Nancy-Université Master IST-IE — 2007–08 — UE 203d

Mise en œuvre de serveurs d'application — TD n° 2

1 Introduction

Dans ce TD, vous regarderez le contenu d'une application J2EE.

Ensuite, vous utiliserez les pages JSP pour créer une petite interface web pour une bibliothèque imaginaire.

2 Contenu d'une application

La semaine dernière vous avez installé une application sur le serveur. Cela s'est fait automatiquement quand vous avez cliqué sur l'icône pour publier. En pratique, eclipse s'est contenté de rassembler l'application dans un fichier `.ear` et de copier ce fichier dans le répertoire `deploy` de la configuration `default`.

1. À l'aide du navigateur graphique, allez dans le répertoire `/home_PC/burel/jboss-4.2.2.GA/server/default/deploy/` .
2. Double-cliquez sur votre application (a priori `TD1_`*votre_login*`.ear`).
   Quelle application s'ouvre-t-elle ?
   Les fichiers `.ear` sont en fait simplement des fichiers compressés de type `.zip`, mais avec un contenu particulier.
3. Décompressez le contenu de l'application dans un nouveau répertoire de votre dossier personnel.
   Normalement, le contenu est le suivant : un répertoire `META-INF` et un fichier `TD1_`*votre_login*`Web.war` .
   Le répertoire `META-INF` contient deux fichiers :
   – `application.xml` est un fichier XML qui décrit votre application. Son contenu devrait être similaire à ce qui suit :

```
<application xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
             http://java.sun.com/xml/ns/javaee/application_5.xsd"
              id="Application_ID" version="5">
  <module>
    <web>
```

```
      <web-uri>TD1_votre_loginWeb.war</web-uri>
      <context-root>TD1_votre_loginWeb</context-root>
    </web>
  </module>
</application>
```

Cela veut dire que votre application contient uniquement un module, qui est de type conteneur web, contenu dans le fichier `TD1_votre_loginWeb.war` et disponible à l'adresse `TD1_votre_loginWeb`. L'application aurait également pu contenir des modules de type conteneur EJB (`<ebj>...</ejb>`), de type connecteur vers des ressources non J2EE (`<connector>...</connector>`) ou encore de type application cliente (`<java>...</java>`).

– `MANIFEST.MF` contient quant à lui des informations concernant la création de l'application.

4. Ouvrez le fichier `TD1_`*votre_login*`Web.war` .

   Une fois encore, on s'aperçoit qu'il s'agit d'un fichier compressé. Celui-ci contient trois éléments :
   – le fichier `index.html` que vous aviez créé ;
   – un répertoire `META-INF` avec un fichier `MANIFEST.MF` dedans ;
   – un répertoire `WEB-INF` avec un fichier `web.xml` dedans.

   Ouvrez ce dernier. Il devrait ressembler à ceci :

```
<web-app xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                 http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         id="WebApp_ID" version="2.5">
  <display-name>TD1_votre_loginWeb</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

   Cela dit que votre module web s'appelle `TD1_votre_loginWeb` et que la page affichée par défaut est recherchée dans la liste `index.html`, `index.htm`, `index.jsp`, `default.html`, `default.htm` et `default.jsp` .

Quand nous aurons créé un module de type conteneur EJB, vous pourrez également regarder son contenu de cette façon.

3 Suppression d'une application

De la même façon que pour ajouter une application sur le serveur il suffit de la copier dans le répertoire `deploy`, pour la retirer il suffit de la supprimer dans le répertoire.

Néanmoins, si vous faites ainsi, eclipse ne sera pas au courant que vous avez retiré l'application, et il la remettra à nouveau lors la prochaine publication. Pour empêcher cela, il vaut mieux dire à eclipse de retirer l'application du serveur :

1. Dans l'onglet Servers, développez le serveur JBOSS le cas échéant.
2. Cliquez droit sur TD1_*votre_login*, puis sur Remove.
3. Répondez Ok.
4. Sélectionnez le serveur JBOSS, et cliquez sur la cinquième icône pour publier. Eclipse supprime le fichier `TD1_`*votre_login*`.ear` du répertoire `deploy`.

Comme vous n'avez pas beaucoup de place sur votre compte, il vous sera conseillé de faire ainsi avec les TDs dont vous n'aurez plus besoin, d'autant plus qu'en cas de dépassement de quota, vous ne penserez pas forcément aux fichiers qui ne se trouvent pas dans votre répertoire personnel.

4 Création de pages JSP

Dans la suite, vous allez réutiliser le projet de la semaine dernière. Comme vous venez de le retirer du serveur, il faut que vous le remettiez :

1. Cliquez droit sur le serveur JBOSS. Choisissez Add and Remove Projects.
2. Cliquez sur votre projet, puis sur Add >.
3. Cliquez sur Finish.

4.1 Première page

Nous allons maintenant remplacer la page par défaut statique `index.html` en une page dynamique `index.jsp` :

1. Dans le Project Explorer à gauche, retrouvez le fichier index.html. Cliquez droit dessus, choisissez Delete. Répondez Yes.
2. Toujours dans le Project Explorer, cliquez droit sur TD1_*votre_login*Web.
3. Choisissez New ▶ JSP.
4. Dans File name entrez `index.jsp`. Cliquez sur Next.
5. Choisissez le template New JSP File (xhtml). Cliquez sur Finish.
   Remarquez la ligne
   ```
   <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
            pageEncoding="ISO-8859-1"%>
   ```
   Nous avons bien affaire à une page JSP pour le langage java.
6. Remplacez `Insert title here` par `Ma première page dynamique`.
7. Après `<body>`, ajoutez `<p>Le contenu de cette page est malgré tout statique.</p>`
8. Enregistrez (icône disquette).
9. Publiez (5e icône de l'onglet Servers).
10. Allez dans firefox et visitez l'adresse `http://localhost:8080/TD1_`*votre_login*`Web/` .

4.2 Ajout d'instructions java

Nous allons rajouter quelques instructions java dans notre page :

1. Supprimez `<p>Le contenu de cette page est malgré tout statique.</p>`
2. À la place, inscrivez le texte suivant :

```
<h2>Création d'une variable et affichage</h2>
  <% int x = 2; %>
  <p><var>x</var> vaut initialement <%= x %>.</p>
<h2>Modification</h2>
  <% x = x + 1; %>
  <p><var>x</var> vaut maintenant <%= x %>.</p>
```

3. Enregistrez, publiez, vérifiez dans firefox.

4.3 Inclusion de page

Nous allons rajouter un entête et un pied de page qui seront communs à toutes les pages du site. Comme on veut améliorer la maintenance du site, on ne veut pas avoir à modifier chaque page du site sitôt que l'on veut modifier l'entête. Par conséquent, on va écrire ces entête et pied de page dans des fichiers séparés, et on les inclura dans les pages avec JSP.

1. Dans le Project Explorer, cliquez droit sur TD1_*votre_login*Web, créez un nouveau fichier JSP.
2. Appelez ce fichier `entete.jspf`. Le `f` indique qu'il s'agit uniquement d'un fragment de page. Cliquez sur Finish pour le créer.
3. Supprimez le contenu de `<!DOCTYPE` à `</html>`.
4. À la place, entrez `<h1>Site de la bibliothèque imaginaire</h1>`
5. Sauvegardez.
6. Créez un nouveau fichier JSP nommé `piedDePage.jspf` .
7. Supprimez le contenu de `<!DOCTYPE` à `</html>`.
8. Entrez à la place :

```
<%@page import="java.util.Date"%>
<hr/>
<p>Nous sommes le
  <% Date d = new Date();
     out.print(d.toLocaleString()); %>
</p>
```

   Quelques commentaires sur ces lignes :
   - La première sert à importer des classes java existantes, ici la classe standard Date qui représente une date.
   - On ajoute ensuite une barre horizontale, puis un paragraphe.
   - Dans le code JSP entre `<%` ... `%>`, on crée d'abord un nouvel objet de la classe Date, qui est initialisé à la date courante.

– Puis on affiche (`out.print`) une version en français (`toLocaleString()`) de la date.

9. Enregistrez et revenez dans le fichier `index.jsp` .
10. Juste après `<body>`, ajoutez `<%@ include file="entete.jspf" %>`
    Juste avant `</body>`, ajoutez `<%@ include file="piedDePage.jspf" %>`
11. Enregistrez tout, republiez, puis regardez le résultat dans firefox.
    Remarquez que vous avez vraiment une page dynamique : si vous rafraîchissez la page, la date change.

4.4 Interaction avec l'utilisateur

Nous allons rajouter la possibilité à l'utilisateur de demander la liste des livres qu'il a empruntés, en entrant son nom.

1. Dans `index.jsp`, supprimez le contenu entre `<h2>Création` et le dernier `</p>` .
2. Ajoutez le code suivant :

```
<h2>Consultation des livres empruntés</h2>
<form action="emprunts.jsp">
  <p>Nom de l'utilisateur <input name="utilisateur"/></p>
  <button>Rechercher</button>
</form>
```

   Nous créons donc un formulaire qui demande d'entrer un nom d'utilisateur puis qui appelle la page `emprunts.jsp` avec ce nom en paramètre. Il nous faut donc créer cette page.
3. Créer une nouvelle page JSP nommée `emprunts.jsp` . Donnez lui un titre adéquat.
4. Juste après `<body>`, ajoutez `<%@ include file="entete.jspf" %>`
   Juste avant `</body>`, ajoutez `<%@ include file="piedDePage.jspf" %>`
5. Il nous faut récupérer le paramètre `utilisateur` . Pour cela, il faut utiliser le code java `request.getParameter("utilisateur")` . Ajoutez le code `<h2>Livres empruntés par <%=request.getParameter("utilisateur")%></h2>`
6. Enregistrez, republiez. Testez en entrant comme nom d'utilisateur `personne`. Si tout va bien, la page `http://localhost:8080/TD1_`*votre_login*`Web/emprunts.jsp?utilisateur=personne` est affichée.

4.5 Affichage conditionnel

1. Essayez d'afficher la page `http://localhost:8080/TD1_`*votre_login*`Web/emprunts.jsp` , c'est-à-dire sans passer de paramètre `utilisateur`.
   Ce n'est pas exactement la page que nous souhaitons. Au lieu de cela, nous devrions afficher autre chose quand le nom de l'utilisateur n'est pas donné.
2. Remplacer la ligne `<h2> ... </h2>` par le code

```
<% String util = request.getParameter("utilisateur");
   if (util == null) { %>
     <!-- Partie affichée quand l'utilisateur n'est pas donné -->
     <p> Veuillez entrer un nom d'utilisateur. </p>
<% } else { %>
     <!-- Partie affichée quand l'utilisateur est donné -->
     <h2>Livres empruntés par <%=util%></h2>
<% }; // fin du if %>
```

On remarque qu'on utilise le `if` de java pour distinguer les parties à afficher.

3. Enregistrez, publiez, testez sous firefox. En particulier, afficher les pages
   - `http://localhost:8080/TD1_`*votre_login*`Web/emprunts.jsp`
   - `http://localhost:8080/TD1_`*votre_login*`Web/emprunts.jsp?utilisateur=`
   - `http://localhost:8080/TD1_`*votre_login*`Web/emprunts.jsp?utilisateur=personne`

4.6 Inclusion et forwarding

Plutôt que d'afficher uniquement « Veuillez entrer un nom d'utilisateur. », on aimerait plutôt afficher le formulaire de la page d'accueil.

1. Remplacez `<p> Veuillez entrer un nom d'utilisateur. </p>` par `<%@ include file="index.jsp" %>` .
2. Enregistrer, publiez, affichez la page `http://localhost:8080/TD1_`*votre_login*`Web/emprunts.jsp` .

   C'est n'est pas tout à fait ce que nous espérions, car le contenu entier de la page d'accueil a été inséré. Une solution serait de mettre dans un fichier JSP à part le formulaire, et d'inclure ce fichier à la fois dans `index.jsp` et `emprunts.jsp`. Toutefois, le mieux est de forwarder la page des emprunts sur la page d'accueil quand celle-ci est appelée sans utilisateur.
3. Remplacez `<%@ include file="index.jsp" %>` par `<jsp:forward page="index.jsp"/>`
4. Enregistrer, publiez, affichez la page `http://localhost:8080/TD1_`*votre_login*`Web/emprunts.jsp` . On retombe sur la page d'accueil.

4.7 Tableau créé dynamiquement

Nous allons rajouter un tableau dans `emprunts.jsp` qui sera créé dynamiquement en fonction du contenu de la liste de livres empruntés. Cette liste sera en fait créée par la partie métier de l'application, que nous n'avons pas écrite pour l'instant. Nous ajouterons donc une liste créée à la main en attendant.

1. Tout d'abord, nous allons avoir besoin de classes java Vector et Iterator du package java.util. Il faut donc les importer dans la page, par exemple en ajoutant `<%@page import="java.util.*"%>` après `<!DOCTYPE ... .dtd">`
2. Après `</h2>`, rajoutez

```
<% // Vecteur de livres créés a la main
   Vector livres = new Vector ();
   livres.add("Les Frères Karamazov");
   livres.add("Fondation");
   livres.add("Du côté de chez Swann"); %>
  <table>
    <tr> <th> Titre </th> </tr>
    <% // on crée un itérateur sur les livres
       Iterator iterator = livres.iterator();
       // on affiche chaque élément dans un ligne
       while (iterator.hasNext()) {
         Object livre = iterator.next(); %>
    <tr> <td> <%=livre %> </td> </tr>
    <% }; // fin du while %>
  </table>
```

3. Enregistrer, publiez, testez.
4. On va rajouter une deuxième colonne, contenant la date de retour (qui est indisponible pour l'instant et sera fournie par la couche métier), ainsi qu'une troisième contenant un lien pour rendre le livre.
   (a) Après `<th> Titre </th>` rajoutez `<th> Date de retour </th>`
   (b) Après `<td> <%=livre %> </td>` rajoutez
   ```
   <td> Inconnue </td>
   <td> <a href="retour.jsp?livre=<%=livre%>"> Retour </a> </td>
   ```
5. Créez la page JSP `retour.jsp` . Elle devra contenir l'entête et le pied de page, et devra afficher « Le livre TITRE a bien été rendu. » avec le vrai titre à la place de « TITRE ».
   Indice : pour récupérer le titre du livre passé en argument, il faut utiliser le code java `request.getParameter("livre")`
   Quand nous aurons écrit la logique métier, cette page devra bien entendu appeler la requête de la partie métier pour le retour du livre et afficher un résultat en conséquence.
6. Enregistrer, publiez, testez.

S'il vous reste du temps, vous pouvez rajoutez dans la page d'accueil un deuxième formulaire qui permet de faire une recherche sur le titre des ouvrages. La page de résultat affichera un tableau avec le titre du livre, la date de retour s'il est déjà emprunté ou un formulaire pour l'emprunter dans le cas contraire. Vous pourrez vous inspirer de l'application disponible à l'adresse `http://localhost:8080/TD1_burel0091Web/` . (Les utilisateurs disponibles pour faire des tests dans cette application sont Guillaume, Noelle et Herve.)