Atelier EJB N°1 : Stateless Session Bean

Environnement du travail

- Eclipse Helios SR2
  http://www.eclipse.org/downloads/packages/release/helios/sr2
  Package: Eclipse IDE for Java EE Developers
- jboss-5.0.1.GA (serveur d'application)
  http://sourceforge.net/projects/jboss/files/JBoss/JBoss-5.0.1.GA/
  Fichier : jboss-5.0.1.GA.zip

1. Création et déploiement d'un Stateless Session Bean

1.1 Création d'un EJB Project sous Eclipse

- Lancer Eclipse
- Créer un nouveau projet de type EJB
  - File > New > Others > EJB > EJB Project
  - Dans le champ « Project Name », spécifier le nom du projet : *TP1StatelessBean*
  - Dans le champ « Target Runtime », Spécifier Jboss V5.0 comme serveur d'application
    - Si Jboss V5.0 n'existe pas dans la liste déroulante, cliquer sur le bouton « New Runtime » pour définir un nouveau server runtime environment
  - Dans le champ « EJB Module Version », Choisir la version 3.0
  - Valider la création du projet en cliquant sur le bouton « Finish »

1.2 Création du Stateless Session Bean

- Créer un Stateless Session Bean:
  - TP1StatelessBean > Ejbmodule > New > Session Bean (EJB3.x)
  - Dans le champ « Java Package », saisir `ma.ensias.ejb3.tp1` comme nom du package
  - Dans le champ « Class Name », saisir `SearchBookFacade` comme nom de classe
  - Dans le champ « State Type », choisir le type Stateless
  - Cocher les deux cases « Remote » et « Local »

- Valider la création du Session Bean en sur le bouton « Finish »

Remarquer dans l'arborescence du projet que les fichiers suivants sont crées :

`SearchBookFacade.java`	La classe métier du statless session bean
`SearchBookFacadeRemote.java`	L'interface métier distante
`SearchBookFacadeLocal.java`	L'interface métier locale

1.3 Définition des interfaces métier

Dans les deux interfaces métier locale et distante, déclarer la méthode `bookSearch` qui permet d'afficher les livres relatifs à un mot clé.

```
List<String> bookSearch(String bookType);
```

1.4 Implémentation des méthodes métier

Dans la classe `BookSearchFacade`, implémenter la méthode `bookSearch` comme décrite ci-dessous :

```
public List<String> bookSearch(String bookType) {
	List<String> bookList = new ArrayList<String>();
	if (bookType.equals("java")) {
		bookList.add("Java for dummies");
		bookList.add("Beginnig Java 6");
	} else if (bookType.equals("C++")) {
		bookList.add("C++ for dummies");
	}
	return bookList;
}
```

1.5 Déploiement du Stateless Session Bean

- Définir JBoss comme serveur d'application du projet TP1StatelessBean :
  - TP1StatelessBean > Properties > Server
  - Sélectionner JBoss V5.0 at localhost
- Déployer l'EJB `BookSearchFacade` sur le serveur JBOSS :
  - TP1StatelessBean > Run AS > Run on Server
- Vérifier que l'EJB `BookSearchFacade` est publié sur le naming service de JBoss :
  - Dans un navigateur web, taper le lien : http://localhost:8080/jmx-console
  - JBoss > service=JNDIView > Bouton Invoke de l'operation "list"
  - Dans "Global JNDI Namespace", identifier les lignes suivantes:

```
+- SearchBookFacade (class: org.jnp.interfaces.NamingContext)
|   +- local (class: Proxy for: ma.ensias.ejb3.tp1.SearchBookFacadeLocal)
|   +- local-ma.ensias.ejb3.tp1.SearchBookFacadeLocal (class: Proxy for: ma.ensias.ejb3.tp1.SearchBookFacadeLocal)
|   +- remote-ma.ensias.ejb3.tp1.SearchBookFacadeRemote (class: Proxy for: ma.ensias.ejb3.tp1.SearchBookFacadeRemote)
```

```
|   +- remote (class: Proxy for: ma.ensias.ejb3.tp1.SearchBookFacadeRemote)
```

2. Développement de l'application cliente

2.1 Création d'un projet de type Application Client Project

- Sous Eclipse, créer un nouveau projet de type Application Client
  - File > New > Application Client Project
  - Dans le champ « Project Name », spécifier le nom du projet : *TP1StatelessBeanClient*
  - Dans le champ « Target Runtime », Spécifier Jboss V5.0 comme serveur d'application
  - Cliquer sur le bouton « Next », décocher la case « Create a default Main class »
  - Valider la création du projet en cliquant sur le bouton « Finish »
- Ajouter le projet *TP1StatelessBean* au build path du projet *TP1StatelessBeanClient*
  - *TP1StatelessBeanClient* > Properties > Java Build Path
  - Dans l'onglet « Projects », Ajouter le projet *TP1StatelessBean*

2.2 Définition de la classe SearchBookClient

- Créer une nouvelle classe sous le projet *TP1StatelessBeanClient*
  - *TP1StatelessBeanClient* > appClientModule > new > class
  - Dans le champ « Package », saisir `ma.ensias.ejb3.tp1.client` comme nom du package
  - Dans le champ « Name », saisir `SearchBookClient` comme nom de classe
  - Valider la création de la classe en cliquant sur le bouton « Finish »
  - Définir le constructeur et la méthode main de la classe `SearchBookClient`

```
public class SearchBookClient {

	public SearchBookClient() {
	}

	public static void main(String[] args) {
		SearchBookClient searchFacadeTest = new SearchBookClient();
		searchFacadeTest.doTest();
	}
}
```

2.3 Initialisation du contexte de nommage

- Dans la classe `SearchBookClient`, ajouter la méthode `getInitialContext` qui permet d'initialiser le contexte de nommage du serveur JBoss :

```
    InitialContext getInitialContext() throws javax.naming.NamingException {

                Properties p = new Properties();

                p.put(Context.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");
                p.put(Context.URL_PKG_PREFIXES,
"jboss.naming:org.jnp.interfaces");
                p.put(Context.PROVIDER_URL, "jnp://localhost:1099");

                return new InitialContext(p);

    }
```

2.4 Appel des méthodes métiers

- Dans la classe `SearchBookClient`, ajouter la méthode `doTest` qui permet :
  - D'initialiser le contexte de nommage en appelant la methode `getInitialContext`
  - Récupérer une instance de l'objet `SearchBookFacadeRemote` à partir du service de nommage
  - Appeler la méthode métier `bookSearch`
  - Afficher les resultats

```
    void doTest() {
          try {
                InitialContext ic = getInitialContext();
                System.out.println("SearchFacade Lookup");
                SearchBookFacadeRemote searchFacade = (SearchBookFacadeRemote)
ic.lookup(" ??? ");

                System.out.println("Searching books");
                // Dans ce code, le mot clé de recherche est donne en dur
                // Vous pouvez améliorer ce code en lisant ce mot clé
                // à partir de l'entrée standard
                List<String> bookList = searchFacade.bookSearch("java");

                System.out.println("Printing books list");
                for (String book : (List<String>) bookList) {
                      System.out.println(" -- " + book);
                }
          } catch (NamingException e) {
                e.printStackTrace();
          }
    }
```

2.5 Exécution de l'application cliente

- *TP1StatelessBeanClient* > SearchBookClient.java > Run As > Java Application

3. Callback Methods

Le conteneur d'EJB appelle ces méthodes à des stades appropriés du cycle de vie du bean.

Pour un Stateless session bean, il existe deux types de callback method :

PostConstruct	Toute methode qui a l'annotation `@PostConstruct`. Ce type de callback s'exécute après la création d'une instance d'un Stateless Session Bean
PreDestroy	Toute methode qui a l'annotation `@PreDestroy`. Ce type de callback s'exécute avant la supression d'une instance d'un Stateless Session Bean

- Dans le Session Bean `SearchBookFacade`, définir une méthode `searchBookByCountry` qui permet de lister les livres relatifs à un pays
- Définir une callback method de type PostConstruct pour remplir une HashMap `countryBookMap` avec des valeurs initiales
- Définir une callback method de type PreDestroy qui permet de vider `countryBookMap`

```
HashMap<String, String> countryBookMap = new HashMap<String, String>();
```

```
@PostConstruct
public void initializeCountryBookList() {
      countryBookMap.put("Australia", "Welcome to Australia");
      countryBookMap.put("Australia", "Australia History");
      countryBookMap.put("Morocco", "Welcome to Morocco");
      countryBookMap.put("Morocco", "Morocco History");
}
```

```
@PreDestroy
public void destroyBookList() {
      countryBookMap.clear();
}
```

4. Interceptors

La spécification EJB 3 fournit des annotations appelé intercepteurs, qui vous permettent de d'intercepter un appel de méthode métier.

Par exemple, les intercepteurs peuvent être utilisés pour :

- Effectuer des contrôles de sécurité supplémentaires avant l'exécution d'une méthode métier critique

- Effectuer des analyses de performance en calculant le temps d'exécution d'une méthode métier

- Ajouter une méthode `timerLog` pour calculer le temps d'exécution des méthodes métier du Session Bean `SearchBookFacade`

```
@AroundInvoke
public Object timerLog(InvocationContext ctx) throws Exception {
```

```
        String beanClassName = ctx.getClass().getName();
        String businessMethodName = ctx.getMethod().getName();
        String target = beanClassName + "." + businessMethodName;
        long startTime = System.currentTimeMillis();
        System.out.println("Invoking " + target);
        try {
              return ctx.proceed();
        } finally {
              System.out.println("Exiting " + target);
              long totalTime = System.currentTimeMillis() - startTime;
              System.out.println("Business method {" +
businessMethodName + "} in " + beanClassName + " takes " + totalTime +
"ms to execute");
        }
   }
```

5. Pour plus de pratique

- Faire appel à l'interface locale du Session Bean `SearchBookFacade` à partir d'une page jsp qui s'exécute dans la meme JVM du Session Bean
- Explorer l'utilisation des intercepteurs de type `@Interceptor` et `@Interceptors`