TP – Mise en œuvre d'un serveur d'applications (JBoss) : prise en main

JBoss est un serveur d'application Web conforme aux spécifications EJB. Il est Open Source, 100% Java et est distribué gratuitement (`http://www.jboss.org`). Pour son exécution, il est donc nécessaire d'avoir le JDK préalablement installé sur la machine.

L'objectif de ce TP est d'installer le serveur d'application, de l'exécuter, d'identifier ses fonctionnalités. La mise en œuvre d'une petite application illustrera l'utilisation du service de nommage.

1. Mise en place de l'environnement

JBoss est installé dans le répertoire /opt/jboss-3.2.6
(ce répertoire est uniquement le résultat du décompactage de l'archive chargée sur le site Web de JBoss).

- Examiner l'arborescence et identifier à quelles fonctionnalités correspondent les répertoires.
- Tester le lancement du serveur par :
  ```
  $ cd /opt/jboss-3.2.6/bin
  $ ./run.sh
  ```
- Que se passe-t-il ? Quelles sont les solutions possibles ? Pour un environnement multi-utilisateurs, quelle est la solution la plus adaptée ? Comment faut-il alors modifier l'environnement JBoss (options, variables, etc.)?
- Copier le fichier jboss-3.2.6-server-default.tar.gz présent dans le répertoire /home/Partage/enseignants/Info/TP-LeMouel/4TC-MID/ dans un répertoire de votre compte, décompresser le et exécuter le serveur :
  ```
  $ tar xvfz jboss-3.2.6-server-default.tar.gz
  $ cd jboss-3.2.6
  $ source env_jboss (examiner les variables modifiées)
  $ ./bin/run.sh
  ```
- Une fois lancé le serveur, vérifier son bon fonctionnement à partir d'un navigateur avec les URLs `http://localhost:8080` (console Tomcat) et `http://localhost:8080/web-console` (console JBoss), `http://localhost:8080/jmx-console` (console JMX). En identifier et détailler les différents éléments.

2. Exploitation du service de nommage JNDI

Un service de nommage permet d'associer un nom unique à un objet et facilite ainsi l'obtention de cet objet. JNDI est l'acronyme de « Java Naming and Directory Interface ». Cet API fournit une interface unique pour utiliser différents services de nommages ou d'annuaires: LDAP, DNS, RMI Registry, etc. JNDI est un service important de J2EE car il est utilisé par les différents composants de J2EE comme les EJBs, RMI, JDBC et JMS.

API JNDI

Les classes les plus utilisées sont :

`javax.naming.Context` : Une interface qui défini les méthodes pour chercher un objet java par son nom (`lookup`), lier un nom à un objet java (`bind`/`rebind`), et créer/détruire un sous-contexte (`createSubcontext` /`destroySubcontext`)

`javax.naming.InitialContext` : Toutes les opérations de nommage sont relatives à un contexte. La classe `InitialContext` concrétise l'interface `Context` et fournit le point de départ pour la résolution des noms.

`javax.naming.NamingException` (une classe d'exception de nommage) : Une classe d'`Exception` levée par les diverses opérations liées à JNDI.

Etat du service JNDI

- A partir de JMX, trouver le lien « `service=Naming` », cliquer dessus et trouver sur quel port le service est actuellement en train de s'exécuter.
- A partir de la console JMX, pour connaître les différents services actuellement enregistrés dans JBoss, trouver le lien « `service=JNDIView` », cliquer dessus, trouver l'opération « `list` » et invoquer cette méthode à l'aide d' « `invoke` ».

Initialisation de l'utilisation du service JNDI

JBoss fournit son propre service JNDI. Comme le service JNDI peut avoir plusieurs implantations selon le fournisseur de service, une application client qui utilise le service doit initialiser quelques variables d'environnement selon l'implantation du fournisseur.

- Ces propriétés peuvent être spécifiées dans un fichier indépendant `jndi.properties` :

```
java.naming.factory.initial=org.jnp.interfaces.NamingCont
extFactory
java.naming.provider.url=jnp://localhost:1099
java.naming.factory.url.pkgs=org.jboss.naming.client
```

- Ces propriétés peuvent également être spécifiées dans le code en modifiant directement les variables de contexte :

```
Properties env = System.getProperties();
env.put("java.naming.factory.initial",
     "org.jnp.interfaces.NamingContextFactory ");
env.put("java.naming.provider.url",
     "jnp://localhost:1099");
Context ctx = new InitialContext(env);
```

Le squelette des programmes clients ci-après se trouve dans l'archive /home/Partage/enseignants/Info/TP-LeMouel/4TC-MID/ MID_TD-4_JBoss_Install.tar.gz

Client test : interrogation du service JNDI

- Servez vous de la documentation des classes `Context`, `InitialContext` et `NamingException` pour compléter le code suivant qui permet d'accéder au service JNDI :

```
import &&&.. ;
import &&&.. ;

public class ClientJNDI {

     public static void main(String str[]){

          try{

             Context ctx=new &&&&&&..;
             Object obj=ctx.lookup( nom_JNDI_d_un_objet );

          } catch(&&&& ex){ ex.printStackTrace(); }
     }
}
```

Client test : enregistrement dans le service JNDI

- Ecrivez deux classes java s'échangeant des objets en passant par le service JNDI de JBoss :
  - La première classe doit utiliser la méthode `bind()` pour réaliser le "binding" d'un objet dans le service de nommage du serveur. Cet objet est une instance d'une classe `SharedData`, dans laquelle vous mettez ce que vous voulez.
  - La deuxième classe doit récupérer les entrées avec la fonction `lookup()` sur les objets déposés (sans l'utilisation des méthodes `list()` ou `listBindings()`).

Testez les deux manières d'entrer les variables de contexte !