SYSTEMES D'INFORMATION 2

TP 04 – PHP

Jean-Christophe ROLDAN & Samuel MERTENAT

Télécommunications
T3a & T3f

Fribourg, le 12 novembre 2015

TABLE DES MATIERES

SYSTEMES D'INFORMATION 2

TP 04 - PHP

1 INTRODUCTION

Ce quatrième travail pratique est l'occasion de se familiariser avec la distribution « XAMPP » (« MAMP » sous Mac OS X) qui permet de disposer facilement d'un serveur Web, d'un serveur MySQL et des dépendances liées au langage PHP. D'autre part, nous nous intéresseront plus précisément à ce dernier langage afin de développer différentes fonctions et de communiquer avec une base de données.

2 MISE EN PLACE

Pour la réalisation de ce travail pratique, il est nécessaire de mettre en place un serveur Web, un serveur de bases de données et les dépendances nécessaires à l'utilisation du langage de programmation PHP.

Le développement et les tests sont réalisés sur une machine Macintosh et la distribution MAMP a été installée afin de faciliter l'installation des outils cités précédemment.

Ordinateur :

- Macbook Pro Retina 15.4, Mac OS X EL Capitan, version 10.11.1

Logiciels :

- Apache, version 2.2.29, port 8888
- MySQL, version 5.5.42, port 8889
- PHP, version 5.6.10
- phpMyAdmin, version 4.4.10

Les fichiers sources sont placés dans le dossier « MAMP/htdocs/lab4 ».

Pour stocker les données, nous avons créé une base de données « phplab », qui contient une table appelée « employee ». Cette table est créée à l'aide du script fourni sur le réseau « db_setup.sql ».

Afin de pouvoir communiquer avec la base de données depuis PHP, nous avons créé un fichier « db.inc.php » à inclure dans chacune de nos pages nécessitant une connexion avec les données.

Pour une connexion persistante: p:localhost:…

```php
<?php
  // http://www.w3schools.com/php/php_mysql_select.asp
  $servername = "localhost:8889";
  $username = "root";
  $password = "root";
  $dbname = "phplab";
```

Pour enlever les warnings: @myali

```
    // Creates a connection to the database
    $conn = new mysqli($servername, $username, $password, $dbname);

    // Checks the connection
    if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error);
    }
  ?>
```

3 CONCEPTION

Le but est de réaliser une petite application, qui nous permettra de réaliser les actions suivantes :

- Ajouter une personne dans la base de données
- Afficher la liste des personnes enregistrées
- Supprimer une personne de la liste, en cliquant sur la croix.

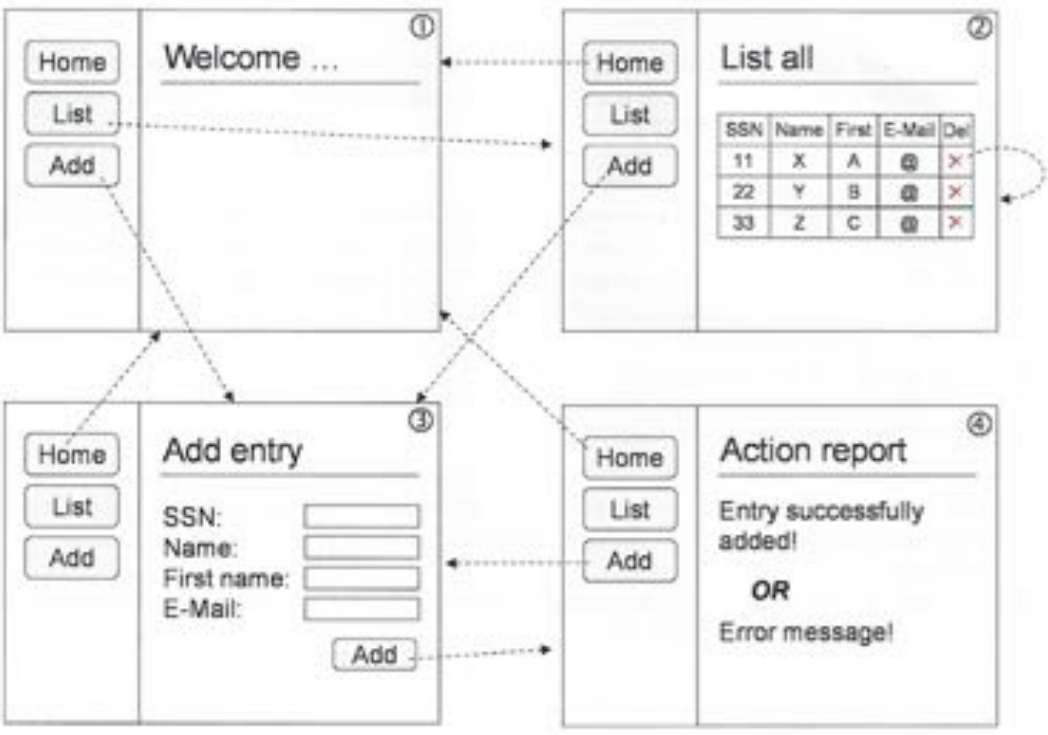

Figure 1: Spécification de l'interface

4 REALISATION

4.1 AJOUT D'UNE NOUVELLE ENTREE

La page « form.php » contient, comme son nom l'indique, un formulaire pour l'ajout d'une entrée dans la base de données. Les champs « SSN » et « Name » sont obligatoires, les autres sont facultatifs. Le formulaire est identique au modèle fourni sur Moodle. Lorsque l'utilisateur appuie sur le bouton « Add », les données sont transmises au fichier « add_entry.php », par le biais de la méthode POST.

D'autre part, nous comptabilisons le nombre d'entrée(s) réalisé(s) durant la session, à l'aide des variables du même nom. Pour réaliser cela, nous devons impérativement utiliser l'instruction ci-dessous, au tout début du fichier « header.inc.php ». Par la même occasion, nous initialisons la variable de session « count_entry » à zéro, si celle-ci n'a pas encore été déclarée.

```
<?php
  // Start the session
  // http://www.w3schools.com/php/php_sessions.asp
  session_start();
  // Creates a session variable to store the number of entries during the session
  if (!isset($_SESSION['count_entry']))
    $_SESSION['count_entry'] = 0;
?>
```

Dans le fichier « add_entry.php », nous débutons par vérifier que les paramètres ont été bien transmis par le formulaire. Si ce n'est pas le cas, nous affichons un message à l'utilisateur, en lui demandant de renouveler la procédure.

```
      <?php
        // define variables and set to empty values
        $ssn = $lastname = $firstname = $email = "";

        if (isset($_POST['Submit']) && $_SERVER["REQUEST_METHOD"] == 'POST') {
          if (empty($_POST['ssn']) || empty($_POST['lastname'])) {
      ?>
            <p>ERROR. Mandatory fields are missing.</p>
            <p>Go back to the <a href="form.php">form</a>.</p>
      <?php
          } else {
            // Increments the number of entries
            $_SESSION['count_entry']++;
            add_entry($_POST['ssn'], $_POST['lastname'], $_POST['firstname'], $_POST['email']);
          }
        }
  ?>
```

Par contre, si les paramètres sont correctes, nous incrémentons la variable de session et nous appelons la fonction « add_entry() », permettant d'ajouter une nouvelle personne à la base de données.

Cette fonction requiert une instance de la liaison à la base données et effectue une requête SQL « INSERT INTO ». Nous affichons ensuite à l'utilisateur le nombre d'entrée(s) effectuée(s) durant la session courante et nous lui proposons de réaliser une nouvelle entrée ou de visionner la liste de celles-ci.

```
<?php
  require('footer.inc.php');

  function add_entry($ssn, $lastname, $firstname, $email) {
    require_once('db.inc.php');

    $sql_query = "INSERT INTO employee (ssn, lastname, firstname, email) values ('$ssn', '$lastname',
'$firstname', '$email')";

    if (!$conn->query($sql_query))
      die('Invalid query ' . mysql_error());
    $conn->close();

    echo "<p>Entry successfully added !</p>";
    if ($_SESSION['count_entry'] == 1) {
      echo "<p>This is your first entry</p>";
    } else {
      echo "<p>This is not your first entry! You have already added " . $_SESSION['count_entry'] . "
entries.</p>";
```

```
    }

    echo "<p>Go back to the <a href='form.php'>form</a> or to the <a href='list.php'>list</a>.</p>";
  }
?>
```

4.2 AFFICHAGE DES ENTREES

La page « list.php » permet de lister toutes les entrées de la base de données, en fonction d'une requête SQL. Si la base de données ne contient aucune entrée, un message est affiché à l'utilisateur. Sinon, une table de personnes est générée dynamiquement et une croix, placée à côté de chaque entrée, permet de la supprimer.

Nous débutons par récupérer l'instance de la connexion à la base de données, par le biais du fichier « db.inc.php ». Nous effectuons ensuite la requête SQL « SELECT * FROM employee » afin de récupérer toutes les entrées de la table « employee ».

```
    require_once('db.inc.php');

    // Retrieves the data from the DB
    $sql = "SELECT * FROM employee";
    $result = $conn->query($sql);
```

En fonction du nombre d'entrée(s) récupérée(s), nous générons une table, avec un formulaire pour chaque entrée où nous affichons un message à l'utilisateur « 0 results stored in the database ».

```
   if ($result->num_rows > 0) {
      // Creates a table with a border of 2 and 4 columns
      echo
        "<table border='2'>".
          "<th>SSN</th>".
          "<th>Name</th>".
          "<th>Firstname</th>".
          "<th>E-mail</th>".
          "<th>Del</th>";

      // Enters the data of each row
      // Form: http://www.w3schools.com/php/php_forms.asp
      while($row = $result->fetch_assoc()) {
        echo "<tr>".
          "<td>{$row['ssn']}</td>".
          "<td>{$row['lastname']}</td>".
          "<td>{$row['firstname']}</td>".
          "<td>{$row['email']}</td>".
          "<td>".
            "<form action=remove_entry.php method=post>".
              "<input type=image src=img/delete.png />".
              "<input type=hidden name=ssn_id value={$row['ssn']} />".
            "</form>".
          "</td>".
          "</tr>";
      }
      // Closes the table
      echo "</table>";
    } else {
      echo "0 results stored in the database";
    }
```

Nous créons un formulaire pour chaque entrée afin de pouvoir transmettre l'identifiant « ssn », par le biais d'un champ caché du formulaire, de la personne à supprimer en cas de besoin. Le formulaire pointe ensuite vers le fichier « remove_entry.php », qui nécessite, par le biais de la méthode POST, l'ID de la personne à effacer de la base de données.

Pour finir, nous fermons la connexion à la base par le biais de l'instruction suivante :

```
    $conn->close();
```

4.3 SUPPRESSION D'UNE ENTREE

La suppression d'une entrée se fait à l'aide du fichier « remove_entry.php », appelé suite à un clic sur la croix, à côté d'une entrée (fichier « list.php »). Ce script nécessite qu'on lui transmette, par le biais de la méthode POST, l'identifiant d'une personne.

Nous débutons par vérifier si le paramètre a bien été transmis et que celui-ci est bien un nombre.

```
  if (!isset($_POST['ssn_id'])) {
    echo "The parameter ssn was not set or sent correctly";
  } else if (!is_numeric($_POST['ssn_id'])) {
    echo "The parameter ssn is not a number";
  } else {
```

Si c'est bien le cas, nous récupérons ensuite l'instance de la connexion à la base de données, par le biais du fichier « db.inc.php ». Nous préparons ensuite la requête SQL destinée à supprimer la personne désirée.

```
    require_once('db.inc.php');

    // Prepares the query
    // http://www.w3schools.com/php/php_mysql_prepared_statements.asp
    $stmt = $conn->prepare("DELETE FROM employee WHERE ssn = ?");
    $stmt->bind_param("i", $_POST[ssn_id]); // "i" for integer
    $stmt->execute();

    $stmt->close();
    $conn->close();
```

Attention: Si l'on rafraîchit la page, veillez à ne pas effectuer à nouveau la requête

Nous indiquons ensuite à l'utilisateur que l'opération s'est bien déroulée et nous l'invitons à consulter à nouveau la liste. Nous souhaitions, au préalable, rediriger l'utilisateur vers la page « list.php » à l'aide de l'instruction « header('Location: list.php'); », mais nous avons rencontré des difficultés à la faire fonctionner suite à l'utilisation d'un fichier séparé pour la connexion à la base de données.

```
<div id="main">
  <h2>Action report</h2>
  <p>The employee has been successfully deleted.</p>
  <p>Go back to the <a href="list.php">list</a>.</p>
</div>
```

5 TESTS ET VALIDATION

Afin de vérifier le bon fonctionnement de l'application, nous réalisons les tâches suivantes :

- Ajouter une entrée sans tous les champs requis
- Ajouter une entrée avec tous les champs requis
- Visionner la liste
- Supprimer l'entrée ajoutée précédemment
- Visionner à nouveau la liste

5.1 AJOUT D'UNE NOUVELLE ENTREE

Nous débutons par ajouter une nouvelle entrée, sans renseigner tous les champs obligatoires. Nous obtenons le message d'erreur suivant, nous invitant à réaliser à nouveau la démarche.

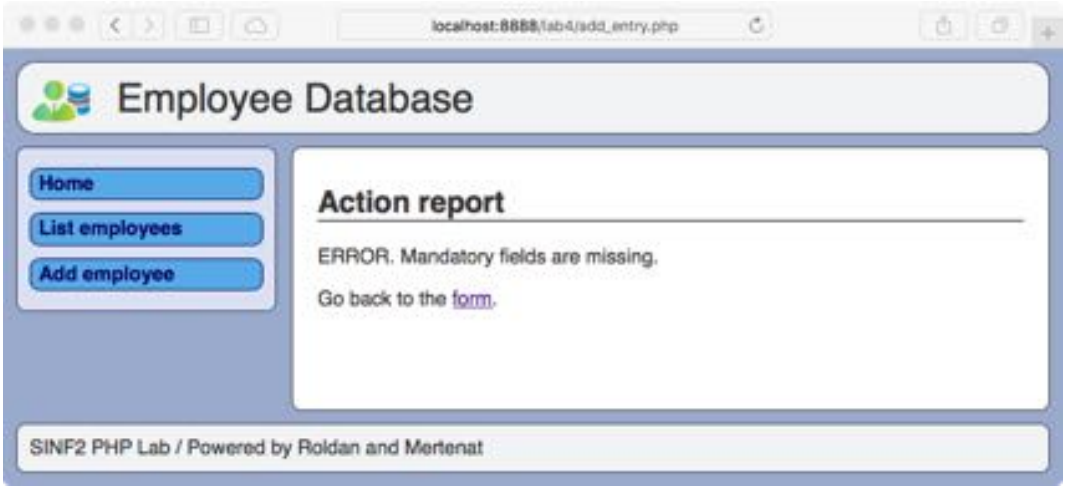

Figure 2: Erreur lors de champs obligatoires manquants

En remplissant cette fois-ci le formulaire de manière rigoureuse, nous pouvons observer le message ci-dessous, nous indiquant le bon fonctionnement de l'opération.

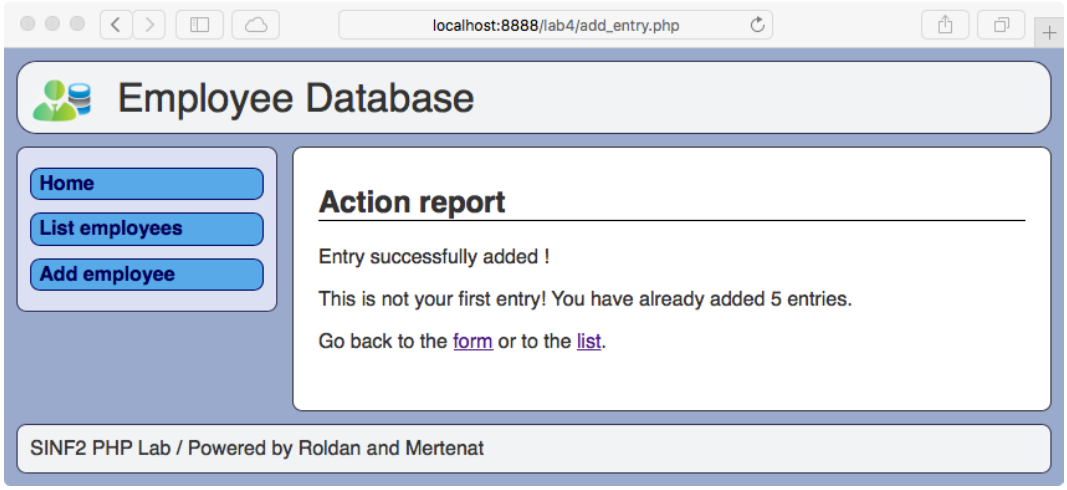

Figure 3: Ajout d'une nouvelle entrée avec succès

Si l'entrée n'avait pas été la 5ème de l'utilisateur, nous nous serions vus notifier du message suivant « This is your first entry ».

5.2 AFFICHAGE DES ENTREES

En visionnant cette fois-ci la liste des entrées, en appuyant sur le bouton « List employees », nous obtenons la vue ci-dessous.

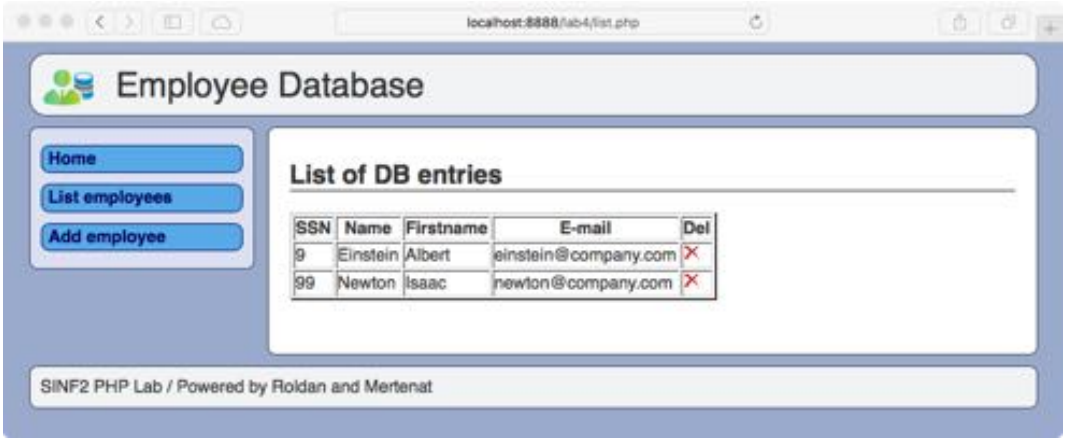

Figure 4: Liste avec des entrées

Si la liste avait été vide, nous n'aurions pas eu un tableau, mais simplement le message visible sur la figure ci-dessous.

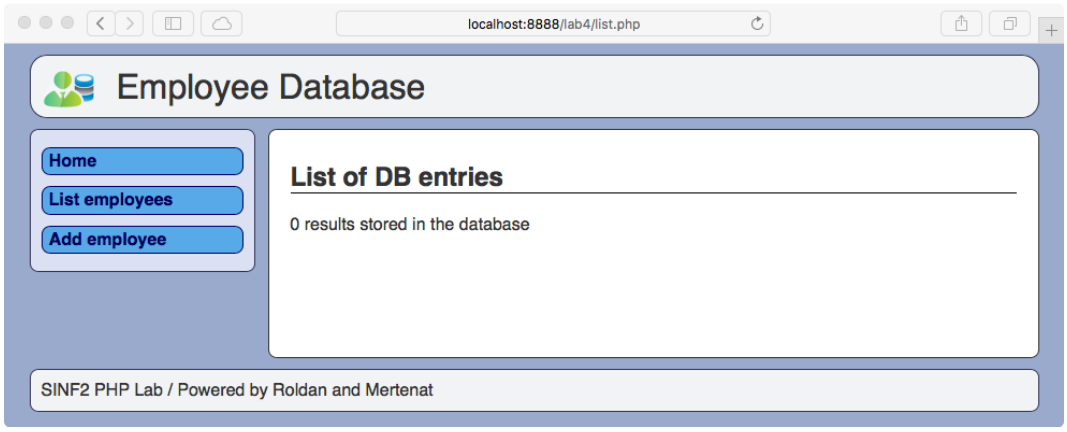

Figure 5: Liste vide

5.3 SUPPRESSION D'UNE ENTREE

En décidant de supprimer un employer de la liste, il suffit de cliquer sur la croix, en rouge, à côté de celui-ci. En effaçant un de ceux-ci, le script nous informe de la bonne conduite de l'opération et nous invite alors à consulter à nouveau la liste.

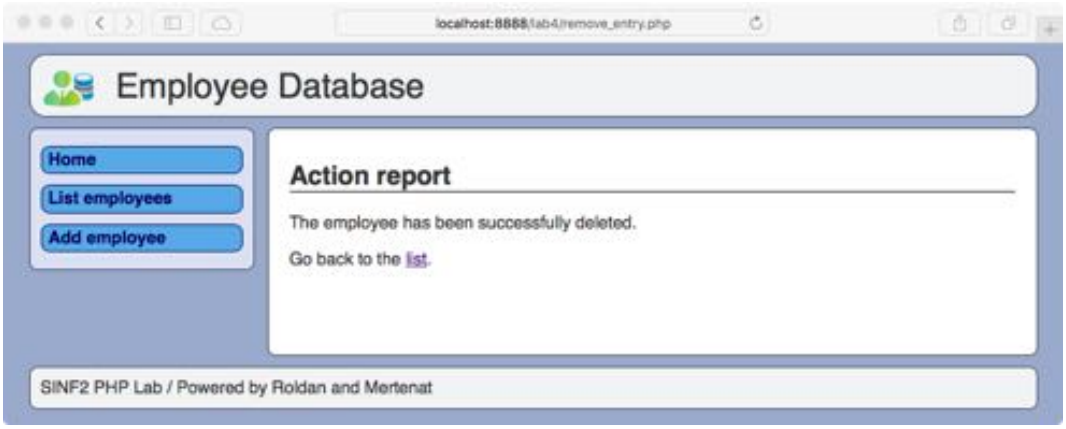

Figure 6: Après la suppression d'une entrée

6 CONCLUSION

Ce travail pratique a été l'occasion d'appréhender le langage PHP en réalisant de petits scripts afin de peupler une base de données, d'en afficher les données ou de permettre à l'utilisateur de supprimer une entrée, à l'aide d'une petite application Web.