie</title>
  </head>
  <body>
    <form action="saisie.php" method="post">
    <div>
      Nom :
      <input type="text" name="nom" value=""
           size="20" maxlength="20" />
      Mot de passe :
      <input type="password" name="mot_de_passe" value=""
           size="20" maxlength="20" />
      <br />Sexe :
      <input type="radio" name="sexe" value="M" />Masculin
      <input type="radio" name="sexe" value="F" />Féminin
      <input type="radio" name="sexe" value="?"
                          checked="checked" />Ne sait pas
      <br />Photo :
      <input type="file" name="photo" size="50" />
      <br />Couleurs préférées :
      <input type="checkbox" name="couleurs[bleu]" />Bleu
      <input type="checkbox" name="couleurs[blanc]" />Blanc
      <input type="checkbox" name="couleurs[rouge]" />Rouge
      <input type="checkbox" name="couleurs[pas]"
                               checked="checked" />Ne sait pas
      <br />Langue :
      <select name="langue">
        <option value="E">Espagnol</option>
        <option value="F" selected="selected" >Francais</option>
        <option value="I">Italien</option>
      </select>
      <br />Fruits préférés :<br />
      <select name="fruits[]" multiple="multiple" size="8">
        <option value="A">Abricots</option>
        <option value="C">Cerises</option>
        <option value="F">Fraises</option>
        <option value="P">Pêches</option>
        <option value="?" selected="selected">
                         Ne sait pas</option>
      </select>
      <br />Commentaire :<br />
      <textarea name="commentaire" rows="4" cols="50"></textarea>
      <br />
      <input type="hidden" name="invisible" value="123" /><br />
```

```
      <input type="submit" name="soumettre" value="OK" />
      <input type="image" name="valider" alt="valider"
src="valider.gif" />
      <input type="reset" name="effacer" value="Effacer" />
      <input type="button" name="action" value="Ne fait rien" />
    </div>
    </form>
  </body>
</html>
```

Résultat



PHP peut intervenir à deux endroits par rapport au formulaire :

– Pour la construction du formulaire, si ce dernier doit contenir des informations dynamiques.
– Pour le traitement du formulaire (c'est-à-dire des données saisies par l'utilisateur dans le formulaire).

Chapitre 2
Gestion des utilisateurs

Durée : 3 heures 35

Mots-clés

Code PHP, formulaire, requêtes HTTP, service web, base de données, MySQL, phpMyAdmin.

Objectifs

C'est vers la création des premières pages PHP avec accès à la base de données que sont tournés les prochains TP. Ce chapitre n'introduit pas de framework PHP, mais inscrit néanmoins le développement dans un écosystème auquel participent d'autres technologies telles que Node.js ou HTML 5.

Prérequis

Pour valider les prérequis nécessaires, avant d'aborder le TP, répondez aux questions ci-après :

1. Quel est l'attribut de la balise `<form>` qui détermine l'URL de traitement des données d'un formulaire ?
   - **a.** `process`
   - **b.** `data-process`
   - **c.** `action`
   - **d.** `react`
2. Que signifie la valeur `POST` de l'attribut `method` ?
   - **a.** Les données sont envoyées au serveur par mail.
   - **b.** Les données sont envoyées sous forme d'une liste clé-valeur dans la query string de l'URL.
   - **c.** Les données sont signées.
   - **d.** Les données sont envoyées sous forme d'une liste clé-valeur dans le corps de la requête HTTP.

3. Quelle est la variable PHP qui reprend les données postées d'un formulaire ?

- **a.** `$_SERVER`
- **b.** `$_POST`
- **c.** `$_FORM`
- **d.** `$_QUERY`
- **e.** `$_FORM`

4. Quelles sont les principales fonctions de PHP pour exécuter une requête MySQL ?

- **a.** `db_mysql_connect`, `db_mysql_query`, `db_mysql_close`
- **b.** `mysql_connect`, `mysql_query`, `mysql_fetch`, `mysql_close`
- **c.** `php_mysql_query`, `php_mysql_fetch_array`

5. Selon le standard REST, quel est le verbe HTTP pour un appel de service web avec paramètres retournant des données ?

- **a.** `GET`
- **b.** `POST`
- **c.** `GETLIST`
- **d.** `HEAD`

6. Quels sont les formats en sortie généralement produits par un service web REST ?

- **a.** XML
- **b.** JSON
- **c.** PHP
- **d.** HTML

7. Comment s'appelle l'utilitaire de base de données pour MySQL écrit en PHP ?

- **a.** MySQLMyPHP
- **b.** Maria Admin
- **c.** phpMyAdmin
- **d.** MyPHPAdmin

Corrigé p. 115

Énoncé 2.1 Création de la base de données MySQL

Durée estimative : 30 minutes
La première étape dans la réalisation de notre site dynamique consiste à créer la base de données et sa première table `utilisateur'.

- Lancez phpMyAdmin et connectez-vous au serveur local.
- Créez un compte d'utilisateur app_teamup.
- Créez la base de données `teamup` avec un classement de caractère insensible à la casse.
- Créez la table `utilisateur` formée des colonnes suivantes :

Colonne	Attributs de la colonne
`id_utilisateur`	Int, auto_increment, clé primaire
`utilisateur_nom`	Varchar(100)
`utilisateur_login`	Varchar(100)
`utilisateur_pwd`	Varchar(100)
`utilisateur_email`	Varchar(100)
`utilisateur_creation`	Datetime, peut être NULL

- Insérez dans la table deux enregistrements.
- Sélectionnez le contenu de la table

Corrigé p. 116

Énoncé 2.2 Saisie d'un utilisateur avec un template HTML simple

Durée estimative : 45 minutes
Les choses sérieuses commencent ! Ce TP met en place le code PHP pour ajouter des utilisateurs dans la base de données. Même s'il ne repose pas sur la logique MVC, la structure du code suit pourtant un découpage entre différentes préoccupations : présentation, traitement, service et accès aux données.

- Ajoutez dans le répertoire **application** un dossier **dal** (pour Data Access Layer) et dans ce dernier un fichier **userdao.php** contenant une classe vide `UserDAO`.
- Ajoutez dans le répertoire **application** un dossier **service** ainsi qu'un fichier **userservice.php** contenant une classe vide `UserService`.

- Ajoutez dans le répertoire **application/models** un fichier **userentity.php** contenant une classe `UserEntity`. Définissez dans cette classe un champ pour chaque colonne de la table `utilisateur`.
- Ajoutez dans le répertoire **application** un fichier **adduser.php**. Copiez-collez le contenu du fichier **views/_layout.php**, retirez le contenu de la section `<body>` tout en conservant la barre de navigation. Vous devez inclure le fichier de configuration **config.php**.
- Installez un formulaire « auto-postant » avec des champs pour saisir chaque colonne de la table `utilisateur` (sauf la date de création). La clé `id_utilisateur` est représentée par un champ caché. Prévoyez un bouton de soumission du formulaire.
- Ajoutez dans le fichier de configuration des propriétés globales pour se connecter à la base de données MySQL : `$cx_server`, `$cx_login`, `$cx_pwd`, `$cx_dbname`. Définissez dans ce même fichier config.php une fonction globale `get_default_connection()` qui retourne l'ensemble des propriétés de connexion à la base de données.
- Ajoutez dans la classe `UserDAO` un constructeur. Le constructeur doit récupérer les paramètres de connexion depuis la classe `get_default_connection()` et la stocker dans une propriété privée `db_connection`.
- Ajoutez dans la classe `UserDAO` une méthode `adduser($userentity)`. Cette méthode insère dans la table `utilisateur` un enregistrement dont les valeurs de champs sont portées par le paramètre `$userentity`.
- Ajoutez dans la classe `UserService` une méthode `adduser($userentity)`. Cette méthode instancie la classe `UserDAO` et invoque la méthode `adduser`.
- Dans le fichier **utilisateur.php**, aménagez un script PHP et testez si la requête HTTP est de type `POST`. Vous devez extraire les données du formulaire postées et initialiser une instance de `UserEntity`. Instanciez la classe `UserService` et appelez la méthode `adduser`.
- Testez le formulaire, saisissez des valeurs et cliquez sur le bouton de soumission des données. Vérifiez dans la base de données la présence du nouvel enregistrement.

Corrigé p. 120

Énoncé 2.3 Affichage d'un utilisateur avec template réactif (responsive)

Durée estimative : 40 minutes
Cet exercice associe PHP et Bootstrap pour présenter la liste des utilisateurs dans un template réactif. Le code PHP est employé pour effectuer une requête de sélection auprès de la base de données et le framework Bootstrap apporte ses composants d'affichage optimisé pour différents équipements (Web, mobile…).

- Ajoutez dans la classe `UserDAO` la méthode `getuserlist($filtrenom=null)`. Comme dans le TP précédent, récupérez la connexion depuis la propriété `db_connection`. Formez deux requêtes SQL pour sélectionner les enregistrements de la table utilisateur, selon que le paramètre de la méthode `$filtrenom` est null ou non. S'il est null, la requête sélectionne la totalité des enregistrements de la table `utilisateur`. Si le paramètre `$filtrenom` est non null, ajoutez une condition sur la colonne `utilisateur_nom` dans la requête.

Avant d'utiliser l'opérateur SQL `like`, vérifiez que le paramètre `$filtrenom` ne comporte pas de signes pouvant dénaturer la requête SQL et ainsi exposer ce code à des attaques par injection de script SQL. Les signes interdits sont notamment le point-virgule et les apostrophes.

  Instanciez pour chaque enregistrement un objet `UserEntity` et retournez un tableau indexé contenant la collection d'objets obtenue à partir de la lecture de la table `utilisateur`.

- Ajoutez dans `UserService` une méthode `getuserlist($filtrenom)` chargée d'appeler `UserDAO::getuserlist($filtrenom)`.
- Créez une page application/utilisateurs.php et changez dans le fichier **menu.json** la propriété `route='#'` en `route='utilisateurs.php'` pour l'entrée située à l'index 1.

  Reprenez dans utilisateurs.php la mise en page générale depuis _layout.php et testez la page.
- Aménagez un script PHP, instanciez `UserService` et appelez la méthode `getuserlist` sans paramètre.
- Itérez à l'aide d'une boucle `foreach` dans la collection d'objets `UserEntity` et affichez les propriétés `utilisateur_id`, `utilisateur_nom`, `utilisateur_email` dans un tableau.

  Ajoutez les attributs Bootstrap pour rendre le tableau réactif. Testez le tableau.
- Dans un formulaire, ajoutez un champ texte ayant pour identifiant `filtrenom` afin de filtrer la liste à partir du nom. Le formulaire doit aussi comporter un bouton pour soumettre le formulaire.

- Définissez les propriétés du formulaire sur `POST` et utilisateurs.php. Modifiez le script PHP pour appeler la méthode `getuserlist($filtrenom)` si le verbe est un `POST`.

  Testez le filtrage de la liste avec des préfixes de nom.

- Ajoutez un lien vers la page adduser.php.

Corrigé p. 125

Énoncé 2.4 Gestion des équipes

Durée estimative : 60 minutes

Ce TP est en quelque sorte la combinaison des deux précédents. Il s'agit de réaliser l'interface de gestion des équipes d'utilisateurs en reprenant la même logique de séparation du code PHP en plusieurs couches. Patience, ce travail n'est pas superflu et trouvera très vite son utilité quand le site sera plus étoffé.

- Ajoutez dans la base de données les tables `equipe(id_equipe,equipe_nom)` et `utilisateur_equipe(id_equipe,id_utilisateur)`.
- Créez dans Eclipse la classe `TeamEntity` reprenant chaque colonne de la table `equipe` sous la forme d'un champ.
- Ajoutez au projet deux classes `TeamService` et `TeamDAO` comportant les méthodes `getteamlist()`, `addteam($teamentity)`, `editteam($teamentity)`. Employez des requêtes `SELECT`, `INSERT` et `UPDATE` pour implémenter ces trois méthodes.
- Ajoutez les méthodes `getuserteam($id_equipe)`, `adduserteam($id_utilisateur, $id_equipe)`, `removeuserteam($id_utilisateur, $id_equipe)`. Utilisez des requêtes `SELECT`, `INSERT` et `DELETE` pour implémenter ces trois méthodes.

  Définissez également la méthode `getusernotinteam($id_equipe)` qui sélectionne les utilisateurs n'appartenant pas à une équipe.

- Dans le répertoire **application**, ajoutez la page **equipes.php** en reprenant le contenu du fichier **_layout.php**. Installez un lien dans la page **utilisateurs.php** pour naviguer vers **equipes.php**.
- Divisez l'écran **equipes.php** en colonnes : d'abord une zone de texte et une liste pour ajouter et afficher les équipes, puis encore deux listes pour afficher la composition d'une équipe et les utilisateurs n'appartenant pas à cette équipe.

  Chargez la liste des équipes avec la classe `TeamService`.

  Ajoutez un bouton de soumission `cmd_addteam` pour créer une nouvelle équipe.

  Dans le code `POST`, appelez la méthode `addteam()` et rafraîchissez le contenu de la liste des équipes.