Laboratoire du cours de Systèmes distribués

Année académique 2012 – 2013

3ème année Informatique et Systèmes – Informatique de Gestion

Mounawar Madani
François Caprasse
Cédric Thiernesse

Laboratoire 1 : Un premier EJB à la main…

Objectifs

- ✓ Découvrir Ant.
- ✓ Découvrir les EJB stateless et stateful.
- ✓ Comprendre et mettre en œuvre dans sa version basique le processus de packaging et de déploiement.
- ✓ Utiliser les outils javac, jar, … des plates-formes JDK et JEE

Partie 1 : EJB stateless

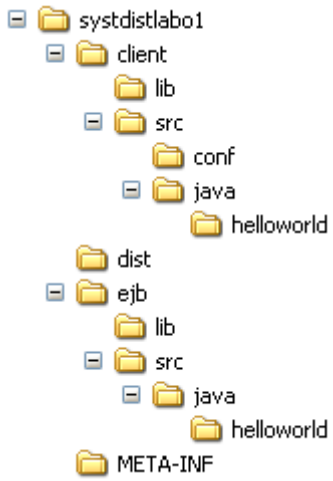
Etape 1 : Installation de l'IDE NetBeans et du serveur GlassFish

- ✓ Installer le dernière version de NetBeans et Glassfish (attention à bien conserver les paramètres user/pwd de GlassFish)
- ✓ Compléter les paramètres nécessaires au bon fonctionnement de Ant :

  http://forums.mediabox.fr/wiki/tutoriaux/java/ant_intro

  http://ant.apache.org/manual/install.html
- ✓ Démarrer et vérifier le bon fonctionnement du serveur GlassFish : http://docs.oracle.com/cd/E18930_01/html/821-2432/aboaa.html#scrolltoc

Etape 2 : Création de la structure du projet en terme de répertoires



Etape 3 : Création du module EJB

- ✓ Copier le fichier build.xml du module EJB dans le répertoire \systdistlabo1\ejb
- ✓ Editer et inspecter en détail le fichier build.xml qui permettra l'utilisation de Ant.
- ✓ Dans le répertoire \systdistlabo1\ejb\lib, placer le fichier javaee.jar ou javax.ejb.jar(un fichier contenant les .class dont nous avons besoin) (alternative afin de ne pas avoir de copie de ce fichier : modifier le classpath dans le fichier build.xml)
- ✓ Créer dans le répertoire \systdistlabo1\ejb\src\java\helloworld les fichiers :

HelloWorldBean.java

```
package helloworld;
import javax.ejb.Stateless;

@Stateless
public class HelloWorldBean implements HelloWorldRemote
{
        public String sayHello()
        {
                return  "hello";
        }
}
```

et

HelloWorldRemote.java

```
package helloworld;
import javax.ejb.Remote;

@Remote
public interface HelloWorldRemote
{
String sayHello();
}
```

- ✓ Compiler :
  - soit grâce à la commande : ant compile
  - soit en utilisant javac :

```
javac -sourcepath src  -d ./build/classes/ ./src/java/helloworld/*.java -classpath
./lib/javaee.jar
```

- ✓ Créer les jars de l'EJB grâce à la commande : ant jar

  Ceci a pour effet de créer 2 jars dans le dossier dist de l'EJB :

  - ejb-labo1.jar qui contient le module l'EJB.
  - lib\Interfaces-ejb-labo1.jar qui contient l'interface de l'EJB.

Etape 4 : Création du module Client

- ✓ Copier le fichier build.xml du client dans le répertoire \systdistlabo1\client
- ✓ Editer et inspecter en détail le fichier build.xml qui permettra l'utilisation de Ant.
- ✓ Dans le répertoire \systdistlabo1\client\lib, placer le fichier javaee.jar **et le fichier jar d'interface de l'EJB qui se trouve dans le répertoire /systdistlabo1/ejb/dist/lib**
- ✓ Créer dans le répertoire \systdistlabo1\client\src\java\helloworld le fichier :

Main.java

```
package helloworld;

import javax.ejb.EJB;

public class Main
{
   @EJB
   private static helloworld.HelloWorldRemote helloWorldRemote;

   public Main()
   {
   }

   public static void main(String[] args)
   {
         try
         {
                  System.out.println(helloWorldRemote.sayHello());
         }
         catch(Exception e)
         {
                  e.printStackTrace();
         }
   }
}
```

- ✓ Créer dans le répertoire \systdistlabo1\client\src\conf les fichiers :

application-client.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<application-client xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" metadata-complete="true"
version="5" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application-client_5.xsd">
<display-name>helloworld-app-client</display-name>
<ejb-ref>
<ejb-ref-name>helloworld.Main/HelloWorldBean</ejb-ref-name>
<ejb-ref-type>Session</ejb-ref-type>
<remote>helloworld.HelloWorldRemote</remote>
<ejb-link>ejb-labo1.jar#HelloWorldBean</ejb-link>
<injection-target>
<injection-target-class>helloworld.Main</injection-target-class>
<injection-target-name>helloWorldRemote</injection-target-name>
</injection-target>
</ejb-ref>
</application-client>
```

et

MANIFEST.MF

```
Main-Class: helloworld.Main
```

- ✓ Compiler grâce à la commande : ant compile
- ✓ Création du jar grâce à la commande : ant jar

Etape 5 : Création et déploiement de l'Enterprise Application (EAR)

- ✓ Créer dans le répertoire \systdistlabo1\META-INF les fichiers :

application.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<application version="5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application_5.xsd">
  <display-name>applicationEnterpriset</display-name>
  <module>
    <java>labo1-app-client.jar</java>
  </module>
  <module>
    <ejb>ejb-labo1.jar</ejb>
```

```
  </module>
</application>
```

et

sun-application.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-application PUBLIC "-//Sun Microsystems, Inc.//DTD Application
Server 9.0 Java EE Application 5.0//EN"
"http://www.sun.com/software/appserver/dtds/sun-application_5_0-0.dtd">
<sun-application>
</sun-application>
```

- ✓ Créer le fichier EAR grâce à la commande :

  systdistlabo1>jar cvfM ./dist/applic.ear ./META-INF/* -C ./client/dist . -C ./ejb/dist .

- ✓ Créer dans le répertoire \systdistlabo1 le fichier :

pass.properties

```
#
#Tue May 06 12:48:49 CEST 2008
AS_ADMIN_PASSWORD=adminpwd
```

- ✓ Déployer l'Enterprise Application :

  systdistlabo1/dist>asadmin deploy --user admin --host nommachine --port 4848 --passwordfile ../pass.properties applic.ear

- ✓ Vérifier le bon déploiement via la console d'administration de GlassFish : http://localhost:4848/

Etape 6 : Récupération et exécution du client en local

- ✓ Dans un répertoire testclientlocal, récupérer les fichiers « client » de l'Enterprise Application :

  testclientlocal >asadmin get-client-stubs --user admin --appname NomEnterpriseApplication .\

- ✓ Lancement de l'application cliente : il suffit maintenant de lancer l'applicationen tapant (régler les chemins si nécessaire):

  appclient -client XXXXXXClient.jar

Etape 7 : Exécution du client sur une machine quelconque

Sur le serveur :

- ✓ Dans un répertoire testclientdistant\MonApplication, récupérer les fichiers clients de l'Enterprise Application :

  Testclientdistant\MonApplication>asadmin get-client-stubs --user admin --appname NomEnterpriseApplication .

  On obtient ainsi un fichier nommé applicClient.jar et un dossier applicClient.

- ✓ Obtenir l'environnement qui sera indispensable à votre client pour s'exécuter :

  testclientdistant >package-appclient

  On obtient alors l'environnement pour une machine cliente : appclient.jar qui se trouve par défaut dans le répertoire d'installation de Glassfish suivit de /lib/appclient/appclient.jar

- ✓ Décompresser appclient.jar dans le dossier testclientdistant\ContainerClient

- ✓ Modifier le fichier sun-acc.xml dans testclientdist\ ContainerClient \glassfish\domains\domain1\config :

balise <target-server name="XXX.XXX.XXX.XXX" address="XXX.XXX.XXX.XXX" port ="3700"/>

Remplacer simplement XXX.XXX.XXX.XXX par l'adresse ip de la machine sur laquelle le serveur d'application JEE tourne avec votre « Enterprise application ».

Sur le client distant (une autre machine que le serveur) :

- ✓ Copier l'ensemble du dossier testclientdistant sur la machine cible.
- ✓ Lancement de l'application cliente : il suffit maintenant de lancer l'application comme précédemment en utilisant l'environnement que nous venons de copier :

testclientdist\ContainerClient\glassfish\bin > appclient –client ..\..\..\Monapplication\XXXXClient.jar

Partie 2 : EJB stateful

Maintenant que vous avez créé un premier EJB, que vous êtes familiarisés avec l'utilisation de Ant, avec le processus de déploiement, … vous allez réitérer l'ensemble des étapes de la première partie pour créer un EJB stateful.

Il s'agit de créer un nouvel EJB qui pourra être

Cet EJB (EJBCalcul) comportera une variable membre de type vecteur destinée à recevoir un ensemble de nombres.

Il comportera également 5 méthodes :

- ✓ addNombre qui permettra d'ajouter un nombre au vecteur
- ✓ getSomme qui renverra la somme des nombres compris dans le vecteur
- ✓ getMoyenne qui renverra la moyenne des nombres compris dans le vecteur
- ✓ getMinimun qui renverra le plus petit nombre compris dans le vecteur
- ✓ getMaximum qui renverra le plus grand nombre compris dans le vecteur

Il est évident qu'il faudra également écrire un client permettant de tester notre EJBCalcul. Il faudra pouvoir exécuter **deux clients simultanément** et pouvoir **saisir les nombres** au clavier de manière à analyser le comportement de votre EJB.

Une fois cet EJB créé, il faudra **se poser une série de questions** :

- ✓ Que se passe-t-il si plusieurs clients accèdent simultanément à notre EJB ?
- ✓ Que se passerait-il si celui-ci avait été déclaré stateless et non stateful ?

Vous êtes donc invités à tester ces différents cas de manière à répondre à ces questions !

Remarques générales:

- ✓ N'oubliez pas de démarrer le serveur GlassFish avant d'exécuter le client.
- ✓ Attention à configurer correctement votre firewall pour les accès à distance.