TD 5 : JPA annotation et Hibernate ORM framework

1. Préparer l'environnement de travail

Ajoutez dans votre projet Eclipse toujours la même librairie utilisée dans les séances de TD précédentes qui contient toutes les fichiers .jar nécessaires jusqu'à la fin du cours COO.

Pour faire fonctionner le programme de démonstration Hibernate du cours, créez deux bases de données dans MySQL nommées respectivement « inheritence » et « association ».

2. Rappel du programme de démonstration du cours

Ordre de packages à (ré)étudier : « inheritence » et puis « association ». Commencez par adapter les informations dans les fichiers de configuration XML, surtout le login / mot de passe de votre MySQL.

Les annotations JPA dans les classes à persister permettent de réaliser l'ORM. Ces classes doivent respecter la convention de **JavaBean**.

Les classes DataInit et DBConnection permettent de créer les tables relationnelles correspondant à l'ORM des classes à persister, et fournissent des connexions aux bases de données.

Pour « inheritence », l'ordre d'exécution des classes de test est « TestPersist » et puis « TestQuery ». A chaque fois que ces tests sont exécutés, la base de données est réinitialisée, avec les tables supprimées et recréées. Comprenez surtout le langage HQL (Hibernate Query Language), qui est un langage au niveau **d'objet** mais pas au niveau des tables relationnelles : **ce n'est pas SQL**.

Pour « association », l'ordre d'exécution des classes de test est « TestCascadePersist », « TestFetch » et « TestCascadeDelete ».

Note : les programmes dans les deux packages travaillent indépendamment sur les deux bases de données, ayant des configurations différentes dans les deux fichiers XML.

3. Programme « Bank Simulation » : nouvelle version avec Hibernate

Vous reprendrez la version précédente du programme « bank » (celle de JDBC). L'objectif est de réaliser la persistance avec Hibernate framework au lieu de JDBC. Vous devrez créer une nouvelle classe « HibernatePersistence » implémentant l'interface « StatisticPersistence ». Ensuite, le travail se réalisera en plusieurs étapes :

1) Annotation des classes à persister. Créez une nouvelle base de données (ex. « bank2 ») pour ce programme de nouvelle version. Les classes à annoter sont « SimulationEntry » et les classes client et opération. Attention, cette-fois ci, on persistera aussi les opérations des clients. Pensez à bien gérer l'héritage et les relations d'association. Ajoutez de **nouveaux attributs** dans ces classes si nécessaire. Par exemple, des collections pour faire « one-to-many » entre « SimulationEntry » et « AbstractClient ». Entre « AbstractClient » et « AbstractOperation », la relation doit être « one-to-one ». Vous avez aussi besoin d'ajouter un attribut dans la classe « AbstractClient » qui indique si le client est servi. N'oubliez pas de mettre à jour cette information lors de l'enregistrement d'un client (servi ou non servi), pour qu'elle soit persistée dans la base de données.

2) Réalisation des 3 méthodes fonctionnelles. Implémentez-les avec le mécanisme de persistance et des requêtes HQL. Adaptez le programme vis-à-vis de nouveaux besoins de la solution Hibenrate.

3) Testez votre programme. Changez dans le programme pour qu'il utilise « HibernatePersistence ». Ce switch doit être également fait par création des beans dans Spring. **Cela devra produire le même graphique que la séance précédente (JDBC) si tout fonctionne bien.**