Travaux Dirigés de JEE n°4

Cours de Master 2 Informatique

——

Relation entre objets, HQL et Hibernate avancé

L'ensemble du code, des noms de classes, des champs, des méthodes doivent être en anglais. Le code doit répondre aux normes SUN et être obligatoirement testé unitairement.

Le rendu du TD doit être fait par mail à l'adresse suivante : `loyaute@univ-mlv.fr`. Il s'agit de rendre un fichier zip constitué d'un rapport au format pdf et du code source dans un jar non exécutable

---

▸ **Exercice 1.** *Relation entre un Auteur et un Livre*

Nous allons nous intéresser à la relation existante entre un auteur et un livre. Nous allons considérer ici qu'un auteur peut écrire plusieurs livres. Nous n'allons pas nous intéresser au fait qu'un livre puisse être écrit par plusieurs auteurs !

Nous considérons donc une relation @OneToMany d'un auteur vers ses livres. Vous allez réaliser cette relation. Pour se faire, plusieurs paramètres sont à rajouter dans l'annotation @OneToMany :

- *mappedBy qui indique quel champ est la clé étrangère ;*
- *fetch qui permet de définir la stratégie (paresseuse ou non) de la remontée des objets ;*
- *cascade qui indique quelles opérations (création, modification, suppression) doivent être réalisées en cascade sur la table fille.*

L'annotation @ManyToOne doit être rajoutée dans la classe Book pour permettre de réaliser cette relation.

Enrichissez le fichier import.sql et réalisez les tests unitaires pour vérifier vos réalisations !

▸ **Exercice 2.** *Hibernate Query Language*

Dans les classes DAO que vous aviez réaliser, rajoutez des méthodes permettant de :

- *Trouver un auteur en fonction de son numéro de sécurité social ;*
- *Trouver un livre en fonction de son ISBN ;*
- *Trouver l'ensemble des livres d'un auteur ;*
- *Trouver les auteurs nés à une date donnée ;*
- *Trouver les auteurs décédés à une date donnée ;*
- *Trouver les livres parus à une date donnée.*

Dans un premier temps, rajoutez dans vos classes existantes les champs manquants. Dans la classe Book, une date de parution à l'aide de la libraire joda-time, une chaîne de caractères représentant l'ISBN du livre. Dans la classe Author, une date de naissance, une date de décés à l'aide de la libraire joda-time et une chaîne de caractères représentant son numéro de sécurité sociale.

Maintenant, pour réaliser ces méthodes vous allez utiliser HQL, le langage de requetage d'Hibernate. Ce langage très proche du SQL utilise les mêmes mots clés. Cependant, contrairement au SQL, HQL se situe au niveau objet ! Les noms des tables sont donc remplacés par les noms des objets et les noms des colonnes par les noms des champs des classes. Afin de réaliser vos requêtes, vous allez utiliser la méthode createQuery fournie par les DAOs génériques (projets umlv-dao et umlv-hibernate que vous aviez télécharger précédemment).

Il est important, pour éviter l'injection de SQL dans votre code de ne pas réaliser de concaténation des paramètres de méthodes dans votre requête. Vous allez donc utiliser la méthode setParameter de l'interface Query. Ce qui donne par exemple :

```
Query query = createQuery("SELECT b FROM TOTO b WHERE b.isbn=:isbn");
query.setParameter("isbn", paramMethodeISBN);
```

▸ **Exercice 3.** *Différence entre NamedQuery et Query*

- *Quelle différence(s) existe t-il entre NamedQuery et Query ?*
- *Déduisez-en le ou les cas d'utilisation(s) d'une NamedQuery ;*
- *Idem pour Query.*

▸ **Exercice 4.** *Criteria Hibernate*

Nous allons maintenant nous intéresser au Criteria Hibernate. La méthode permettant de faire une requête avec l'API Criteria est disponible dans la DAO générique. Réécrivez les méthodes précédentes (exercice sur HQL) avec l'API Criteria d'Hibernta

- *Quels sont les différents types de Criteria ?*
- *Doit-on toujours utiliser les Criteria ?*

Comparez les performances avec JAPEX d'une méthode écrite avec HQL et la même méthode écrite à l'aide de l'API Criteria.

▸ **Exercice 5.** *Héritage de classes*

Il existe trois grandes stratégies pour représenter l'héritage et le mapping Objet-Relationnel :

- *TABLE_PER_CLASS ;*
- *SINGLE_TABLE ;*
- *JOINED.*

Explicitez les différences entre ces trois stratégies.

Comment peut-on partager des propriétés entre deux classes via une classe mère sans inclure cette classe mère comme entité (i.e. existence d'une table sous-jacente) :

- *Quel est l'intérêt ?*
- *Quel est le comportement sous-jacent ?*

▸ **Exercice 6.** *LAZY vs EAGER*

- *Expliquez la différence de comportement entre LAZY et EAGER ;*
- *A l'aide de JAPEX, réalisez la comparaison d'un chargement de 100000 auteurs (n'oubliez pas les livres qui sont associés) avec un chargement LAZY et un chargement EAGER. Mettez dans votre rapport les résultats que vous obtenez.*

▸ **Exercice 7.** *Cache de second niveau*

- *Citez les deux méthodes permettant de configurer le cache de second niveau Hibernate ;*
- *Donnez les avantages et inconvénients de chacune des deux méthodes ;*
- *Quel est l'intérêt d'avoir un cache de second niveau ?*
- *Existe t-il plusieurs fournisseurs de cache de second niveau ? Si oui, citez les fournisseurs.*
- *Explicitez les différents paramètres permettant de configurer le cache de second niveau d'Hibernate.*