Institut Galilée LEE
Année 2015-2016 Master 2

T.P. 4

Hibernate / Netbeans

L'objectif de ce TP est de mettre en place et d'utiliser le framework Hibernate dans une application Java de gestion de location de véhicules, voiture, camion ou moto. Il s'agit de gérer des individus et un parc de véhicules. L'application devra permettre, entre autres, de :

— de lister les véhicules du parc
— de lister les individus ayant loué un véhicule
— de permettre à des individus d'emprunter et de rendre des véhicules

Pour ce TP, l'environnement de développement utilisé est Netbeans, entre autre pour le fait que l'IDE intègre le moteur de base de données Derby. Cet environnement est disponible dans les salles de TP de Galilée. Pour le chapitre JDBC, il est nécessaire d'utiliser la base Derby intégrée à Glassfish. Dans tous les cas, le lecteur pourra trouver une version Linux de ces applications à cette adresse : `www-lipn.univ-paris13.fr/~fortier/Enseignement/Applications/`

Pour utiliser les outils de génération (configuration, Reverse Engineering) via l'IDE, il faut que le plugin "Hibernate" soit installé dans Netbeans. L'installation se fait simplement par le menu "Tools → Plugins".

1 Exercice

1. **Base de donnée** : On utilise une base de données derby, installée avec Glassfish. Avant tout, il faut créer une base de nom "location", utilisateur "location" et mot de passe "mdp" (par exemple). Les requêtes SQL de base à exécuter pour créer schéma et table :

```
create schema location;
create table location.personne (
    personne_id integer generated always as identity primary key,
    nom varchar(20),
    age numeric
);
insert into location.personne (nom,age) values ('michael', 25);
insert into location.personne (nom,age) values ('christophe', 30);
```

2. **Initialisation du projet** :
   — Créer une 'Web Application' sous Netbeans, sélectionner le framework 'Hibernate 4.3.1' (vérifier que la base de donnée est bien 'location').
   — Créer les fichiers de configuration pour Hibernate :
     — 'New → Other → Hibernate Configuration Wizard' s'il n'est pas déjà créé ! Il définit la connexion à la base de données (fichier hibernate.cfg.xml).
     — 'New → Other → Hibernate Reverse Engineering Wizard' puis sélectionner la table PERSONNE : Il définit les tables de la BD qui vont servir pour la génération de classes (hibernate.revenge.xml).
     — 'New → Other → Hibernate Mapping Files and POJOs from Database' : permet la génération de classes à partir des tables et le fichier de mapping classe/table (ici : Personne.java et Personne.hbm.xml).
     —

3. **Création et Test de méthodes** :
   — 'New → Unit Tests → Test for Existing Class', puis sélectionner la classe Personne. Cela crée une classe PersonneTest.java dans le répertoire Test Packages. Ajouter la méthode suivante

```
@Test
public void test_select(){
    String hql = "from location.Personne";
    List resultList=null;
    SessionFactory sf = new Configuration()
```

```
                .configure("hibernate.cfg.xml")
                .buildSessionFactory();
    Session session = sf.openSession();
    try {
        Transaction transaction=session.beginTransaction();
        Query q = session.createQuery(hql);
        resultList = q.list();
        transaction.commit();
    } catch (HibernateException he) {he.printStackTrace();}
    // Affichage des resultats
    for (Object o : resultList) {
        Personne personne = (Personne) o;
        System.out.println("nom : " + personne.getNom()
                    +" age : " + personne.getAge());
    }
```

Faire les imports nécessaires (attention : choisissez bien les imports de classes pour Hibernate!). Lancez l'exécution des tests (clic droit sur l'application web et choix de Test).

— Créer et tester les méthodes permettant d'ajouter de supprimer, de mettre à jour une personne (cf cours de M. Fortier)

4. **Relation One-to-One** : Créer dans la BD une table adresse :

```
create table location.adresse (
        personne_id integer,
        rue varchar(50),
        cp numeric,
        ville varchar(30),
        CONSTRAINT pkk primary key(personne_id)
);
```

— Créer les mappings d'objets (personne et adresse)
— Modifier la classe Personne pour déclarer la relation one-to-one avec la classe Adresse
— Créer une classe de test permettant :

(a) d'assurer le bon fonctionnement du lien bidirectionnel en fournissant une référence de l'objet Personne à l'instance de l'adresse. Ceci doit être fait manuellement car Hibernate ne prend pas en charge automatiquement les liens bidirectionnels.

(b) de sauvegarder la personne dans la base de données

— Vérifier les insertions et les correspondances entre les clés primaires

5. **Relation Many-to-Many** : Créer une table vehicule ainsi qu'une table de jointure avec celle des personnes :

```
create table location.vehicule (
        vehicule_id integer generated always as identity primary key,
        marque varchar(20),
        kilometrage numeric,
        modele varchar(20)
);
create table location.personne_vehicule (
        vehicule_id integer not null,
        personne_id integer not null,
        CONSTRAINT pk primary key(vehicule_id,personne_id)
);
```

— Modifier le mapping Personne précédent pour déclarer la relation many-to-many avec la classe Vehicule
— Générer le mapping Vehicule et la classe associée
— Créer une classe de test permettant de vérifier qu'une personne peut louer plusieurs véhicules simultanément