TP 2 : les EJB

Architecture JEE

Master 2 Génie Logiciel

Préparation de l'environnement de développement

- JBoss 7.1.1 full profile :
  http://www.jboss.org/jbossas/downloads/

- Eclipse for EE Developers :
  http://www.eclipse.org/downloads/

- Plugin Jboss Tools -> Depuis Eclipse MarketPlace.

Préférences Eclipse

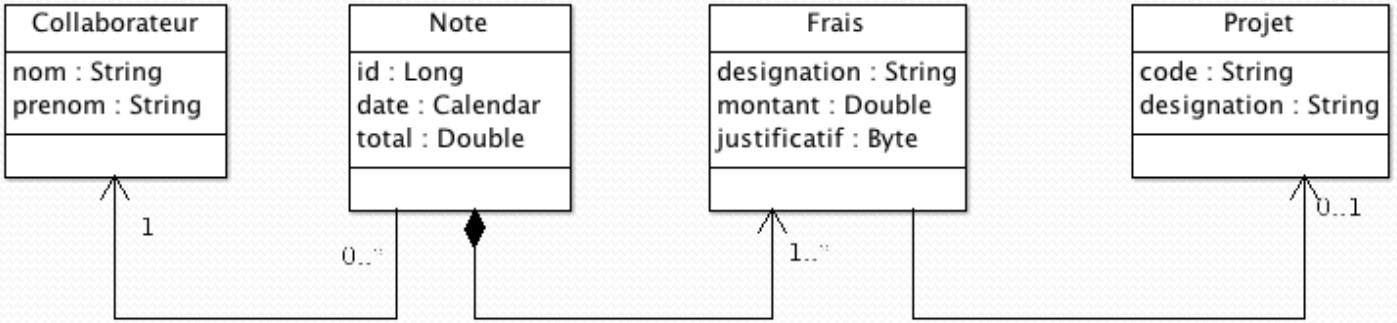
- Choisir un emplacement pour son workspace

- Préférences Eclipse :
  - ✓ Encodage du workspace : UTF-8,
  - ✓ Server Runtime : Jboss 7.1.1,
  - ✓ Instancier un serveur en pointant le fichier de configuration du déploiement (copie de standalone.xml renommée « master-info.xml »

Etude de cas

- Nous allons développer une application permettant à des collaborateurs au sein d'une entreprise de saisir leurs notes de frais.

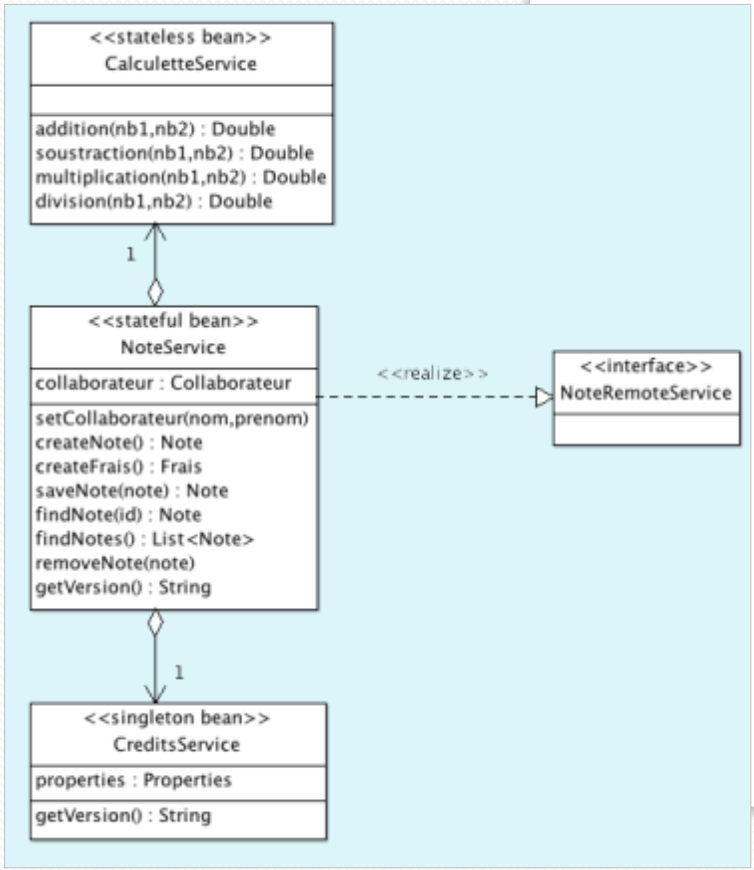
Diagramme des classes « entités »



Spécifications de la couche métier

Nous souhaitons implanter la couche métier sous la forme de 3 services :

- ✓ Une calculatrice permettant de réaliser les 4 types d'opération habituels,
- ✓ Un service dont l'unique but est de nous fournir la version de l'application,
- ✓ Un service nous permettant de gérer des notes de frais : création, chargement, recherche, sauvegarde, suppression.



Couche métier

- NoteService sera un EJB Façade
- Etant testé via un client distant : nécessité de lui associer une interface distante.
- Gestion des entités : pas de persistance pour le moment : stockage des entités créées en mémoire (Map d'entités déclarée sous forme de variable initialisée à l'instanciation du premier EJB)
- Le total se calculera à la sauvegarde, au niveau de NoteService par invocation de CalculetteService.

Remplacer par createNote (note)

Le programme client

Développement d'un programme main exécuté sous Java SE qui exécute successivement les actions suivantes :

- « authentification » du collaborateur,
- Affichage de la version de l'application,
- Création d'une note,
- Sauvegarde de la note,
- Chargement de la note sauvegardée et affichage du total,
- Recherche des notes du collaborateur,
- Suppression de la note,
- Recherche des notes restantes du collaborateur.

Configuration du programme client

Nécessité de développer un ServiceLocator capable de faire des appels distants à l'application « côté serveur ».
Pré-requis :

- ✓ Fichier de propriétés jboss-ejb-client.properties,
- ✓ Fichier de propriétés jndi.properties,
- ✓ Librairie jboss-client.jar,
- ✓ Entités sérialisables.

Les fichiers de configuration des appels distants

Jboss-ejb-client.properties

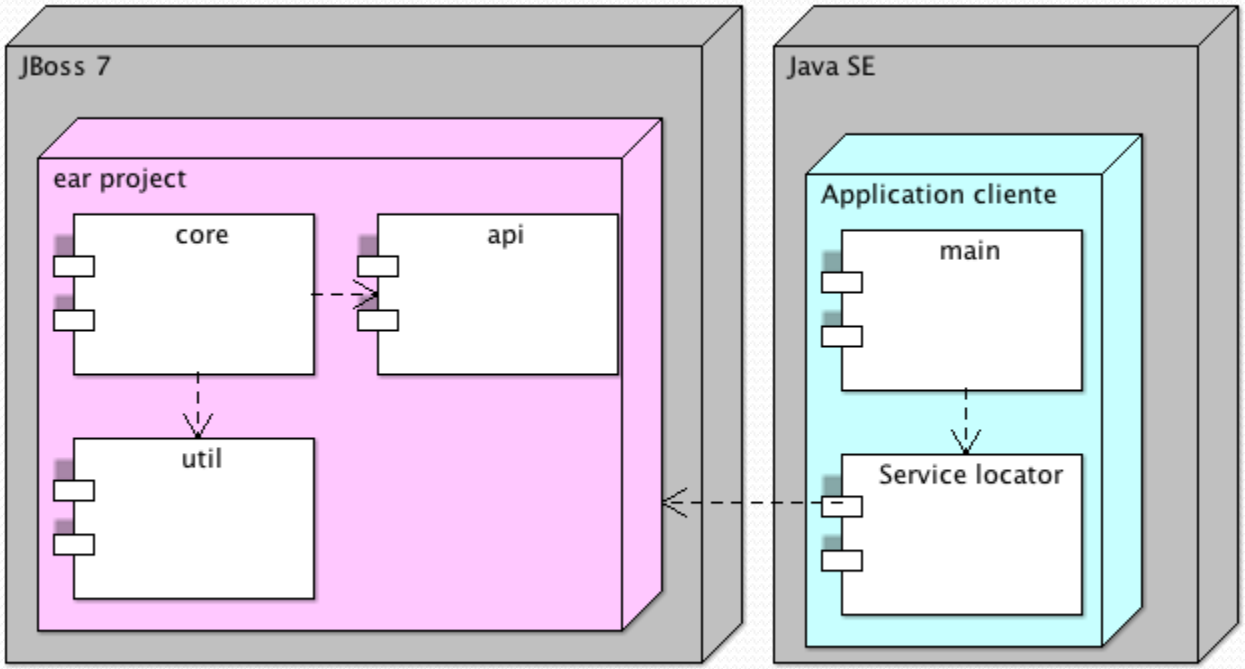
```
endpoint.name=client-endpoint
remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false
remote.connections=default
remote.connection.default.host=localhost
remote.connection.default.port=4447
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOANONYMOUS=false
remote.connection.default.username=
remote.connection.default.password=
```

Jndi.properties

```
java.naming.factory.url.pkgs=org.jboss.ejb.client.naming
```

Doivent être accessibles via le classpath de l'application (les placer dans le répertoire src)

Diagramme de déploiement



Méthodes de rappel

Définir une méthode de rappel à l'instanciation pour :

- ✓ CreditService : récupération des propriétés credits.properties,
- ✓ NoteService : initialisation de la map de notes en mémoire.

Intercepteur autour des appels

Pour pouvoir étudier le comportement de l'application lors d'appels concurrents, définir dans NoteService un intercepteur autour des appels réalisant les actions suivantes :

- En amont de l'appel : log dans la console du début de l'appel puis temporisation de 5 secondes (Thread.sleep(5000)),
- En aval de l'appel : log dans la console.

Authentification de l'utilisateur

- Faire en sorte d'ajouter au prénom du collaborateur mémorisé au niveau de l'EJB l'adresse mémoire hexadécimale de l'instance utilisée.
- Objectif : vérifier que l'application cliente converse bien toujours avec le même EJB dans une situation concurrentielle (plusieurs applications clientes lancées simultanément).

Surcharge d’annotation

- Remplacer le type @Stateful par un type @Stateless si possible sans modifier le code : passer par une écriture dans le descripteur de déploiement ejb-jar.xml

Intercepteur de méthode

- Créer un intercepteur de méthode qui affecte chaque fois à un projet (comptabilité analytique) en amont de la sauvegarde de la note de frais.
- Définir l'interception si possible sans annotation, par une écriture dans le descripteur de déploiement.
- Considérer cette fonctionnalité comme spécifique dans l'approche du packaging (où placer l'intercepteur ?).

Anticiper la suite

- Ce premier TP n'aborde pas la persistance. Nous parlerons de module « coreM » gérant des entités en mémoire.
- Dans le prochain TP, nous remplacerons ce module par un module « coreDB » gérant des entités en base de données.

Prévoir donc une architecture globale qui nous permette de mettre à jour notre système de gestion des entités sans avoir à intervenir dans le code des autres modules serveurs, ni dans celui de l'application cliente.