**Université de Montpellier**
**Faculté des Sciences**
**Département Informatique**

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

# TP EJB 3

## Objectifs du TD

- Programmer un client de service Web
- Utiliser des services Web SOAP et des services Web REST
- Déployer une application avec un client de services Web dans un serveur d'application
- Déployer des EJB comme services Web et les tester

### Exercice 1.

- L'objectif de cet exercice est de modifier l'application écrite dans les TP EJB précédents pour que la conversion soit effectuée non pas en analysant des fichiers XML accessibles en ligne, mais en invoquant les opérations d'un service Web

- Nous allons utiliser un service Web REST, qui s'apprête mieux à notre exercice et à ce type de requêtes assez simples (ne nécessitant pas une sauvegarde d'état entre les invocations ou un contexte transactionnel). Par ailleurs, les services REST n'utilisent généralement pas des enveloppes SOAP pour créer les paramètres et les valeurs de retour des opérations fournies par les services. Ils s'appuient dans ce cas uniquement sur les requêtes et réponses HTTP (correspondant aux méthodes HTTP Get, Post, ...). Ceci rend les échanges entre le client et le service Web plus légers et donc plus efficaces (surtout pour la conversion du montant dans toutes les monnaies disponibles, dans notre exercice).

- Nous allons utiliser le service Web REST de conversion de monnaie documenté et fourni par le site suivant : http://currencies.apps.grandtrunk.net/

- Créer une nouvelle application d'entreprise (JEE) et copier dedans les fichiers du TP 1

- Modifier le bean session pour obtenir les codes de monnaies et le taux de change :
  - Pour obtenir tous les codes de monnaies, il faudra simplement utiliser l'URL suivant http://currencies.apps.grandtrunk.net/currencies de la façon suivante :
    ```
    URL url =
       new URL("http://currencies.apps.grandtrunk.net/currencies");
    BufferedReader in =
       new BufferedReader(new InputStreamReader(url.openStream()));
    ```
  - Il faudra ensuite parcourir le message (flux, ici) texte retourné par le service en utilisant la variable `in` :
    ```
    String inputLine;
    while ((inputLine = in.readLine()) != null) {
       // Dans inputLine, nous avons le code d'une monnaie
       ...
    }
    in.close();
    ```

- Pour obtenir le taux de change de l'Euro vers une monnaie cible, nous allons utiliser l'URL suivant
  `http://currencies.apps.grandtrunk.net/getlatest/<fromcode>/<tocode>`
  de la façon suivante :
  ```
  url =
  new URL("http://currencies.apps.grandtrunk.net/getlatest/"
          +"EUR"+"/"+inputLine);
  BufferedReader in2 =
     new BufferedReader(new InputStreamReader(url.openStream()));
  ```

- Récupérer le taux de change dans : `in2.readLine()`
- Faire attention ici aux monnaies qui sont parfois disponibles dans la liste des monnaies (1er URL), mais dont les taux de change ne sont pas disponibles (2nd URL). Cela retourne une erreur HTTP 400 lors de l'appel à `openStream()` sur l'objet `url`, et ça provoque une exception de type `IOException`. Ajouter ici un `try-catch` pour capturer cette exception et journaliser (loguer) ce code de monnaie. La conversion dans les autres monnaies doit se poursuivre après

- Déployer l'application et la tester sur votre navigateur

**Exercice 2.**

- Copier l'application (ses 3 projets) comme précédemment
- Publier le bean session comme service Web SOAP :
  - Tout d'abord, rendre le bean session Stateless et remettre les attributs de la classe comme variables locales dans les deux méthodes de conversion
  - Annoter la classe du bean par `@WebService` (une annotation se trouvant dans `javax.jws`)
  - Annoter les méthodes du bean par `@WebMethod` (`javax.jws.WebMethod`)
  - Ne pas utiliser des interfaces Java comme types des paramètres ou comme types de retour des méthodes publiées comme opérations du service Web. Dans le code que j'ai donné dans le TP précédent, j'ai indiqué comme type de retour pour l'une des méthodes : Map<Monnaie,Double>. Remplacer Map par HashMap ou tout autre type concret (classe) compatible avec Map et ayant un constructeur sans paramètres, qui sera invoqué par le serveur d'application
- Il existe une autre méthode (plus automatisée) pour publier le bean session comme service Web (si vous avez fait l'étape d'avant, passez alors cette étape) :
  - Cliquer avec le bouton droit de la souris sur le projet du composant EJB et choisir New > Web Service...
  - Donner un nom à votre service : ConverterWS
  - Choisir « Create Web Service from Existing Session Bean », puis cliquer sur « Browse »
  - Choisir le bean session se trouvant dans le même projet (ConverterBean)
  - Lui donner un nom de package : le même que celui du bean session (convert)
  - Cliquer sur Finish
  - NetBeans va créer une classe ayant le nom du service et comportant des méthodes correspondant aux méthodes du bean. Il les a décoré avec les annotations nécessaires (`@WebService`, `@WebMethod`, ...) et y a inclut les délégations qu'il faut
- Déployer le service Web dans GlassFish :
  - Cliquer avec le bouton droit de la souris sur le projet du composant EJB de

l'application (menu à gauche de NetBeans, qui affiche les projets ouverts) et choisir Deploy

- Tester l'exécution du service Web depuis votre machine :
  - Ouvrir sur NetBeans le répertoire « Web Services » du projet du composant EJB. Vous aller apercevoir le service Web (ConverterBean)
  - Cliquer avec le bouton droit sur l'icône du service et choisir « Test Web Service »
  - Une fenêtre du navigateur va s'ouvrir affichant les opérations du service et des champs pour éditer les arguments nécessaires à l'invocation de ces opérations.
  - Remplir les champs de l'une des opérations et cliquer sur le bouton à côté, pour invoquer l'opération
  - Visualiser le document WSDL qui a été généré pour votre service. Le lien est fourni sur la page Web de test du service

Chouki Tibermacine