SYSTEMES D'INFORMATION 2

TP 05 – SERVLETS

Jean-Christophe ROLDAN & Samuel MERTENAT

Télécommunications
T3a & T3f

Fribourg, le 11 décembre 2015

TABLE DES MATIERES

SYSTEMES D'INFORMATION 2

TP 05 - SERVLETS

1 INTRODUCTION

Ce cinquième travail pratique est l'occasion de se familiariser avec un serveur d'application Apache Tomcat et de développer une petite application utilisant des Servlets.

2 MISE EN PLACE

Pour la réalisation de ce travail pratique, il est nécessaire de mettre en place un serveur Tomcat. D'autre part, nous utiliserons le serveur de bases de données installé lors du TP précédent (PHP).

Le développement et les tests sont réalisés sur une machine Macintosh et la configuration est la suivante :

Ordinateur :

- Macbook Pro Retina 15.4, Mac OS X EL Capitan, version 10.11.2

Logiciels :

- Apache Tomcat, version 8.0.30, port 8080
- MySQL, version 5.5.42, port 3306
  - Utilisateur : sinf
  - Mot de passe : classT3
  - Base de données : phplab

3 REALISATION

3.1 HELLOWORLD

Les différents Servlets sont listés dans un fichier « web.xml » (servlet-lab/WEB-INF/) et permet d'indiquer le mappage des liens. Pour que le Servlet « HelloWorld » fonctionne, il est nécessaire de modifier l'url d'appel de celui-ci, pour concorder avec le lien utilisé dans le fichier « index.html » (
HelloWorld servlet).

```
  <!-- HelloWorld -->
  <servlet>
    <description>Classic hello world</description>
    <display-name>HelloWorld Servlet</display-name>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>servlet.HelloWorld</servlet-class>
  </servlet>
```

```
  <servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/hello/world</url-pattern>
  </servlet-mapping>
```

Suite à cette modification, nous pouvons alors appeler correctement le Servlet « HelloWorld ».

Figure 1: Hello World

3.2 ENTETES D'UNE REQUETE HTTP

Pour récupérer l'entête d'une méthode GET ou POST, nous créons un Servlet « HTTPHeadersServlet » et nous ajoutons les lignes suivantes dans le fichier « web.xml ».

```
  <!-- HTTPHeadersServlet -->
  <servlet>
    <description>HTTPHeadersServlet servlet</description>
    <display-name>HTTPHeadersServlet Servlet</display-name>
    <servlet-name>HTTPHeadersServlet</servlet-name>
    <servlet-class>servlet.HTTPHeadersServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HTTPHeadersServlet</servlet-name>
    <url-pattern>/servlets/HTTPHeadersServlet</url-pattern>
  </servlet-mapping>
```

Pour récupérer l'entête et les éventuels paramètres, nous utilisons les deux fonctions ci-dessous :

```
 private void getHeaderFields(HttpServletRequest req, PrintWriter out) {
  out.println("<h2>Header fields (method: " + req.getMethod() + "): </h2>");
  out.println("<h3> name : value </h3>");

  Enumeration<String> headers = req.getHeaderNames();
  while (headers.hasMoreElements()) {
    String header = headers.nextElement();
    String value = req.getHeader(header);

    out.println("<p>" + header + " : " + value + "</p>");
  }
 }

 private void getParameters(HttpServletRequest req, PrintWriter out) {
  out.println("<h2>Parameters : </h2>");
  out.println("<h3> name : value </h3>");

  Enumeration<String> params = req.getParameterNames();
  while (params.hasMoreElements()) {
    String param = params.nextElement();
    String value = req.getParameter(param);
```

```
    out.println("<p>" + param + " : " + value + "</p>");
  }
 }
```

3.2.1 POST

Pour récupérer l'entête et les paramètres d'une méthode POST, nous devons créer un petit formulaire, que nous avons ajouté sur la page d'accueil (« index.html »). Ce formulaire utilise la méthode POST et pointe vers le Servlet « HTTPHeaderServlet ».

```
<!-- POST FORM -->
<h3>Post Form</h3>
<form action="servlets/HTTPHeadersServlet" name="myform" method="POST">
  <table>
    <tr>
      <td><b>Param 1</b> :</td>
      <td><input type="text" name="param1" size="20"  /></td>
    </tr>
    <tr>
      <td><b>Param 2</b> :</td>
      <td><input type="text" name="param2" size="20"  /></td>
    </tr>
    <tr>
      <td></td>
      <td align="right">
        <input type="submit" name="submit" value="Post" >
      </td>
    </tr>
  </table>
</form>
```

Sur les deux figures ci-dessous, nous pouvons constater que les informations introduites dans le formulaire sont bien affichées en tant que paramètres à l'aide du Servlet.

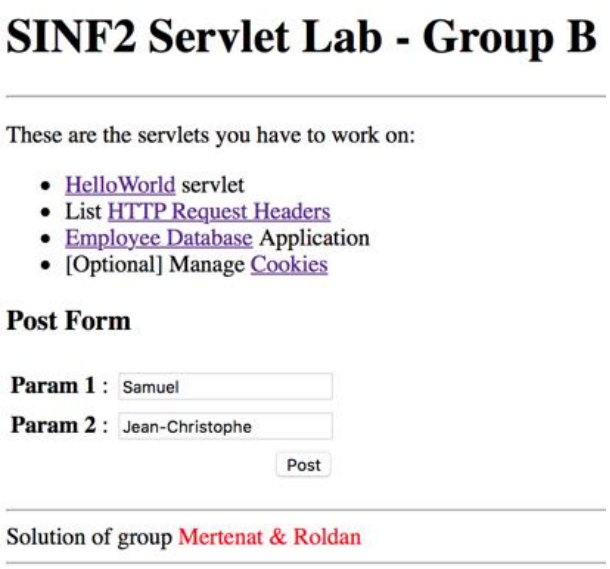

Figure 2: Formulaire POST

Figure 3: Entête de la méthode POST

3.2.2 GET

Avec la méthode GET, nous devons passer les paramètres depuis l'URL. Pour arriver au même résultat qu'avec la méthode POST, nous devons utiliser l'URL suivante : « localhost:8080/servlet-lab/servlets/HTTPHeadersServlet?param1=Samuel&param2=Jean-Christophe ».

Figure 4: Entête GET

3.3 APPLICATION MYSQL

Pour l'utilisation de la base de données, nous devons également faire de nouvelles entrées dans le fichier « web.xml » pour signifier l'utilisation du Servlet « EmployeeDB ».

```
<!-- EmployeeDB -->
<servlet>
  <description>EmployeeDB servlet</description>
  <display-name>EmployeeDB Servlet</display-name>
  <servlet-name>EmplDB</servlet-name>
  <servlet-class>servlet.EmployeeDB</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>EmplDB</servlet-name>
  <url-pattern>/empl-db/app</url-pattern>
</servlet-mapping>
```

Et des modifications dans la méthode « doGet() », afin de pouvoir utiliser les différentes fonctions liées à l'ajout ou à la suppression d'une entrée (paramètre « action »).

```
    // dispatch action
    switch (requestedAction) {
      case "add":
        actionAdd(req, resp);
        break;
      case "list":
        conn = getDBConnection(out);
        actionList(conn, out);
        break;
      case "form":
        actionForm(req, resp);
```

```
        break;
      case "delete":    // add Delete functionality
        actionDelete(req, resp);
        break;
      case "index":
      default:
        actionIndex(req, resp);
        break;
    }
```

3.3.1 AJOUT D'UNE NOUVELLE ENTREE

Le formulaire d'ajout se trouve dans le fichier « form.html », cependant il est nécessaire de modifier le lien pointant vers l'action à effectuer lors de la validation de celui-ci : « <form action="?action=add" name="myform" method="POST"> ».

La fonction d'ajout se présente ainsi, avec l'utilisation d'une requête préparée pour l'ajout à la base MySQL.

```
  private void actionAdd(HttpServletRequest req, HttpServletResponse resp)
      throws IOException {

    final PrintWriter out = resp.getWriter();
    Connection conn = getDBConnection(out);
    try {
      String ssn = req.getParameter("ssn");
      if (ssn == null || ssn.equals("") || ssn.length() > 3
          || Integer.parseInt(ssn) == 0) {
        throw new Exception("SSN not valid");
      }
      String lastname = req.getParameter("lastname");
      if (lastname == null || lastname.equals("")) {
        throw new Exception("Lastname missing");
      }
      String firstname = req.getParameter("firstname");
      String email = req.getParameter("email");

      String query = "INSERT INTO employee(ssn, lastname, firstname, email) VALUES(?, ?, ?, ?)";
      PreparedStatement stmt = conn.prepareStatement(query);

      stmt.setString(1, ssn);
      stmt.setString(2, lastname);
      stmt.setString(3, firstname);
      stmt.setString(4, email);

      stmt.execute();
      stmt.close();

      out.write("<h2>Action report</h2>");
      out.write("<p>Entry successfully added !</p>");
      out.write("<p>Go back to the <a href=\"?action=list\">list</a>.</p>");

    } catch (Exception e) {
      out.write("<h2>Action report</h2>");
      out.write("<p>An error has occured !</p>");
      out.write("<p>" + e.getMessage() + "</p>");
      out.write("<p>Go back to the <a href=\"?action=form\">form</a>.</p>");
    } finally {
      out.println("</body></html>");
      out.close();

      try {
        conn.close();
```

```
      } catch (SQLException e) {
        printSQLException(e, out);
      }
    }
  }
```

Si l'on ajoute une nouvelle entrée, nous obtenons la figure ci-dessous.

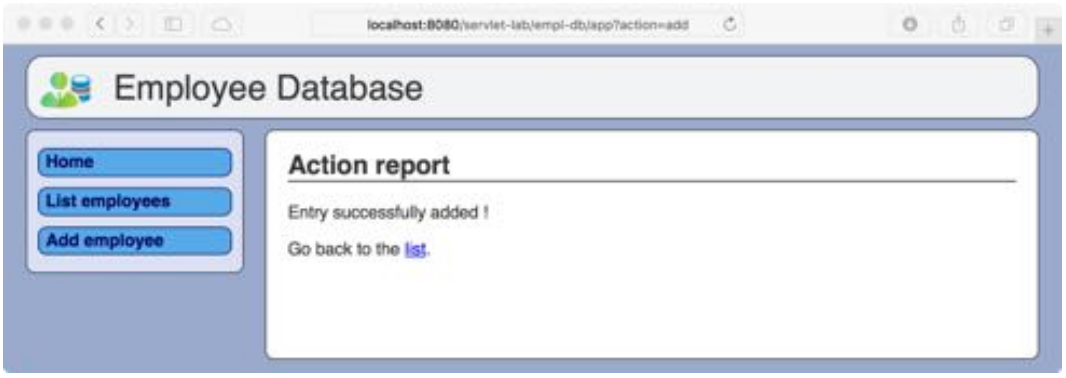

Figure 5: Ajout d'une entrée avec succès

Cependant, si la valeur du champ ssn est déjà présente dans la BDD ou que l'utilisateur rafraichit la page, nous obtenons la figure suivante :

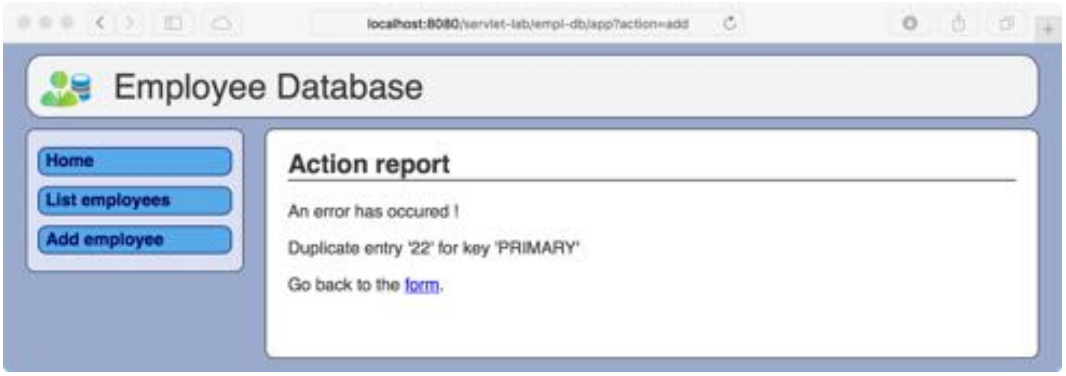

Figure 6: Ajout d'une entrée non fructueuse

3.3.2 AFFICHAGE DES ENTREES

Pour l'ajout de la croix rouge à côté de chaque entrée, quelques petits ajouts sont à réalisés dans le code de la méthode ci-dessous. Cette croix est ajoutée à l'aide d'un formulaire, qui contient également un champ caché pour stocker la valeur du SSN de la personne.

```
  // produce an HTML table containing the results
  private void writeHTMLTableFromResultSet(PrintWriter out, ResultSet rs)
      throws SQLException {

    out.println("<table class=\"employee-list\">");

    // write header row
    ResultSetMetaData metadata = rs.getMetaData();
    int numCols = metadata.getColumnCount();
    out.println("<tr>");
    for (int i = 1; i <= numCols; i++) {
```

```
      out.print("<th>" + metadata.getColumnName(i) + "</th>");
    }             // end for (columns)
    out.println("<th>actions</th>"); // additional column for actions
    out.println("</tr>");

    // write entry rows
    while (rs.next()) {
      out.println("<tr>");
      for (int i = 1; i <= numCols; i++) {
        out.print("<td>" + rs.getString(i) + "</td>");
      }             // end for (columns)
      // additional column for action icons/links
      out.println("<td align=\"center\">");
      out.println("  <form action=app?action=delete method=post>");
      out.println("    <input type=image src=img/delete.png />");
      out.println(
          "    <input type=hidden name=ssn value= " + rs.getString(1) + " />");
      out.println("  </form>");
      out.println("</td>");
      out.println("</tr>");
    }
    out.println("</table>");
  }
```

Si nous affichons la liste, nous obtenons la grille ci-dessous. On peut noter la présente de la croix rouge, permettant de supprimer une entrée.

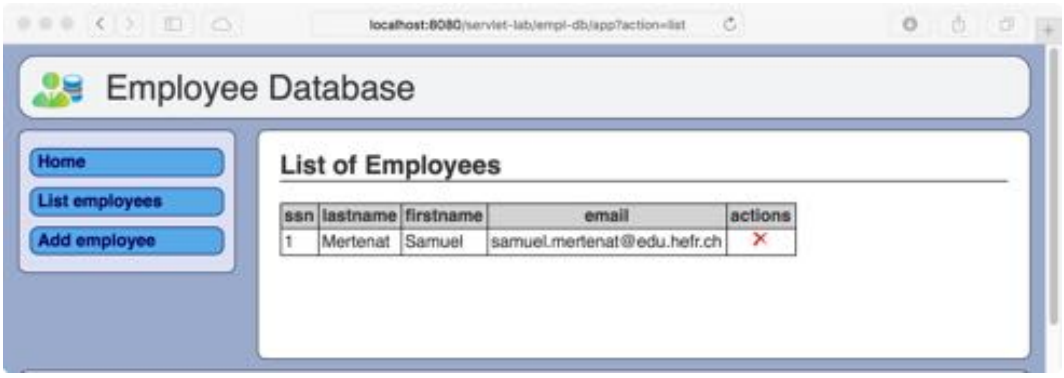

Figure 7: Liste des employés

3.3.3 SUPPRESSION D'UNE ENTREE

Suite à la pression sur la croix rouge, la fonction « actionDelete » du Servlet est appelée. Cette méthode supprime l'entrée, en fonction du ssn passé avec la méthode Post.

```
  private void actionDelete(HttpServletRequest req, HttpServletResponse resp)
      throws IOException {

    final PrintWriter out = resp.getWriter();
    Connection conn = getDBConnection(out);
    try {
      String ssn = req.getParameter("ssn");
      if (ssn == null || ssn.equals("") || ssn.length() > 3
          || Integer.parseInt(ssn) == 0) {
        throw new Exception("SSN not valid");
      }

      String query = "DELETE FROM employee WHERE ssn = ?";
      PreparedStatement stmt = conn.prepareStatement(query);
      stmt.setString(1, ssn);
```

```
      stmt.execute();
      stmt.close();

      out.write("<h2>Action report</h2>");
      out.write("<p>Entry successfully deleted !</p>");
      out.write("<p>Go back to the <a href=\"?action=list\">list</a>.</p>");
    } catch (Exception e) {
      out.write("<h2>Action report</h2>");
      out.write("<p>An error has occured !</p>");
      out.write("<p>" + e.getMessage() + "</p>");
    } finally {
      out.write("</body></html>");
      out.close();

      try {
        conn.close();
      } catch (SQLException e) {
        printSQLException(e, out);
      }
    }
  }
```

Suite à la suppression d'une entrée, nous sommes redirigés vers une page qui nous notifie de la bonne réalisation de la tâche.

Figure 8: Suppression d'une entrée avec succès

3.4 EXTENSIONS

3.4.1 SESSION

Pour stocker le nombre d'entrée(s) dans une variable de session, il nous faut modifier la méthode « actionAdd() » et ajouter les lignes ci-dessous.

```
      // session counter
      HttpSession session = req.getSession(true);
      Object cnt = session.getAttribute("count_entry");

      int counter = cnt == null ? 0 : (int) cnt;
      session.setAttribute("count_entry", ++counter);

      out.write("<h2>Action report</h2>");
      out.write("<p>Entry successfully added !</p>");
      if (counter == 1) {
        out.write("<p>This was your first entry</p>");
      } else {
        out.write("<p>This was not your first entry! You have already added "
            + counter + " entries.</p>");
```

```
        }
        out.write("<p>Go back to the <a href=\"?action=list\">list</a>.</p>");
```

En ajoutant ensuite une ou deux entrées, nous pouvons constater un message différent, avec le nombre effectif d'entrée(s).

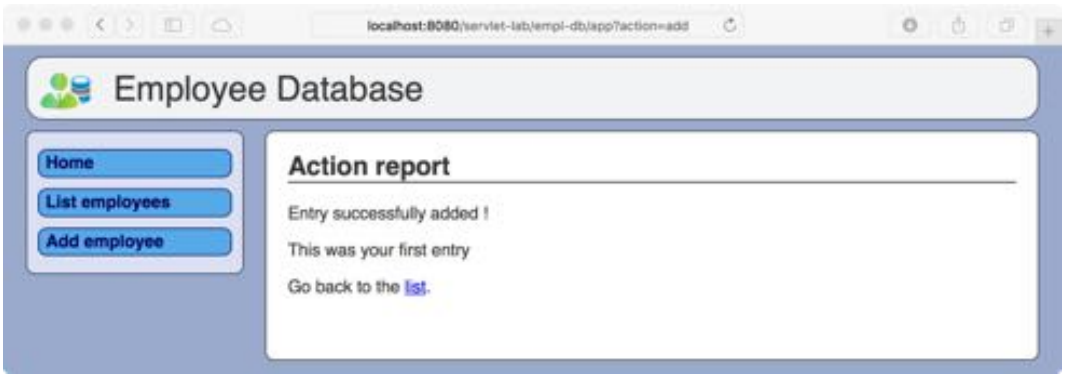

Figure 10: Première entrée ajoutée

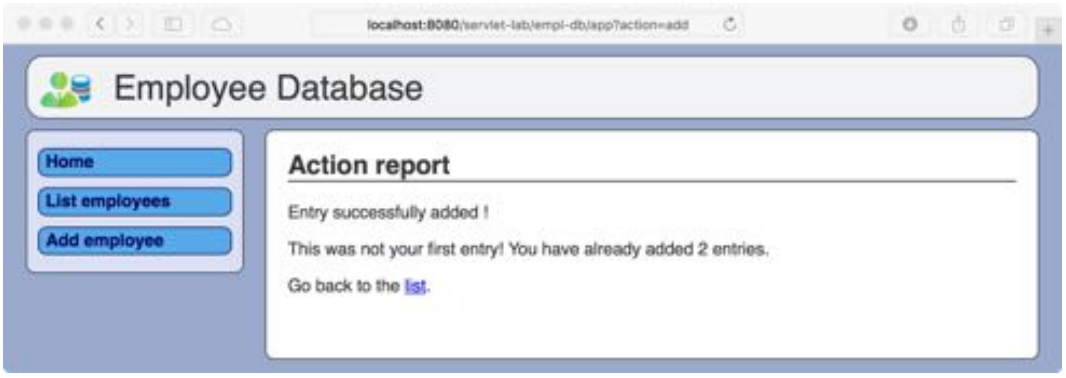

Figure 9: Ajout d'une seconde entrée

3.4.2 MODIFICATION D'UNE ENTREE

Pour réaliser cette fonctionnalité, nous devons ajouter deux nouveaux cas de figure à la méthode « doGet » : « edit » et « update ».

```
      case "edit":    // add edit functionality
        actionEdit(req, resp);
        break;
      case "update":    // add update functionality
        actionUpdate(req, resp);
        break;
```

Pour appeler la méthode de modification d'une entrée, il nous faut ajouter un icône « d'édition » dans la colonne actions. Le procédé est identique à l'ajout de la croix rouge, mentionné précédemment.

Suite à la pression sur cette icône, nous générons un formulaire, identique à celui utilisé pour ajouter une entrée, à la différence près que nous bloquons l'édition du champ ssn, qui est la clé primaire de l'entrée dans la base de données. Nous récupérons initialement les données de l'entrée, que nous ajoutons aux champs du formulaire.

```
  // display edit form
  private void actionEdit(HttpServletRequest req, HttpServletResponse resp) throws IOException {
    final PrintWriter out = resp.getWriter();
    Connection conn = getDBConnection(out);
    try {
      String ssn = req.getParameter("ssn");
      if (ssn == null || ssn.equals("") || ssn.length() > 3
          || Integer.parseInt(ssn) == 0) {
        throw new Exception("SSN not valid");
      }

      String query = "SELECT * FROM employee WHERE ssn = ?";
      PreparedStatement stmt = conn.prepareStatement(query);
      stmt.setString(1, ssn);
      ResultSet rs = stmt.executeQuery();

      rs.next();

      out.write("<h2>Edit employee</h2>");
      out.write("<form action=\"?action=update\" method=\"POST\">");
      out.write("<table>");
      out.write("<tr>");
      out.write("<td><b>SSN</b> :</td>");
      out.write("<td><input  type=\"text\"  name=\"ssn\"  size=\"3\"  maxlength=\"3\"  value=\""  +
rs.getString(1) + "\"readonly/></td>");
      out.write("</tr>");
      out.write("<tr>");
      out.write("<td><b>Last name</b><sup>*</sup> :</td>");
      out.write("<td><input    type=\"text\"    name=\"lastname\"    size=\"20\"    value=\""    +
rs.getString(2) + "\"/></td>");
      out.write("</tr>");
      out.write("<tr>");
      out.write("<td>First name :</td>");
      out.write("<td><input    type=\"text\"    name=\"firstname\"    size=\"20\"    value=\""    +
rs.getString(3) + "\" /></td>");
      out.write("</tr>");
      out.write("<tr>");
      out.write("<td>E-mail :</td>");
      out.write("<td><input type=\"text\" name=\"email\" size=\"20\" value=\"" + rs.getString(4)
+ "\" /></td>");
      out.write("</tr>");
      out.write("<tr>");
      out.write("<td></td>");
      out.write("<td align=\"right\" bgcolor=\"#ccddff\">");
      out.write("<input type=\"submit\" name=\"submit\" value=\"Edit\" >");
      out.write("</td>");
      out.write("</tr>");
      out.write("</table>");
      out.write("<br>");
      out.write("<i>(*) mandatory</i>");
      out.write("</form>");

      stmt.close();

    } catch (Exception e) {
      out.write("<h2>Action report</h2>");
      out.write("<p>An error has occured !</p>");
      out.write("<p>" + e.getMessage() + "</p>");
    } finally {
      out.write("</body></html>");
      out.close();

      try {
        conn.close();
      } catch (SQLException e) {
        printSQLException(e, out);
      }
```

```
    }
}
```

Affichage de la liste initiale, avec une seule entrée.

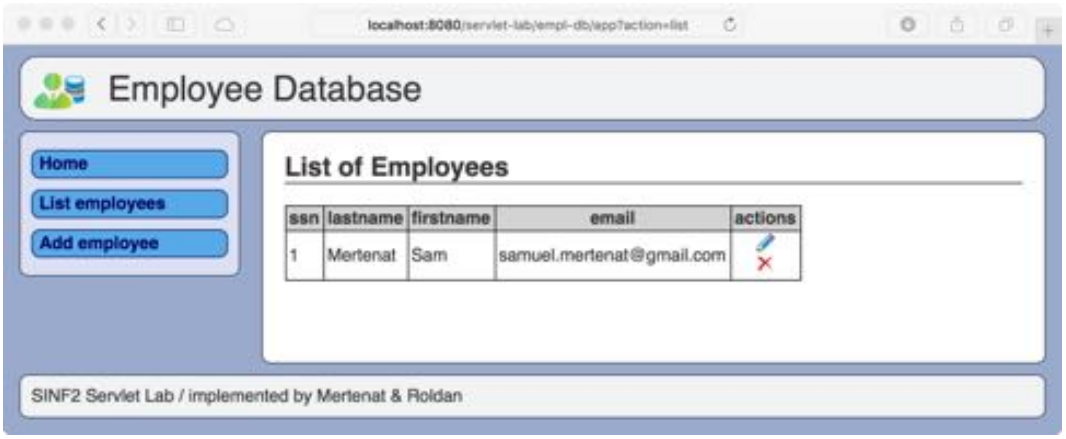

Figure 11: Liste des employés

Nous modifions ensuite le prénom et l'adresse E-Mail.

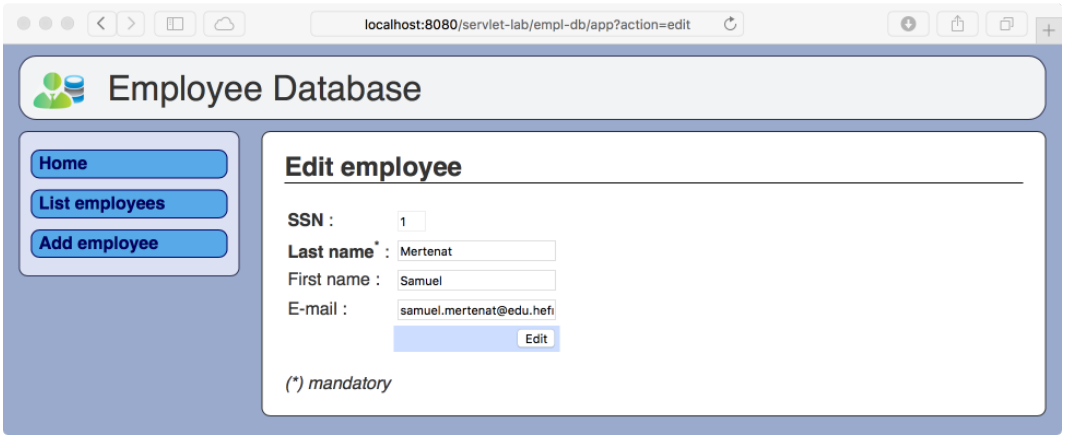

Figure 12: Edition d'une entrée

Nous constatons alors que l'entrée a bien été modifiée.

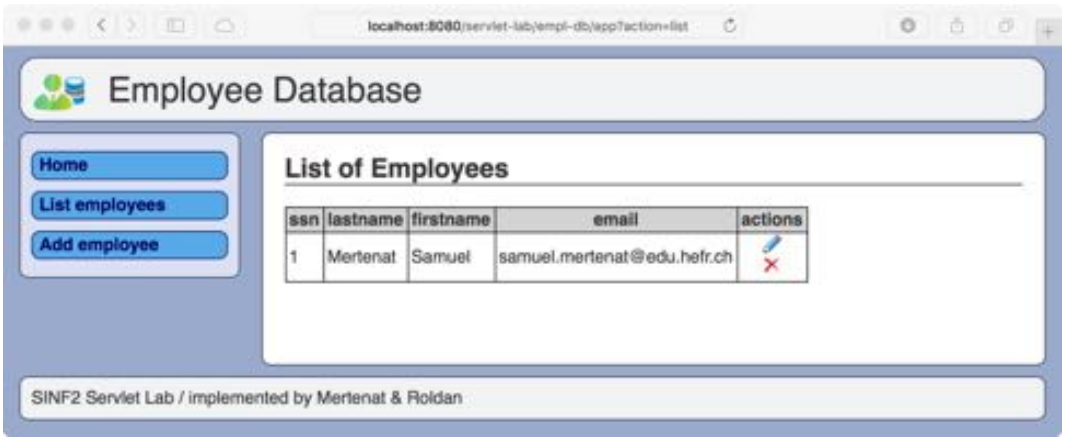

Figure 13: Entrée éditée

3.4.3 COOKIE

L'utilisation du Servlet « CookieServlet » nécessite une nouvelle modification du fichier « web.xml » :

```
<!-- CookieServlet -->
<servlet>
  <description>CookieServlet servlet</description>
  <display-name>CookieServlet Servlet</display-name>
  <servlet-name>CookieServlet</servlet-name>
  <servlet-class>servlet.CookieServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>CookieServlet</servlet-name>
  <url-pattern>/CookieServlet</url-pattern>
</servlet-mapping>
```

Le formulaire pour l'ajout d'un cookie, ajouté au début de la fonction « doGet » ; la méthode POST est utilisée.

```
// handle GET requests
public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {

  resp.setContentType("text/html");
  PrintWriter out = resp.getWriter();

  out.println("<html><head><title>Cookie</title></head><body>");
  out.println("<h2>Do you want to add a new Cookie ?</h2><br>");
  out.println("<form action=?action=add name=\"myform\" method=\"POST\">");
  out.println("Name : <input type=\"text\" name=\"name\" size=\"20\" />");
  out.println("Value : <input type=\"text\" name=\"value\" size=\"20\" />");
  out.println("Age : <input type=\"text\" name=\"age\" size=\"20\" />");
  out.println("<input type=\"Submit\" value=\"add\" name=\"Submit\">");
  out.println("</form>");

  // get action
  String requestedAction = "index";
  if (req.getParameter("action") != null) {
    requestedAction = req.getParameter("action");
  }

  // dispatch action
  switch (requestedAction) {
    case "add":
      actionAdd(req, resp);
      break;
    case "delete":
      actionDelete(req, resp);
      break;
    case "index":
    default:
      actionList(req, resp);
      break;
  }

  out.write("</body></html>");
  out.close();
} // end doGet()
```

La fonction pour afficher les cookies :

```
// retrieve and display cookies' list
private void actionList(HttpServletRequest req, HttpServletResponse resp)
```

```
      throws IOException {

    final PrintWriter out = resp.getWriter();
    out.println("<h2>List of Cookie(s)</h2>");
    out.println("<table border=\"1px solid black\">");
    out.println("<tr>");
    out.println("<th>Name</th>");
    out.println("<th>Value</th>");
    out.println("<th>Max Age</th>");
    out.println("<th>Remove</th>");
    out.println("</tr>");

    // retrieves the cookie(s)
    Cookie cookies[];
    cookies = req.getCookies();

    // generates the table
    for (int i = 0; i < cookies.length; i++) {
      out.println("<tr>");
      out.println("<td>" + cookies[i].getName() + "</td>");
      out.println("<td>" + cookies[i].getValue() + "</td>");
      out.println("<td>" + cookies[i].getMaxAge() + "</td>");
      out.println("<td>");
      out.println("<form action=?action=delete method=POST>");
      out.println("<input type=\"hidden\" name=\"name\" value=\""
          + cookies[i].getName() + "\" />");
      out.println("<input type=image src=empl-db/img/delete.png />");
      out.println("</form>");
      out.println("</td>");
      out.println("</tr>");
    }
    out.println("</table>");
  }
```

La fonction pour ajouter un cookie :

```
  private void actionAdd(HttpServletRequest req, HttpServletResponse resp)
      throws IOException {

    // if a new cookie is sent by POST
    String name = req.getParameter("name");
    String value = req.getParameter("value");
    String age = req.getParameter("age");
    if (name.equals("") == false && value.equals("") == false
        && age.equals("") == false) {
      Cookie cookie = new Cookie(name, value);
      cookie.setMaxAge(Integer.parseInt(age));
      cookie.setPath(req.getRequestURI());
      resp.addCookie(cookie);
    }
  }
```

La fonction pour supprimer un cookie :

```
  private void actionDelete(HttpServletRequest req, HttpServletResponse resp)
      throws IOException {
    // http://stackoverflow.com/questions/890935/how-do-you-remove-a-cookie-in-a-java-servlet
    String name = req.getParameter("name");
    if (name.equals("") == false) {
      Cookie cookies[] = req.getCookies();
      if (cookies != null)
        for (int i = 0; i < cookies.length; i++) {
          if (cookies[i].getName().equals(name)) {
            cookies[i].setValue("");
            cookies[i].setPath("/");
```

```
            cookies[i].setMaxAge(0);
            resp.addCookie(cookies[i]);
          }
        }
    }
  }
```

En vérifiant le fonctionnement du Servlet, nous avons pu constater quelques problèmes. L'ajout est d'un cookie est fonctionnel, cependant, lors de l'ajout d'un second, le premier est bizarrement plus disponible. Quant à la fonction de suppression, celle-ci fonctionne une fois sur deux. Le problème est peut-être dû au placement du formulaire d'ajout ou à l'utilisation de la méthode POST.

Figure 14: Liste des cookies

4 CONCLUSION

Ce travail pratique a été l'occasion d'appréhender le fonctionnement des Servlets 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. Cependant, nous avons rencontré quelques soucis à travailler avec les cookies.