Marie-Pierre Gervais
EC FSR

Entity
TP conversionTaux

L'application EJB `conversionTaux` a comme fonctionnalité première la conversion monétaire basée sur un taux de conversion stocké en base.

Inspiré de nombreux exemples d'applications Web, le service est disponible en une version publique, et des services privilégiés sont offerts à des abonnés, donc des utilisateurs qui se sont créés un compte.

Pour mettre en œuvre progressivement les différents services, l'application est à développer en plusieurs itérations

À chaque itération :

1. Une version de l'application doit être produite. À vous de gérer le versioning, ce qui importe, c'est qu'à toute itération n > 1, l'itération n-1 soit opérationnelle, i.e., son exécution peut être lancée à demande et elle fonctionne.
2. À vous de déterminer l'architecture du code serveur EJB, i.e., les intefaces/beans/entités nécessaires, la nature des beans, etc. Respectez les signatures et les codes retour fournis dans l'énoncé SVP.
3. En suivant les spécifications de l'itération, à vous d'écrire le code client qui teste correctement (de façon exhaustive) que votre code serveur est bon : opérationnel et couverture des spécifications.
4. En prenant l'outil de votre choix selon le SGBD que vous utilisez, n'oubliez pas de vérifier en BD les ajouts/suppressions/modifications de rangée(s)
5. Respecter les noms fournis dans l'énoncé et d'une façon générale, les conventions de nommage fournies dans le cours.
6. La description de l'itération est le minimum de ce qui doit être implanté et opérationnel. Pour vous entraîner, des suggestions complémentaires sont fournies.

Tables des codes retour des méthodes

Les méthodes renvoient toutes une String dont la valeur est une constante.

Les constantes sont définies dans l'interface (pur java) `ConversionTauxCste.java` qui est dans le package `helper` et l'interface des services. Cette interface `ConversionTauxCste` est implantée par le bean.

D'autres constantes pourront être nécessaires au fur et à mesure des itérations et peuvent déjà figurer dans le fichier fourni.

Itération 1

Dupliquer l'exemple `conversionTaux` utilisé pour le test du branchement deGlassfish et d'Oracle.
Dans la duplication, nouvel exercice (itération1) :

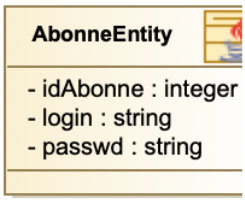
L'application EJB `conversionTaux` doit désormais permettre de s'abonner, c-à-d, de créer un compte. Ainsi le code serveur EJB offre un nouveau service (en plus du service `convertir`), qui est le service d'abonnement d'un utilisateur (on verra ultérieurement quel est son usage).

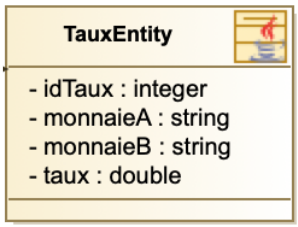
```
String creerCompte(String login, String passwd) ;
```

Pour exécuter ce service `creerCompte`, le bean :

- Utilise une entité nommée `AbonneEntity` (cf. le diagramme UML). Elle correspond à une table `Abonne` en BD.
- Crée une nouvelle instance de `AbonneEntity` en renseignant ses attributs login et passwd avec les valeurs fournies en paramètre. Utiliser l'annotation `@GeneratedValue` pour générer automatiquement la valeur de l'identifiant.
- Gère les cas d'erreur (demande de création de compte avec un login déjà existant

La modélisation UML des données persistantes est alors la suivante :





Le code client teste le code serveur EJB en :

1) Utilisant le service `convertir` en fournissant en paramètres un montant à convertir, la monnaieA (de ce montant) et la monnaieB. Le client affiche la String qu'il reçoit du serveur.
2) Utilisant du service `creerCompte` en fournissant en paramètre un login et un passwd. Le client affiche la String qu'il reçoit du serveur.

N'hésitez pas à ajouter des nouvelles constantes dans le fichier utilitaire `ConversionTauxCste` pour avoir des messages retour (du serveur vers le client) « propres », c-à-d bien explicites (cf ce qui a été fait dans l'exemple test).