Master Mathématiques, Informatique, Décision, Organisation (MIDO)
2ère année
Spécialités ID, MIAGE-IF et MIAGE-SITN et MIAGE-IF App.

ANNEE 2014 / 2015

Désignation de l'enseignement : Persistance des objets et bases de données relationnelles (*Object Relational Mapping*)

Désignation du document : **TP Hibernate**

Enseignement assuré par : Maude Manouvrier

Exécution et test du tutorial Hibernate

L'objectif de ce TP est de commencer à vous familiariser avec Hibernate, ceci à l'aide du tutorial fourni avec la documentation d'Hibernate. La base de données sera créée et manipulée via l'interface PgAdmin3 de PostgreSQL. La programmation se fera sous Eclipse.

Les documents et fichiers nécessaires au bon déroulement de ce TP sont disponibles à l'adresse : **`http://www.lamsade.dauphine.fr/~manouvri/HIBERNATE/TP_HIBERNATE/TP_TutorialHibernate.html`**
Ils sont également disponibles sur mycourse (`https://mycourse.dauphine.fr/`) sous le nom : M2 M2 MIAGE ID/IF/IF-APP/SITN_2014-2015_Persistance objet-relationnel / Hibernate_Maude Manouvrier

1. Récupération du tutorial Hibernate

Les fichiers d'extension `.java` et `.xml` du tutorial Hibernate sont stockés dans les fichiers zippés `Tutorial_Hibernate.zip` ou `Tutorial_Hibernate.tar.gz` disponible sur *mycourse* ou à l'adresse suivante :

`http://www.lamsade.dauphine.fr/~manouvri/HIBERNATE/TP_HIBERNATE/`

Une description de ce tutorial est disponible à l'adresse :
`http://docs.jboss.org/hibernate/core/3.6/reference/fr-FR/html_single/#tutorial` (en français)

`http://docs.jboss.org/hibernate/orm/4.2/manual/en-US/html_single/#tutorial` (en anglais)

2. Installation et exécution du Tutorial

1. Recopier le tutorial dans votre répertoire.
2. Créer un nouveau projet Java sous *Eclipse* en créant le projet à partir de sources existantes (*create project from existing source*), i.e. à partir du répertoire où vous avez recopié les fichiers du tutorial.
3. Ajouter dans l'onglet Librairies (*Add External JARs*) les fichiers .jar suivants :
   - Les fichiers `.jar` contenu dans le fichier : **`http://www.lamsade.dauphine.fr/~manouvri/HIBERNATE/TP_HIBERNATE/lib_Hibernate4.3.7.tar.gz`** ou dans le répertoire d'installation d'Hibernate si vous l'avez installé sur votre machine.
   - Le fichier `.jar` correspondant au pilote JDBC du SGBD avec lequel vous allez travailler (cf. `http://jdbc.postgresql.org/download.html#jars`).
4. Ne pas choisir la perspective Java (en cliquant non dans la fenêtre apparaissant à la fin de la création du projet Java – ou sinon dans le Menu *Window, Open Perspectives* d'Eclipse, sélectionner *Hibernate Console*)
5. Ouvrir le fichier **`hibernate.cfg.xml`**, situé dans le répertoire `src` de votre projet (fenêtre *Package Explorer* à gauche).

   Attention : le fichier transmis sur la page Web précédemment citée est configuré pour utiliser le SGBD PostgreSQL. Le fichier par défaut du tutorial est configuré pour utiliser hsqldb[1]. Si vous utilisez un autre SGBD, vous devez mettre à jour en conséquence les propriétés correspondantes (`connection.driver_class` et `dialect`).

[1] http://hsqldb.org/

6. Mettre à jour les propriétés suivantes du fichier `hibernate.cfg.xml` :
   - `connection.driver_class` : en indiquant le pilote JDBC du SGBD utilisé. Au CRIO Unix, vous utiliserez PostgreSQL. La valeur de cette propriété doit donc être : `org.postgresql.Driver`
   - `connection.url` : en remplaçant le nom de la base de données (ex. `BDTest1` ou `hsql`) par le nom de votre base de données. Au crio UNIX, vous devez indiquer : `jdbc:postgresql:// postgres.crio.dauphine.fr/nom_base` (où `nom_base` correspond au nom de la base de données que vous avez créée lors du TP JDBC).
   - `connection.username` : en y indiquant le login.
   - `connection.password` : en y indiquant votre mot de passe (il faut au préalable ajouter `AddChild #PCDATA` pour ajouter le mot de passe si vous êtes en mode *Design* – sauf si vous êtes en mode *Source*).
   - `dialect` : en indiquant le dialecte SQL utilisé pour le SGBD choisi. Ex. `org.hibernate.dialect.PostgreSQLDialect`
7. Dans le répertoire `events` (fenêtre *Package Explorer* à gauche), sélectionner le fichier `EventManager.java` et avec le bouton droit de la souris sélectionner le menu *Run* …(

) puis *Java Application*.

   Dans la fenêtre d'exécution – onglet *Main*, sélectionner le nom de votre projet (`events.EventManager` doit apparaître dans *Main class*).

   Dans l'onglet Arguments, inscrire `store` et cliquer sur le bouton *Run*.
8. Si tout a bien été configuré, des messages doivent apparaître dans la console d'Hibernate (fenêtre du bas) vous indiquant en particulier les requêtes SQL effectuées.

   Attention : Si vous obtenez l'exception suivante

   ```
   WARN SessionFactoryObjectFactory:123 - Could not unbind
   factory from JNDI
   javax.naming.NoInitialContextException: Need to specify class name in
   environment or system property, or as an applet parameter, or in an
   application resource file:  java.naming.factory.initial
   ```

   Dans ce cas, enlever `name=""` de la balise XML `<session-factory>` dans le fichier de configuration **`hibernate.cfg.xml`**
9. Vérifier après exécution que les quatre relations ont correctement été créées dans votre base de données et qu'un nuplet a été inséré dans la relation *events* (via l'utilitaire *pgadmin3* – Si vous ne connaissez pas cet outil, vous pouvez consulter l'annexe sur polycopié `http://www.lamsade.dauphine.fr/~manouvri/BD/Poly_TP_PostgreSQL.pdf`).

3. Travail à effectuer

L'objectif de ce TP est que vous appreniez à pratiquer Hibernate, que vous compreniez, par l'exemple, ce qui a été expliqué en cours[2] et que vous appreniez également à chercher seul les réponses à vos questions. Penser par conséquent à consulter l'aide en ligne de Hibernate :
`http://www.hibernate.org/docs`

[2] Rappel les transparents de cours sont disponibles sur *Mycourse* ou à l'adresse :
`http://www.lamsade.dauphine.fr/~manouvri/HIBERNATE/SLIDES/`

1. Analyser les programmes du tutorial pour bien les comprendre. En particulier analyser les POJO (`Person.java` et `Event.java`) ainsi que les fichiers de correspondance (`Person.hbm.xml` et `Event.hbm.xml`).

   Pour cela, penser à regarder la documentation en ligne du tutorial **http://docs.jboss.org/hibernate/core/3.6/reference/fr-FR/html_single/#tutorial (en français)**

   http://docs.jboss.org/hibernate/orm/4.2/manual/en-US/html_single/#tutorial (en anglais)

2. Analyser le schéma de la base de données créée lors de l'exécution précédente (étape 7 de la section 2). En particulier, regarder la séquence créée par Hibernate dans la base de données.

3. Exécuter à nouveau le programme avec le paramètre `store`, en ayant préalablement modifié la classe `EventManager`, de telle sorte que l'identificateur (attribut `id`) de l'objet de la classe `Event` soit affiché avant et après que l'objet ait été rendu persistant.

4. Tester les autres arguments du programme `EventManager.java` en étudiant bien les messages affichés dans la console et/ou en mettant à jour en conséquence le fichier de configuration (ex. propriété `hbm2ddl.auto` pour ne pas que la base de données soit re-créée à chaque exécution).

5. Modifier la méthode `createAndStorePerson()` de la classe `EventManager` de telle sorte que l'identificateur (attribut `id`) de l'objet `thePerson` soit affiché avant et après que l'objet ait été rendu persistant. **Que pouvez-vous en déduire sur la séquence utilisée par Hibernate ?**

6. Implémenter les méthodes `equals()` et `hashCode()` des deux POJO et tester les.

   Rappelez-vous que ces méthodes doivent utiliser l'égalité des clés métiers et non pas celle des les clés artificielles (cf. transparents de cours ou Section 4.3 de la documentation en ligne d'Hibernate).

7. Les questions et étapes suivantes ont pour objectif de vous permettre de mieux comprendre la méthode `load()` et la manipulation des objets détachés :

   a. Après avoir implémenté les méthodes `equals()` et `hashCode()` des deux POJO, tester à nouveau le paramètre `addpersontoevent`.
      Pourquoi obtenez-vous une erreur ?

   b. Ajouter l'instruction `String title = anEvent.getTitle();` juste avant l'instruction validant la transaction (i.e. avant l'instruction `session.getTransaction().commit();`) :

      Pourquoi n'y a-t-il plus d'erreur ?

   c. Mettre en commentaire l'instruction précédemment ajoutée et mettre en commentaire l'instruction
      `Event anEvent = (Event) session.load(Event.class, eventId);`
      de la méthode `addPersonToEvent()`

   d. Supprimer les commentaires de l'instruction située juste en dessous :
      `Event anEvent = (Event) session.createCriteria(…)`

   e. Exécuter à nouveau le programme avec le paramètre `addpersontoevent`.
      Pourquoi n'y a-t-il plus d'erreur ?

8. Les questions et étapes suivantes ont pour objectif de vous permettre de mieux comprendre la propriété `inverse` des fichiers de correspondance Hibernate :
    a. Mettre en commentaire toutes les instructions allant de `aPerson.getEvents().add(anEvent);` à la fin de la méthode et ajouter l'instruction `aPerson.addToEvent(anEvent);` juste avant la validation de la transaction (`session.getTransaction().commit();`).
    b. Remettre l'appel à `Criteria` entre commentaires et enlever les commentaires de l'appel à l'instruction `load()` pour l'objet `anEvent`.
    c. Supprimer la propriété `inverse="true"` de la balise `<set name="participants" …>` du fichier `Event.hbm.xml`.
    d. Exécuter à nouveau le programme avec le paramètre `addpersontoevent`. **Pourquoi obtenez-vous une erreur de la base de données ?**
    e. Remettre la propriété `inverse="true"` de la balise `<set name="participants" …>` du fichier `Event.hbm.xml`.
    f. Exécuter à nouveau le programme avec le paramètre `addpersontoevent`. **Pourquoi n'obtenez-vous pas d'erreur cette fois-ci ?**
9. Modifier la classe `Person` et le fichier de correspondance de la classe en remplaçant l'attribut `age` par un attribut `birthday` de type `Date`. Modifier également le programme `EventManager.java` en conséquence.
10. Les questions et étapes suivantes ont pour objectif de vous permettre de modifier la classe `EventManager` de telle sorte que l'affichage d'un événement affiche également le nom et le prénom des participants à cet événement. L'objectif final est ici de mieux comprendre le chargement des objets dans Hibernate.
    a. Dans les instructions correspondant au paramètre `list` de la classe `EventManager` : **Dans quel état est l'objet `theEvent`?**
    b. Ajouter les instructions suivantes à la fin des instructions correspondant au paramètre `list` de la classe `EventManager` :

```
Set participants = theEvent.getParticipants();
Iterator it = participants.iterator();
while (it.hasNext()) {
   Person thePerson = (Person) it.next();
         System.out.println("Firstname: " +
         thePerson.getFirstname()+
         " Lastname: " + thePerson.getLastname());
}
```

    Pourquoi obtenez-vous une exception ?

c. **Que faut-il changer dans le fichier de correspondance de la classe `Event` pour que le code ci-dessus ne génère plus d'erreur ?**

d. Sans modifier le fichier de correspondance de la classe `Event` : **Que faut-il modifier dans la méthode `listEvents()` de la classe `EventManager` pour que le code ci-dessus ne génère plus d'erreur ? -** Astuce : regarder ce que fait la méthode statique `Hibernate.initialize()`.

e. En mettant en commentaire les code ajouté précédemment dans la méthode `listEvents()` de la classe `EventManager` : **Quelles instructions pourriez-vous ajouter au code ci-dessus pour qu'il ne génère plus d'erreur ? -** Astuce : regarder ce que fait la méthode `lock()`.

11. Modifier le programme et le fichier de configuration de telle sorte que le login et le mot de passe de connexion à la base de données soient saisis par l'utilisateur au clavier (éventuellement dans une fenêtre), et soient transmis à Hibernate (et n'apparaissent donc plus en clair dans le fichier de configuration).
12. Créer une nouvelle classe `Activity` modélisant le fait qu'un événement soit composé de plusieurs activités (l'association ne sera implémentée que de manière uni-directionnelle pour ne pas modifier la classe `Event` et donc le fichier de *mapping* correspondant). Une activité est caractérisée par un type et un nom (unique). Une activité est également organisée par une personne (là encore l'association ne sera implémentée de manière uni-directionnelle pour ne pas modifier la classe `Person` et la relation correspondante).
13. Créer le fichier de correspondance de la classe `Activity` et modifier le fichier de configuration.
14. Modifier le programme de telle sorte qu'il permette :
    a. De créer une activité pour un événement donné (récupéré à partir de la base de son identificateur qu'on supposera connu de l'utilisateur) et un organisateur donné (récupéré à partir de la base de son identificateur qu'on supposera connu de l'utilisateur). Lors de la création d'une activité, si l'organisateur n'est pas parmi les participants de l'événement, le programme fera en sorte de l'y ajouter.
    b. De lister les activités d'un évènement donné (récupéré à partir de la base de son identificateur qu'on supposera connu de l'utilisateur).

4. Utilisation de métadonnées de type annotations

L'objectif de cette partie est de vous faire manipuler les annotations JPA en transformant les métadonnées XML du tutoriel en annotations JPA.

Pour cela, effectuer les étapes suivantes :

1. Modifier le fichier de configuration `hibernate.cfg.xml` en remplaçant `<mapping resource …>` par `<mapping class …>` pour chacune des classes persistantes du tutoriel.
2. Importer `org.hibernate.cfg.*` dans le fichier `HibernateUtil.java` et modifier le en remplaçant :
   `sessionFactory = new Configuration() …`
   par `sessionFactory = new` **`Annotation`**`Configuration()…`
3. Importer `javax.persistence.*` dans les POJO pour pouvoir utiliser les annotations.

4. En vous aidant de la documentation *Hibernate Annotation* (**`http://www.hibernate.org/docs`** - Documentation du noyau pour la version 3.6) et des fichiers de métadonnées XML des classes persistantes du tutoriel : remplacer les métadonnées XML des classes persistantes par des annotations.

   NB : Il est inutile de supprimer les fichiers `Event.hbm.xml` et `Person.hbm.xml` qui ne seront plus utilisés si vous avez correctement mis à jour le fichier de configuration `hibernate.cfg.xml`.

5. Tester les différents paramètres du programme `EventManager.java` une fois les annotations créées.

Astuces :

Il existe une annotation spécifique à Hibernate (i.e. non définie dans la norme JPA) permettant de définir des collections de valeurs, dont la syntaxe est la suivante :

```
@org.hibernate.annotations.CollectionOfElements
   @JoinTable(
                  name="nom_table",
                joinColumns = @JoinColumn(name="nom_attribut_jointure")
              )
   @Column(name="nom_colonne_element", nullable=false)
```

Attention : Il faut penser à bien préciser le type des éléments de la collection dans la définition du `Set` et dans le type retour de la méthode `Get` permettant d'accéder à la collection de valeurs (sinon Hibernate va générer une exception indiquant que le type des éléments de la collection ne peut pas être identifié).

5. Utilisation de l'*EntityManager*

L'objectif de cette partie est de vous faire manipuler *Hibernate Entity Manager* en utilisant le tutorial d'*Hibernate Core* (avec les métadonnées exprimées par des annotations) :

Pour cela, effectuer les étapes suivantes :

6. Importer `javax.persistence.*` et `org.hibernate.ejb.*` dans le fichier `HibernateUtil.java` et modifier le en remplaçant :
7. En vous aidant de la documentation *Hibernate Annotation* (**`http://www.hibernate.org/docs`**) : remplacer les appels à `Session` par des appels à `EntityManager` ainsi que les appels aux méthodes propres à `Session` par des appels aux méthodes d'`EntityManager`.
8. Tester les différents paramètres du programme `EventManager.java` une fois les annotations créées.

NB : Vous n'avez à modifier que les fichiers `HibernateUtil.java` et `EventManager.Java`. De plus, vous pouvez utiliser le fichier de configuration initial du tutorial `hibernate.cgf.xml` sans créer de fichier `persistence.xml`.

6. Annexe – Outils nécessaires pour réaliser ce TP chez vous

Pour réaliser ce TP sur votre propre machine, vous devez avoir installé les logiciels et *plugins* suivants :

- Eclipse (`http://www.eclipse.org/downloads/`)
- Hibernate (`http://www.hibernate.org/downloads`)
- *Hibernate Tools* (`http://www.hibernate.org/subprojects/tools.html`)
- Un SGBD (ex. PostgreSQL - `http://www.postgresql.org/` et son interface PgAdmin3 - `http://www.pgadmin.org/`)
- Un pilote JDBC (cf. `http://jdbc.postgresql.org/download.html#jars`)