TP1 - Entreprise Java Beans

Réseaux Middlewares - Jules Chevalier

1 Configuration Minimale

Pour ce TP, vous aurez besoin de outils suivants :
- Un JDK installé (le JRE est insuffisant)
- Un IDE édition Java EE (Eclipse Java EE sera utilisé pour les explications)
- Un serveur d'application, ici Glassfish 3.1

2 Mise en place

2.1 Installer Glassfish

Tout d'abord, les EJB ont besoin d'un serveur d'application pour être déployés. L'un des plus utilisé est Glassfish. Les instructions seront données pour Glassfish, vous pouvez utiliser un serveur différent, à condition d'être autonome en ce qui concerne son utilisation. De même, nous verrons comment développer et déployer ces EJB dans Eclipse. Vous pouvez utiliser un autre IDE aux mêmes conditions.

Pour installer un Glassfish dans Eclipse :
- Installer le plugin Glassfish
  - **Help -> Eclipse Marketplace**
  - Rechercher glassfish
  - Installer **Glassfish Java EE Application Server Plugin for Eclipse**
- Installer Glassfish
  - **Windows -> Preferences**
  - **Server -> Runtime Environnement**
  - **Add**
  - Glassfish 3.1
  - Choisir le dossier où installer glassfish
  - **Install Server**
  - Dans l'onglet **Servers** (en bas), créer un nouveau serveur Glassfish 3.1

2.2 Créer le projet EJB

Vous devez créer un projet EJB. Dans Eclipse, dans **File -> New -> Other**, choisissez **EJB Project**.

Dans les propriétés du projet, ouvrez **Java Build Path**, onglet **Librairies**. Si **JRE System Library** ne correspond pas à un JDK, **Edit** permet de changer la version de Java utilisée dans le projet. Si le JDK n'apparaît pas dans Alternate JRE, ajoutez le dans **Installed JREs**.

Enfin, certaines librairies sont nécessaires à l'exécution du projet. Toujours dans **Java Build Path**, **Add External JARs** permet d'ajouter des librairies. Ajoutez **appserv-rt.jar** et **javaee.jar**, présents dans le dossier d'installation de glassfish **glassfish3/glassfish/lib**.

Créez enfin deux packages, un package **ejbs** qui contiendra les EJB et leurs interfaces, et un package **client** qui contiendra les applications clientes.

3 Écrire un EJB

3.1 Stateless

La première partie consiste à développer des EJB mettant à disposition les même méthodes que le précédent TP, à savoir **hello()** et **echo(String s)**.

Tout d'abord, il faut écrire l'interface distante. C'est une interface simple annotée **@Remote**. Par exemple :

1– SimpleRemote.java

```
1  package ejbs;
2  import javax.ejb.Remote;
3
4  @Remote
5  public interface SimpleRemote {
6     public String hello();
7     public String echo(String s);
8  }
```

Ensuite, il faut implémenter le Bean en lui-même. Il est annoté **Stateless**, et implémente l'interface distante. Les paramètres de **Stateless** permettent de définir le nom de l'EJB (name) et le nom sous lequel il est accessible à distance (mappedName). Par exemple :

2– SimpleBean.java

```
1  package ejbs;
2  import javax.ejb.Stateless;
3
4  @Stateless(name="SimpleEJB", mappedName="SimpleBean")
5  public class SimpleBean implements SimpleRemote{
6     @Override
7     public String hello(){
8        return "Bonjour";
9     }
10    @Override
11    public String echo(String s){
12       return "Echo : "+s;
13    }
14 }
```

Pour déployer vos EJBs, vous pouvez lancer le projet sur le serveur (**clic-droit -> Run as... -> Run on Server**), ou manuellement en exportant le projet au format .jar dans le dossier **glassfish3/glassfish/domains/domain1/autodeploy**.

Il ne manque plus que le client qui récupère et utilise l'EJB. Pour cela, il récupère l'EJB avec **ctx.lookup("SimpleRemote")**, puis utilise ses méthodes.

Voici un exemple de client :

3– Client.java

```
1   public static void main(String[] args) {
2     try {
3       InitialContext ctx = new InitialContext();
4       EchoRemote bean = (EchoRemote) ctx.lookup("SimpleRemote");
5       System.out.println(bean.hello());
6       System.out.println(bean.echo("Billy Bob"));
7     } catch (Exception e) {
8       e.printStackTrace();
9     }
10  }
```

3.2 Statefull

Votre EJB doit maintenant être Stateful, c'est-à-dire qu'il conserve des données d'un appel à l'autre. Ajoutez une méthode **login** qui permet de sauvegarder le nom de l'utilisateur dans une variable de l'EJB. Maintenant, la méthode **hello** doit utiliser le nom de l'utilisateur, s'il s'est déjà loggué. L'interface doit elle aussi être mise à jour. N'oubliez pas de changer l'annotation du Bean en **Stateful**

4– SimpleRemote.java

```
1  package ejbs;
2  import javax.ejb.Remote;
3
4  @Remote
5  public interface SimpleRemote {
6     public void login(String name);
7     public String hello();
8     public String echo(String s);
9  }
```

5– SimpleBean.java

```
1  package ejbs;
2  import javax.ejb.Stateful;
3
4  @Stateful(name="SimpleEJB", mappedName="SimpleBean")
5  public class SimpleBean implements SimpleRemote{
6     String name=null;
7     @Override
8     public void login(String name){
9        this.name=name;
10    }
11    @Override
12    public String hello(){
13       if(name==null)
14          return "Bonjour";
15       return "Bonjour "+name;
16    }
17    @Override
18    public String echo(String s){
19       return "Echo : "+s;
20    }
21 }
```

3.3 Aller plus loin

Les EJB ne s'utilisent pas qu'à distance. Pour utiliser un EJB dans une autre EJB, vous devez utiliser **L'injection de dépendance**. Un EJB n'est jamais instancié directement par le développeur. C'est le conteneur d'EJB qui est chargé de gérer le cycle de vie des EJB. Pour utiliser un EJB localement, on le déclare comme une variable non initialisée, annotée **@EJB**. Lorsque la variable correspondant à l'EJB sera utilisé, le conteneur se chargera de relier une instance à la variable.

Vous devez donc créer un nouvel EJB avec une interface locale (annotée **@Local**), qui sera utilisée par le premier EJB. Par exemple, plutôt que de dire bonjour toujours de la même façon, SimpleBean peut demander à un autre EJB de lui donner aléatoirement un "bonjour".

Vous devez donc avoir une interface RandomLocal, et un bean RandomBean.

6– RandomLocal.java

```java
package ejbs;
import javax.ejb.Local;

@Local
public interface RandomRemote {
   public String randomHello();
}
```

7– SimpleBean.java

```java
package ejbs;
import javax.ejb.Stateless;

@Stateless
public class RandomBean implements RandomLocal {
   @Override
   public String randomHello() {
      String[] hellos = {"Bonjour","Salut","Hello","Yo","Hey"};
      int r = (int)(Math.random()*(hellos.length));
      return hellos[r];
   }
}
```

Pour l'utiliser dans le premier EJB :

8– SimpleBean.java

```java
package ejbs;
import javax.ejb.Stateful;

@Stateful(name="SimpleEJB", mappedName="SimpleBean")
public class SimpleBean implements SimpleRemote{
   String name=null;
   @EJB
   private RandomLocal rnd;

   @Override
   public void login(String name){
      this.name=name;
   }
   @Override
   public String hello(){
      String hi = rnd.randomHello();
      if(name==null)
         return hello;
      return hi+" "+name;
   }
   @Override
   public String echo(String s){
      return "Echo : "+s;
   }
}
```