Université Bretagne Sud – UFR SSI

Master Informatique 2e année – Parcours SAIM – Année 2020/2021

# INF 4001 – Intergiciels répartis et mobiles

# JBoss - EJB 3.0 Session

## 1 Installation et environnement

JBoss Application Server est un serveur d'applications JEE libre. Il supporte notamment les EJB 3.0. Vous trouverez la documentation relative aux EJB 3 de JBoss à l'URL `http://ejb3.jboss.org/docs/`.

En TP, vous utiliserez la version 5.1.0 de JBoss, avec le JDK 7 issu du forum. Les opérations suivantes sont à effectuer préalablement :

- De-zippez l'archive `jboss-5.1.0.GA-jdk6.zip` (fournie dans le forum) dans votre répertoire personnel et faites en sorte que la variable JBOSS_HOME contienne le répertoire d'installation créé (typiquement `$HOME/jboss-5.1.0.GA`)
- Faites en sorte que les fichiers jar suivant soient dans votre CLASSPATH :
  - `$JBOSS_HOME/client/jbossall-client.jar`
  - `$JBOSS_HOME/common/lib/jboss-ejb3-ext-api.jar` (pour la configuration du cache)

La plate-forme JBoss est lancée grâce à la commande `$JBOSS_HOME/bin/run.sh` et est arrêtée avec la séquence de touches CTRL-C ou avec la commande `$JBOSS_HOME/bin/shutdown.sh`. On peut vérifier que la plate-forme est bien lancée en consultant dans un navigateur web la page renvoyée par le serveur HTTP intégré, à l'URL `http://localhost:8080`.

Le déploiement d'un EJB se fait par copie de l'archive ejb-jar (fichier .jar) dans le répertoire `$JBOSS_HOME/server/defaults/deploy`. JBoss détecte automatiquement les changements opérés sur ce répertoire. L'ajout d'un fichier `.jar` sert donc de déploiement de bean et la suppression du fichier fait en sorte de désenregistrer le bean.

Pour une utilisation en distribué, vous pourrez si besoin modifier certains ports utilisés par JBoss. Ces ports sont spécifiés dans la configuration du serveur par défaut située dans le répertoire `$JBOSS_HOME/server/default/conf`. Les fichiers concernés sont les fichiers `bindingservice.beans/META-INF/bindings-jboss-beans.xml` et `deploy/ejb3-connectors-jboss-beans.xml`.

## 2 Stateless Session Beans

Développez un EJB session sans état mettant en œuvre un convertisseur Euro vers Franc. Vous écrirez pour cela une interface *Convertisseur* offrant une méthode de conversion et une classe *ConvertisseurBean* implantant le code métier et accessible via l'interface distante *Convertisseur*.

Déployez ce bean dans la plate-forme JBoss. Notez le nom JNDI choisi pour lui par JBoss.

Écrivez et testez un client Java de votre bean. Ce client affichera le résultat de quelques conversions Euros vers Franc.

Vous pourrez utiliser le contenu suivant pour le fichier *jndi.properties* (avec l'URL du provider correspondant à un serveur JBoss local) :

```
java.naming.factory.initial=org.jboss.naming.NamingContextFactory
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
java.naming.provider.url=localhost
```

## 3 Stateful Session Beans

### 3.1 Gestion de stock

Développez un EJB session avec état mettant en œuvre une gestion de stock. Ce bean (*StockBean*) implante l'interface *Stock* offrant trois méthodes :

- `void ajout(int)` pour ajouter un certain nombre d'articles ;
- `void retrait(int)` pour retirer un certain nombre d'articles ;
- `int stock()` rendant le nombre d'articles courant dans le stock.

Le stock est un entier et sera initialement nul. On ne se préoccupera pas du fait que le stock puisse devenir négatif.

Écrivez et testez un client Java prenant en paramètre un entier *n* et exploitant *n* instances du bean *StockBean* (on pourra utiliser un tableau de références de beans). Pour chacune de ces instances il effectuera un appel à ajout, retrait et stock.

## 3.2 Intercepteurs

Les instances d'EJB avec état sont créées au sein d'une réserve d'instances (pool) de taille finie. Quand cette réserve est vide, il est nécessaire de passiver un bean pour pouvoir en créer un autre. Lorsque l'on invoque un bean, celui-ci est éventuellement activé s'il avait préalablement été passivé. Ce cycle de vie est géré par le conteneur (et donc par le serveur d'application JBoss) et est transparent pour le programmeur du bean.

On peut toutefois observer ce mécanisme en plaçant des intercepteurs sur les opérations relatives à la gestion du cycle de vie des EJB effectuée par JBoss.

Écrivez une classe *Intercepteur* où vous définirez des méthodes d'interception pour la création, la passivation et l'activation. Vous associerez ces intercepteurs à vos EJB de gestion stock en annotant la classe *StockBean* avec

```
@Interceptors(Intercepteur.class)
```

Ces intercepteurs se contenteront d'afficher un message, permettant ainsi de suivre le cylcle de vie des beans.

Testez votre client sur cette nouvelle version de l'EJB *StockBean*. Par défaut, vous devriez voir afficher le message signifiant la création des $n$ instances de beans, mais aucun message d'activation ou passivation. Pour forcer la passivation, il faut réduire la taille de la réserve d'instance pour qu'elle devienne plus petite que $n$. Ceci se fait en rajoutant, au niveau de la classe *StockBean* l'annotation suivante :

```
@CacheConfig(maxSize = m, idleTimeoutSeconds = t)
```

où $m$ est la taille de la réserve d'instances et $t$ le nombre de secondes d'inactivité après lequel le bean est passivé.