Objectifs

Savoir mettre en place et manipuler un ORM basé sur JPA

JPA s'appuie sur la notion de POJO managés par un EntityManager. c'est l'annotation @Entity qui indique que le POJO sera bien managé. Le manager s'appuie sur une unité de persistance définie dans `persistence.xml`. Ce fichier peut contenir plusieurs unités de persistance, chacune avec un nom symbolique.

```
<persistence-unit name="pu" transaction-type="RESOURCE_LOCAL">
```

Dans chaque classe utilisant un POJO, il est nécessaire de récupérer une instance de l'EntityManager. Sur cette instance, il est alors possible de demander la persistance d'une instance (`persist`), de retrouver un objet (`find`) ou de créer une requête (`createNamedQuery` ou `createNamedQuery`). Ces différentes opérations doivent être impérativement exécutées en mode transactionnel (entre `begin` et `commit`).

1 Partie 1

Dans cette première partie, pour bien débuter, **on ne s'occupe que d'une seule table**, simple et sans clé étrangère, par exemple une table `annuaire(num,nom,prenom,sexe,tel,fonction)`. On souhaite écrire un programme Java classique avec un simple `main` en cinq parties : créer quelques objets (`persist`) , retrouver un objet via sa clé (`find`)), modifier cet objet (se fait automatiquement), afficher la collection de tous les objets créés (définir une `namedQuery` findAll), détruire un des objets (`remove`) et ré-affiche à nouveau l'ensemble pour vérification.

Pour cela, vous allez tester trois approches :

1.1 JPA directement sous eclipse (ou autre IDE)

1.1.1 Generate Entities from tables

1. Lancez Eclipse et créez un projet JPA (Fire / new / JPA Project) nommé **tpjpa111**
2. suivre les explications détaillées au dos
3. JPA Tools / Generate tables from entities
4. Ajoutez une classe `Test.java` avec un simple main

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("tpjpa");
EntityManager em = emf.createEntityManager();
Client client = new Client();
...
em.getTransaction().begin();
em.persist(client);
em.getTransaction().commit();
```

5. Améliorer ce `main` pour qu'il se décompose en 5 parties conformément à ce qui est cité dans le paragraphe d'introduction.

Arborescence Eclipse

```
tpjpa111
|-- build
|   |-- classes
|     |-- META-INF
|     |   |-- persistence.xml
|     |-- model
|         |-- Client.class
|         |-- Test.class
|-- src
    |-- META-INF
    |   |-- persistence.xml
    |-- model
        |-- Client.java
        |-- Test.java
```

1.1.2 Generate tables from entities

1. Lancez Eclipse et creez un projet JPA (File / new / JPA Project) nommé **tpjpa112**
2. suivre les explications détaillées au dos
3. créez un POJO (d'une table inexistante) avec les annotations `@Entity` et `@Id`
4. JPA Tools / Generate tables from entities
5. Recopiez la classe `Test.java` précédente qui doit fonctionner quasi telle qu'elle.

1.2 JPA avec Maven

1. Créer un projet Maven basé sur l'archetype (`mvn archetype:generate`, `maven-archetype-quickstart`, groupID=`fr.da2i`, artifactId=`tpjpa12`)
2. Ajoutez dans le pom les dépendances nécessaires

```
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>eclipselink</artifactId>
    <version>2.7.4</version>
</dependency>
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>2.2.1</version>
</dependency>
```

3. Réalisez une copie du projet précédent. Notamment, placez les fichiers précédents dans les bons répertoire de l'arborescence Maven (Attention : le fichier `persistence.xml` doit impérativement être placé dans `main/resources/META-INF`).
4. éventuellement importez ce projet dans Eclipse

Arborescence Maven

```
tpjpa12
|-- pom.xml
|-- run.sh
|-- src
    |   main
    |   |   java
    |   |   |-- fr
    |   |       |-- da2i
    |   |           |-- Test.java
    |   |           |-- Client.java
    |   |-- resources
    |       |-- META-INF
    |       |   |-- persistence.xml
    |-- test
        |-- java
```

2 Partie 2

Modifier le TP WEB "Annuaire" précédent en remplaçant votre propre DAO par un EntityManager

3 Partie 3

On considère une BDD classique avec 3 tables permettant la gestion des commandes de l'entreprise à ses fournisseurs.

```
fournisseurs(fno,nom,ville)
produits(pno,libelle,prix)
commandes(cno,pno,fno,qute)
```

Ecrire une application web s'appuyant sur JPA qui contiendra notamment 3 pages : `ListerCommandes.jsp` qui affiche dans une table HTML, par fournisseur, les commandes associées. `ListerFournisseurs.jsp` qui affiche par numéro de fournisseur les numéros de commandes associées, et enfin `ListerCommandes.jsp` qui affiche dans une table HTML, la liste des commandes avec le nom du fournisseur et le libelle du produit associés.

4 A rendre

— Sur Moodle, l'archive ZIP de l'ensemble des 3 parties, chacune exécutable via un `mvn:exec:java` ou `mvn tomcat7:run` placé dans un shell `run.sh` placé à la racine de chaque projet

Aide Eclipse

1. Dans un premier temps assurez vous d'avoir les jars nécessaires : `postgresql.jar`, `eclipselink.jar` et `javax.persistence.jar` (fourni avec JavaEE mais pas avec JavaSE)
2. File / New JPA Project
   - indiquer un nom du projet
   - JPA version 2.1
   - Basic JPA config
3. Panneau JPA Facet
   - EclipseLink
   - type "Disable configuration"
   - Cliquer sur "Add Connection" si pas déjà existante
   - Sélectionner la connexion à la base
   - cocher 'Discover automatically"
4. Sur le nom du projet, colonne de gauche
   (a) Dépliez src/META-INF/persistence.xml
   (b) Onglet général : mettre un nom pour la P.U.
   (c) Onglet connection
   - Transaction : ressource local
   - Batch : JDBC
   - Populate from connection

   (d) Onglet Schema Generation
   - Choisir Database = drop-and-create

   (e) Visualiser le résultat sur l'onglet source. Sauvegardez le fichier (Cntl S). c'est un exemple de fichier de persistance à garder pour le futur
   (f) Sur le nom du projet : properties / Java Build Path
   - Ajouter les 3 packages `postgresql.jar`, `eclipselink.jar` et `javax.persistence.jar` au classpath

   (g) Sur le nom du projet / JPA Tools
   i. soit New / JPA Entity ; puis remplir son entity bean avec constructeur vide, les paramètres, les getters et setters et 2 tables impératifs : @Entity et @Id
   ii. soit New / JPA Entities from Tables
5. Si une erreur "Table cannot be resolved" survient : Generate Tables from Entities, puis Validate (et éventuellement Refresh)
6. Reste juste à écrire une classe avec un main pour créer les objets