Université de Montpellier
Faculté des Sciences
Département Informatique

Filière : Master Informatique **Intervenant :** Chouki TIBERMACINE

TP EJB 1

Objectifs du TD

- Développer un composant EJB session sous l'environnement Netbeans
- Développer un composant Web et le connecter au composant EJB
- Déployer les composants dans le serveur d'application GlassFish d'Oracle
- Écrire une application cliente du composant EJB et l'exécuter en dehors du serveur

Exercice 1.

- Écrire un composant EJB session sans état qui permet de faire la conversion d'un montant en euros dans d'autres monnaies (Dollars américains/canadiens, Yen, ...),
- Écrire une page HTML qui contient un formulaire de saisie du montant et du choix de la monnaie cible
- Écrire une page JSP qui récupère les informations saisies dans le formulaire HTML et qui utilise le bean session pour effectuer la conversion de monnaie selon les taux de change en cours disponibles dans cet URL : http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml
- Déployer toute l'application dans le serveur d'application libre GlassFish d'Oracle
- Tester l'application sur votre navigateur Web.

Installations nécessaires.

Les deux outils suivants doivent être installés sur votre machine pour réaliser le TP :

- le JDK : http://www.oracle.com/technetwork/java/javase/downloads/index.html
- l'environnement Netbeans : http://www.netbeans.org/downloads/ (pack Java EE)

Il faudra s'assurer que le serveur d'application GlassFish est bien fourni avec Netbeans.

Noter que la version de Netbeans installée n'inclut pas par défaut le serveur d'application GlassFish.

- Télécharger le serveur GlassFish à partir du site suivant et l'installer dans votre répertoire personnel (~/ sous Linux). https://glassfish.java.net/download.html (télécharger la plate-forme complète)
- Lors de l'installation ajuster éventuellement le port HTTP si l'assistant de configuration vous indique que le port est déjà utilisé par ailleurs (mettre par exemple 8081 à la place de 8080)
- Laisser les autres paramètres par défaut
- Indiquer le chemin vers le SDK en cliquant sur le bouton « ... » (/usr/local/jdk/, par exemple)

Une fois l'environnement installé, voici la procédure à suivre pour réaliser le TP.

- Lancer Netbeans (commande netbeans& sous Linux).

- Créer un nouveau projet Java EE :
  - cliquer sur File -> New Project
  - dans la rubrique Categories, choisir Java EE, puis dans la rubrique Projects, choisir Enterprise Application. Cliquer sur Next
  - donner un nom à votre projet (Converter), choisir un répertoire pour votre projet (là où Netbeans va placer vos sources et les fichiers compilés)
  - choisir le serveur d'application où vont être déployés vos composants : lors du premier démarrage, ajouter un serveur d'application
    - cliquer sur Add...
    - choisir GlassFish Server, puis cliquer sur Next
    - choisir le chemin d'installation du serveur d'application (cliquer sur Browse, puis aller vers le répertoire où vous avez installé GlassFish : sous répertoire de votre répertoire personnel : ~/)
    - laisser les paramètres par défaut, puis cliquer sur Finish.
  - cliquer sur Finish
  - Netbeans va créer deux composants vides (un composant EJB Converter-ejb et un composant Web Converter-war). Chacun des deux composants est représenté par un répertoire qui contient plusieurs sous-répertoires contenant les sources et les fichiers de configuration. NetBeans crée également un autre répertoire pour l'application principale (Converter) avec plusieurs sous-répertoires de configuration

- Implémenter le bean session :
  - cliquer avec le bouton droit de la souris sur l'icône du composant EJB dans le menu à gauche (Converter-ejb)
  - choisir New -> Session Bean...
  - donner un nom au bean session (ConverterBean)
  - donner un nom de package Java dans lequel la classe créée sera placée (converter, par exemple)
  - s'assurer que le type de bean session est bien Stateless
  - Netbeans va créer une classe Java (ConverterBean)
  - créer une interface Java dans le même package et y ajouter la déclaration de la méthode suivante :

```
public double euroToOtherCurrency(double amount, String currencyCode);
```

- Éditer la classe du bean :
  - ajouter l'implémentation de l'interface :

```
public class ConverterBean implements Converter {...}
```

  - ajouter la méthode de l'interface puis écrire son implémentation
  - dans cette implémentation, vous pouvez utiliser l'URL suivant (qui a été donné dans la page précédente) pour rechercher le taux de change courant : http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml
  - pour analyser ce document XML, utiliser le parser jDOM 2 que vous devez télécharger depuis le site suivant (voir l'extrait de code donné plus loin pour commencer à utiliser jDOM) : http://www.jdom.org/downloads/

- décompresser l'archive ZIP téléchargée dans le répertoire de votre choix et ajouter dans le CLASSPATH du composant EJB le JAR extrait du ZIP (jdom-xxx.jar) :
  cliquer sur le bouton droit de la souris sur le composant EJB, choisir Properties, puis Libraries. Ensuite, cliquer sur « Add JAR/Folder » et indiquer le chemin jusqu'à « jdom-2.0.5.jar »
- Voici un extrait de code Java pour commencer (ajouter les import et les try-catch) :

```
SAXBuilder sxb = new SAXBuilder();
URL url = new URL(
   "http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml");
Document document = sxb.build(url);
Element racine = document.getRootElement();
Namespace ns = Namespace.getNamespace(
       "http://www.ecb.int/vocabulary/2002-08-01/eurofxref");
Element elem = racine.getChild("Cube", ns);
```

  Noter ici l'utilisation d'un espace de nom (ns) pour la navigation dans les éléments du document XML. Noter également la présence de plusieurs niveaux d'éléments « Cube ». Compléter le code ci-dessus pour naviguer jusqu'aux monnaies et leurs taux de change
- la documentation complète de l'API jDOM 2 est disponible ici :
  http://www.jdom.org/docs/apidocs/

- Implémenter le composant Web :
  - cliquer avec le bouton droit de la souris sur le composant Web (Converter-war)
  - choisir New -> HTML...
  - donner un nom à votre page Web (index), puis cliquer sur Finish
  - un fichier vide index.html est généré et pré-rempli de code HTML de base
  - éditer cette page Web en créant un formulaire où l'on peut saisir un montant à convertir et une monnaie cible
  - on demande l'exécution d'un script JSP (action="convert.jsp") dans le formulaire, lorsque ce dernier est soumis au serveur
  - cliquer encore une fois avec le bouton droit de la souris sur l'icône du composant Web (Converter-war) et choisir New -> JSP...
  - donner un nom à votre page JSP (convert)
  - s'assurer que c'est 'JSP File' qui est sélectionné, puis cliquer sur Finish (un fichier convert.jsp est alors généré)
  - éditer cette page JSP qui permet de récupérer les données saisies par l'utilisateur et qui fait appel au composant EJB pour convertir la montant saisi :
    A l'intérieur de l'élément <body> de la page JSP ajouter ce qui suit :

```
<jsp:useBean class="convert.ConverterBean" id="beanConv"/>
```

    Ceci indique au serveur qu'il faut faire une injection de dépendance qui permet de connecter le composant EJB au composant Web, lors du déploiement (nous allons désormais pouvoir référencer le composant EJB ayant une classe convert.ConverterBean, sous le nom beanConv)

```
<%@page import="java.util.*" %>
```

    Ceci sert à importer toutes les classes du package java.util
    Ajouter les imports nécessaires ...
    <% Tout ce qui est placé ici c'est du code JAVA classique (jusqu'à "%>")

```
double amount =
     Double.parseDouble(request.getParameter("amount")));
```

    Cette instruction permet de récupérer l'élément du formulaire dont le nom est `amount` (équivalente à `$_POST['amount']` ou `$_GET['amount']` de PHP)

Ajouter le code nécessaire pour récupérer le code de la monnaie cible
`amount = beanConv.euroToOtherCurrency(amount,currency);`
Ceci permet d'appeler la méthode `euroToOtherCurrency` du bean
`out.println("<h4>Le montant converti est : </h4>"+amount);`
Ceci permet de générer le code HTML placé en paramètre
(équivalent à `echo '<h4>...';` de PHP) ...

```
%>
```

- Votre application est prête à être déployée et exécutée

- Déployer et exécuter votre application : cliquer avec le bouton droit sur l'icône Converter, puis Run

- Pour dé-déployer votre application :
  - choisir l'onglet Services dans la partie supérieure gauche de Netbeans
  - aller dans : Servers -> GlassFish Server ... -> Applications
  - cliquer avec le bouton droit sur l'icône de votre application, puis choisir Undeploy

- Pour effacer les messages affichés dans les consoles de sortie de Netbeans (celle du serveur d'application ou celle du déploiement de votre application) :
  - cliquer au centre de la console avec le bouton droit de la souris, puis choisir Clear

- A la fin de la séance de TP, ne pas oublier de :
  - dé-déployer vos applications
  - arrêter le serveur d'application :
    - choisir l'onglet Services dans la partie supérieure gauche de Netbeans
    - cliquer sur Servers
    - cliquer avec le bouton droit sur GlassFish Server ..., puis choisir Stop
  - fermer Netbeans

Exercice 2.

- A la place du composant Web de l'application (considéré comme le client du composant EJB), nous allons écrire une application cliente Java indépendante (classe avec une méthode main qui va utiliser le composant EJB et qui va être déployée dans une JVM indépendante, pas celle utilisée par le serveur d'application) :
  - Éditer la classe du bean et ajouter l'annotation suivante avant l'entête de la classe :
    `@Remote(Converter.class)`
    Ceci indique au serveur d'application, lors du déploiement, que le bean session a une interface Remote qui s'appelle Converter, et peut donc être utilisé par des clients distants. S'assurer que dans la classe du bean, il y a : `implements Converter` (changer le nom de l'interface, le cas échéant). Ne pas implémenter plusieurs interfaces dans la classe du bean.
  - Déployer l'application Java EE dans le serveur d'application GlassFish
  - Nous allons désormais travailler en dehors de Netbeans (ne pas le fermer). Nous allons utiliser un Terminal et un éditeur de texte (Emacs, par exemple)
  - Copier l'interface du bean (Converter.class) dans un répertoire de votre choix (nommé : client/). Par contre, veiller à mettre ce fichier dans la même hiérarchie de répertoires que celle correspondant aux packages de l'interface, et ses éventuelles sous-packages (client/converter/Converter.class par exemple, si l'interface est déclaré dans un package qui s'appelle converter)

- Écrire une classe Java (Client.java) avec une méthode main dans le même répertoire (client/)
- Cette méthode doit effectuer un lookup du bean en utilisant les instructions suivantes :

```
Converter converter = (Converter) InitialContext.doLookup(
        "java:global/Converter/Converter-ejb/ConverterBean");
```

  La chaîne de caractères passée en argument à la méthode lookup ci-dessus est le nom JNDI donné automatiquement par le serveur d'application au bean lorsqu'il a été déployé. Ce nom a été utilisé lors du déploiement pour enregistrer le bean auprès du serveur de noms JNDI.
  (Ne pas oublier d'ajouter les imports nécessaires : javax.naming.InitialContext, ...)
- Pour que le bean soit retrouvé en dehors du serveur d'application, il faudra éditer quelques propriétés de configuration : créer un fichier texte, le nommer jndi.properties et le placer dans le même répertoire que l'emplacement à partir duquel vous allez lancer l'application Java (là où vous allez écrire la commande `java` : dans le répertoire client/ par exemple)
- Mettre dans ce fichier les 3 lignes suivantes

```
java.naming.factory.initial=com.sun.enterprise.naming.SerialInitContextFactory
org.omg.CORBA.ORBInitialHost=...
org.omg.CORBA.ORBInitialPort=3700
```

  Ne pas oublier de remplacer « ... » par localhost. Dans l'exercice suivant, on va y mettre l'adresse IP de la machine sur laquelle se trouve le serveur d'application dans lequel le composant EJB est déployé
- Ajouter à la méthode main le code qu'il faut pour demander à l'utilisateur un montant à convertir et le code de la monnaie cible (utiliser simplement la classe Scanner), puis afficher le résultat retourné par la méthode du bean
- Compiler la classe, puis l'exécuter :

```
export CLASSPATH=.:jndi.properties:~/glassfish4/glassfish/lib/appserv-rt.jar
java Client
```

Le jar ajouté dans le CLASSPATH ci-dessus comporte les classes nécessaires que la JVM instancie pour retrouver le serveur de nom (et son contexte initial indispensable au lookup du bean).

Exercice 3.

Les machines des salles de TP sont sur le même réseau. On va déployer le composant EJB sur une machine et lancer l'exécution de l'application Java cliente de cet EJB sur une autre machine.

- Demander à un voisin l'adresse IP de sa machine. Vous pouvez aussi ouvrir une session sur une machine voisine
- Éditer le fichier jndi.properties pour y mettre l'adresse IP à la place de localhost
- S'assurer que le nom JNDI du composant EJB recherché (passé en argument à la méthode lookup dans l'application client Java) correspond bien à celui qui a été donné par le serveur d'application là où le composant EJB a été déployé
- Si vous n'avez rien modifié dans le code Java de l'application cliente, pas besoin de re-compiler (seul le fichier jndi.properties, lu par la JVM à l'exécution, aura été modifié)
- Lancer l'exécution de votre programme comme précédemment (en supposant que le CLASSPATH comporte les mêmes informations que dans l'exercice précédent) :

```
java Client
```

C. TIBERMACINE