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

Mise en œuvre de serveurs d'application — TD n° 4

1 Introduction

Dans ce TD, vous implanterez une application cliente, vous étendrez la logique métier sans modifier les clients, et vous ajouterez de la sécurité à votre application.

À la fin du TD n'oubliez pas de supprimer les fichiers que vous avez mis de le répertoire `deploy` du serveur d'application (voir une commande possible pour cela à la fin de cet énoncé).

2 Application cliente

Pour accéder à l'application sur le serveur, on peut soit utiliser un navigateur qui accède à la couche web, soit utiliser une application cliente, c'est-à-dire un programme spécifique qui va faire des requêtes à l'application sur le serveur. Par exemple, un distributeur bancaire automatique ne va pas utiliser la couche web, plus vulnérable, mais une application cliente.

Pour faire une application cliente utile, il faudrait écrire une interface graphique. Ceci n'est pas le but de ces TDs. On se contera donc d'une application qui fonctionnera sans interaction avec l'utilisateur.

1. Dans le Project Explorer, cliquez droit sur TD1_*votre_login*, choisissez New ▶ Application Client Project
2. Dans Project Name: indiquez `TD1_`*votre_login*`Client`
3. Cliquez sur Finish.
4. Dans le Project Explorer, cliquez droit sur TD1_*votre_login*Client, choisissez Properties.
5. Cliquez sur Java Build Path, puis sur l'onglet Project. Cliquez sur Add.
6. Sélectionnez le projet client associé au conteneur d'EJB (normalement TD1_*votre_login*EJBClient). Ceci permettra d'utiliser les EJB dans l'application cliente. Cliquez sur Ok.
7. Cliquez sur Ok.

8. À l'aide du Project Explorer, ouvrez le fichier Main.java dans TD1_*votre_login*Client ▶ appClientModule ▶ (default package).
9. Au début du fichier, ajoutez

```
import biblio.*;
import java.io.*;
import java.util.Properties;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.naming.NamingException;
```

   Ces classes java seront utilisées par la suite.

Vous allez modifier la méthode `public static void main(String[] args)` qui est celle appelée quand l'application est lancée.

10. Tout d'abord nous allons indiquer comment accéder au serveur d'application. Ceci ce fait à l'aide d'un fichier de propriétés que nous allons charger.
    (a) Après `// TODO Auto-generated method stub`, ajoutez

```
try {
  Properties p = new Properties();
  p.load(new java.io.FileInputStream("biblio.props"));
```

    (b) Cliquez droit sur TD1_*votre_login*Client, choisissez New ▶ File.
    (c) Dans File Name: indiquez `biblio.props`, cliquez sur Finish.
    (d) Utilisez le Project Explorer pour ouvrir le fichier nouvellement créé.
    (e) Remplissez le avec les lignes suivantes :

```
java.naming.factory.initial=\
  org.jboss.security.jndi.JndiLoginInitialContextFactory
java.naming.provider.url=localhost:1099
java.naming.factory.url.pkgs=org.jboss.naming
```

11. Revenez dans le fichier `Main.java`. Ajoutez le code suivant, qui récupère les interfaces distantes maison, crée un livre et un utilisateur puis fait emprunter le livre par l'utilisateur, à la suite de ce que vous avez ajouté précédemment. (Remplacez `votre nom` et `votre adresse mail` par ce qu'il faut.)

```
// Récupération des interfaces maison
Utilisateur_votre_loginHome uHome =
    Utilisateur_votre_loginUtil.getHome(p);
Livre_votre_loginHome lHome = Livre_votre_loginUtil.getHome(p);
Admin_votre_loginHome aHome = Admin_votre_loginUtil.getHome(p);
// Nouvel utilisateur
uHome.create("votre nom", "votre adresse mail", "etudiant");
// Nouveau livre
Livre_votre_login steppes = lHome.create("Le loup des steppes");
```

```
// Nouveau bean d'administration
Admin_votre_login admin = aHome.create();
// Emprunt
admin.emprunte("votre nom", steppes.getCote());
System.out.println("Le livre " + steppes.getTitre() +
     " est emprunte par " + steppes.getEmprunteur() + ".");
```

12. Entrez à la suite le suivant qui permet de récupérer les différentes exceptions (erreurs) qui pourrait intervenir lors du programme :

```
} catch (RemoteException e) {
  System.err.println("Problème de connexion :");
  System.err.println("  " + e.getMessage());
} catch (IOException e) {
  System.err.println("Impossible de trouver le fichier de propriétés :");
  System.err.println("  " + e.getMessage());
} catch (NamingException e) {
  System.err.println("Impossible de se connecter à l'application :");
  System.err.println("  " + e.getMessage());
} catch (CreateException e) {
  System.err.println("Problème de création des beans :");
  System.err.println("  " + e.getMessage());
} catch (Exception e) {
  System.err.println("Impossible d'emprunter le livre :");
  System.err.println("  " + e.getMessage());
}
```

13. Dans le Project Explorer, cliquez droit sur Main.java, choisissez Run as ▶ Java application. L'onglet console passe devant et affiche ce que le programme sort. Vérifiez l'absence de message d'erreur (écrit en rouge).

3 Modification de la logique métier

Vous allez maintenant modifier la logique métier pour envoyer un mail à l'utilisateur quand il emprunte un livre, en lui indiquant la date de retour. Ceci doit se faire sans que l'on ait à modifier la couche web ni les applications clientes.

Tout d'abord vous allez créer un bean contrôlé par message dont le rôle sera d'envoyer le mail en question, de façon asynchrone pour ne pas surcharger le reste de l'application.

1. Cliquez droit sur TD1_*votre_login*EJB.
2. Choisissez New ▶ XDoclet Enterprise JavaBean.
3. Cochez Message Driven Bean. Cliquez sur Next.
4. Dans Java package, indiquez `biblio`

5. Dans Classe name, indiquez `Confirme`*votre_login*`Bean`. Cliquez sur Next.
6. Cliquez sur Finish.

Le nouveau bean se trouve alors dans le Project Explorer à gauche à la position TD1_*votre_login*EJB ▶ ejbModule ▶ biblio ▶ Confirme_*votre_login*Bean.java. Double-cliquez dessus pour l'ouvrir.

7. Remplacez `* destination-jndi-name="Confime_votre_login"` par :

```
*     destination-jndi-name="queue/Confirme_votre_login"
*     connection-factory-jndi-name="java:/JmsXA"
*
* @jboss.destination-jndi-name name="Confirme_votre_login"
```

8. Il faut maintenant implanter la méthode `onMessage` qui doit envoyer le mail. Après

```
public void onMessage(javax.jms.Message message) {
    // begin-user-code
```

entrez le code suivant :

```
// vérification du type de message
if (message instanceof MapMessage) {
  MapMessage msg = (MapMessage) message;
  try {
    // création d'une session d'envoi de mail
    Context initial = new InitialContext();
    Session session =
    (Session) initial.lookup("java:/Mail");
    // création d'un nouveau mail
    Message mail = new MimeMessage(session);
    // Expéditeur par défaut
    mail.setFrom();
    // Destinataire
    mail.setRecipients(Message.RecipientType.TO,
      InternetAddress.parse(msg.getString("adresse"),false));
    String titre = msg.getString("titre");
    // Sujet
    mail.setSubject("Emprunt de " + titre);
    // Contenu
    String contenu = "Bonjour " + msg.getString("nom") + ",\n\n"
      + "Ceci est un mail pour confirmer que vous avez emprunté\n"
      + "le livre intitulé " + titre + ".\n"
     + "La date de retour est le " + msg.getString("date") + ".\n\n"
      + "Cordialement.\n-- \nLa bibliothèque imaginaire\n";
    mail.setText(contenu);
```

```
    mail.setHeader("X-Mailer", "JBossAS");
    mail.setSentDate(new Date());
    // Envoi du mail
    Transport.send(mail);
    System.out.println("Mail envoyé.");
// Gestion des erreurs
  } catch (Exception e) {
    System.err.println("Problème avec le message :\n" + e);
  }
} else { System.err.println("Type de message non reconnu"); }
```

Il faut maintenant faire appel au bean contrôlé par message dans la méthode `emprunte` du bean `Admin_votre_loginBean`

9. Retournez dans le fichier `Admin_`*votre_login*`.java`, créé lors du TD n° 3.
10. Dans la méthode `emprunte`, ajoutez à la suite de `l.setEmprunteur(u);` le code suivant :

```
// Connexion à la file liée au bean contrôlé par message
Queue q = Confirme_votre_loginUtil.getQueue();
QueueConnection qc = Confirme_votre_loginUtil.getQueueConnection();
QueueSession qs = qc.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
QueueSender sender = qs.createSender(q);
// Création d'un message
MapMessage msg = qs.createMapMessage();
msg.setString("adresse", u.getAdresse());
msg.setString("titre", l.getTitre());
msg.setString("date", l.getRetour().toLocaleString());
msg.setString("nom", u.getNom());
// Envoi du message
sender.send(msg);
qs.close();
```

11. Ajoutez `import javax.jms.*;` après `package biblio;` en début de fichier.
12. Sauvegardez tout, compilez deux fois, publiez sur le serveur. Lancez l'application cliente pour tester (cliquez droit sur Main.java, choisissez Run as ▶ Java application). Si tout va bien, vous devriez recevoir un mail vous informant de votre emprunt.

4 Ajout de sécurité

Pour l'instant, n'importe qui peut accède au serveur d'application et le modifier. Nous allons introduire deux niveaux de sécurité, `administrateur` et `usager`. Le premier sera autorisé à créer des livres et des utilisateurs, tandis que le second pourra uniquement consulter les livres empruntés.

1. Dans `Utilisateur`*votre_login*`Bean.java`, ajoutez au-dessus de `public java.lang.String ejbCreate(String nom, String adresse, String statut)` juste avant `* <!-- end-xdoclet-definition -->` le code suivant :

   ```
   *
   * @ejb.permission role-name="administrateur"
   ```

2. À l'aide du Project Explorer, cliquez droit sur META-INF dans TD1_*votre_login*EJB ▶ ejbModule. Choisissez New ▶ File.
3. Nommer le fichier `jboss-security.xml`
4. Entrez le contenu suivant dedans :

   ```
   <security-domain>java:/jaas/securite</security-domain>
   ```

5. Sauvegardez, compilez deux fois, publiez.
6. Allez dans firefox à l'adresse `http://localhost:8080/TD1_`*votre_login*`Web`. et cliquez sur le bouton d'initialisation. Quel message d'erreur s'affiche-t-il ?
7. Lancez l'application cliente. Quel message d'erreur s'affiche-t-il ?

Il faut maintenant créer deux fichier qui contiendront l'ensemble des utilisateurs de l'application, leur mot de passe et leurs rôles.

8. Dans le Project explorer, cliquez droit sur ejbModule, faites New ▶ Other.
9. Choisissez General ▶ File. Cliquez sur Next.
10. Dans File name: entrez `users.properties`. Cliquez sur Finish.
11. Entrez le texte suivant, qui indique le nom des utilisateurs avec leur mot de passe :

    ```
    Riri=coincoin
    Fifi=quackquack
    Loulou=quakquak
    ```

12. Créez de la même façon un fichier `roles.properties` avec le contenu suivant :

    ```
    Riri=administrateur,usager
    Fifi=usager
    Loulou=administrateur
    ```

13. Sauvegardez, compilez deux fois, publiez.

L'application cliente et la couche web doivent s'authentifier pour accéder aux ressources.

14. Rajouter le code suivant dans le fichier `biblio.props` de l'application cliente, pour s'authentifier en tant que Riri :

    ```
    java.naming.security.principal=Riri
    java.naming.security.credentials=coincoin
    ```

15. Testez à nouveau l'application cliente. Vous devriez recevoir un mail.
16. À l'aide du Project Explorer, ouvrez le fichier web.xml dans TD1_*votre_login*Web ▶ WebContent ▶ WEB-INF.
17. Ajoutez le texte suivant à la suite de `</welcome-file-list>` :

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>initialisation</web-resource-name>
    <url-pattern>/init.jsp</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>administrateur</role-name>
  </auth-constraint>
</security-constraint>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>accueil</web-resource-name>
    <url-pattern>/index.jsp</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>usager</role-name>
  </auth-constraint>
</security-constraint>
<security-role>
  <role-name>usager</role-name>
</security-role>
<security-role>
  <role-name>administrateur</role-name>
</security-role>
<login-config>
  <auth-method>BASIC</auth-method>
</login-config>
```

18. Créez un fichier `jboss-web.xml` dans WEB-INF.
19. Remplissez-le avec la ligne suivante :

```
<security-domain>java:/jaas/securite</security-domain>
```

20. Sauvegardez, compilez deux fois, publiez.
21. Dans firefox, allez sur la page `http://localhost:8080/TD1_`*votre_login*`Web/` . On vous demande de vous authentifier. Connectez-vous en tant que `Loulou` (mot de passe `quakquak`). Pourquoi une page d'erreur s'affiche-t-elle ? (Remarque : Loulou peut néanmoins accéder à la page d'initialisation directement en allant à l'adresse `http://localhost:8080/TD1_`*votre_login*`Web/init.jsp`)

22. Quittez firefox, et redémarrez-le. Allez sur la page `http://localhost:8080/TD1_`*votre_login*`Web/` et connectez-vous en tant que `Riri` (mot de passe `coincoin`). Vérifiez que vous pouvez accéder à toutes les requêtes (Riri est à la fois administrateur et usager).

L'interface pour se connecter n'est pas personnalisée, il s'agit du mécanisme d'authentification basique de HTTP. Vous allez améliorer cela.

23. Dans web.xml, changez `<auth-method>BASIC</auth-method>` en

```
<auth-method>FORM</auth-method>
<form-login-config>
  <form-login-page>/login.jsp</form-login-page>
  <form-error-page>/error.jsp</form-error-page>
</form-login-config>
```

24. Créer un nouveau fichier JSP nommé `login.jsp` dont le contenu de `<body>` sera :

```
<%@ include file="entete.jspf" %>
<h2>Authentifiez-vous</h2>
<form method="post" action="j_security_check">
<p>Nom : <input type="text" name="j_username"/></p>
<p>Mot de passe : <input type="password" name="j_password"/></p>
<button>Connexion</button>
</form>
<%@ include file="piedDePage.jspf" %>
```

25. Vous pouvez éventuellement aussi créer un fichier JSP `error.jsp` qui indiquera que l'authentification a échoué.
26. Sauvegardez tout, compilez deux fois, publiez. Vérifiez le bon fonctionnement dans firefox.
27. S'il vous reste du temps, rajoutez les annotations XDoclets du type `* @ejb.permission role-name="administrateur"` ou `* @ejb.permission role-name="usager"` de façon adéquate pour toutes les méthodes des EJB Livre_*votre_login*, Utilisateur_*votre_login* et Admin_*votre_login*.

5 Suppression des fichiers sur le serveur

À la fin du TD, n'oubliez pas d'enlever les fichiers que vous avez mis sur le serveur d'application (ils comptent dans votre quota). Pour cela, le plus simple est d'utiliser la commande suivante :

```
find /home_PC/burel/jboss-4.2.2.GA/server/default/deploy/ \
  -user $(id --user) -ok rm '{}' ';'
```

en répondant `y` aux questions posées.