Chapitre 5 : JPA and EclipseLink

Note

- Ce cours couvre les besoins les plus importants pour la persistance En Java (vous pourrez retrouver un cours très complet sur JPA2 et sur la persistance en java en général sur internet).
- Une bonne référence en ligne sur JPA2
  - http://www.objectdb.com/api/java/jpa, en particulier la section JPA2 annotations

La persistance par sérialisation

- Sérialisation = sauvegarde de l'état d'un objet sous forme d'octets.
- A partir d'un état sérialisé, on peut reconstruire l'objet
- En java, au travers de l'interface **`java.io.Serializable`**, des méthodes de **`java.io.ObjectInputStream`** et **`java.io.ObjectOutputStream`**

La persistance par sérialisation

- Défauts très nombreux…
  - Gestion des versions, maintenance…
  - Pas de requêtes complexes…
  - Ex : on sérialize mille comptes bancaires. Comment retrouver ceux qui ont un solde négatif ?
- Solution : stocker les objets dans une base de donnée!

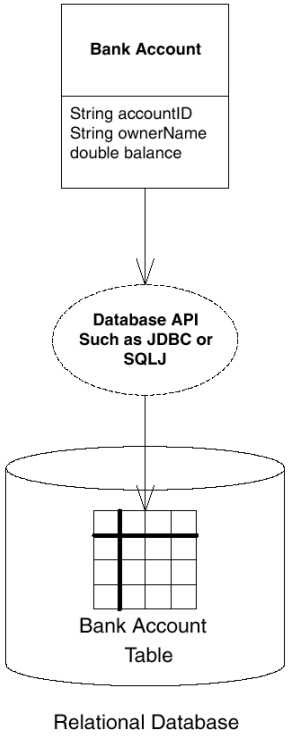
Il s'agit de : Object-Relational Mapping (ORM)

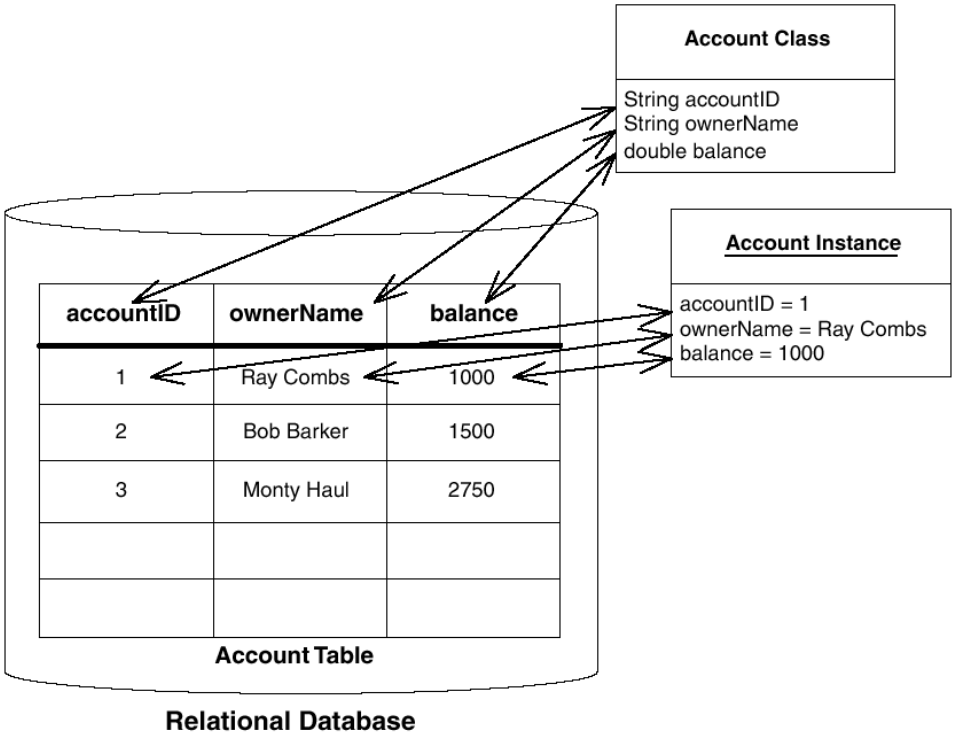
- The process of mapping Java objects to database tables and vice versa is called "Object-relational mapping" (ORM).
- The Java Persistence API (JPA) is one possible approach to ORM.
- JPA is a specification and several implementations are available.
- Popular implementations are **Hibernate**, **EclipseLink** and **Apache OpenJPA**.
- The reference implementation of JPA is EclipseLink !

Principe : Object-Relational Mapping (ORM)

- On stocke l'état d'un objet dans une base de donnée.
  - Ex : la classe Personne possède deux attributs **`nom`** et **`prenom`**, on associe cette classe à *une table* qui possède deux colonnes : **`nom`** et **`prenom`**.
- On décompose chaque objet en une suite de variables dont on stockera la valeur dans une ou plusieurs tables.
- Permet des requêtes complexes.

Exemple (compte bancaire): Object-Relational Mapping (ORM)





Qu'est-ce qu'un Entity Bean

- Ce sont des objets qui savent se *mapper* dans une base de donnée.
- Ils utilisent un mécanisme de persistance
- Ils servent à représenter sous forme d'objets des données situées dans une base de donnée
  - Le plus souvent un objet = une ou plusieurs ligne(s) dans une ou plusieurs table(s)

Qu'est-ce qu'un Entity Bean

- All entity classes must define a primary key, must have a non-arg constructor and or not allowed to be final. Keys can be a single field or a combination of fields.
- JPA allows to auto-generate the primary key in the database via the @GeneratedValue annotation.
- By default, the table name corresponds to the class name. You can change this with the addition to the annotation @Table(name="NEWTABLENAME").

Exemple avec un compte bancaire

A travers une entité « Compte_bancaire »

- On lit les informations d'un compte bancaire en mémoire, dans une instance d'une entity bean,
- On manipule ces données, on les modifie en changeant les valeurs des attributs d'instance,
- Les données seront mises à jour dans la base de données automatiquement !

Persistance des champs

- By default each field is mapped to a column with the name of the field. You can change the default name via @Column (name="newColumnName").
- The following annotations can be used.

Table 1. Annotations for fields / getter and setter	
@Id	Identifies the unique ID of the database entry
@GeneratedValue	Together with an ID this annotation defines that this value is generated automatically.
@Transient	Field will not be saved in database

Mapping Relationnel

- JPA allows to define relationships between classes, e.g. it can be defined that a class is part of another class (containment).
  - Classes can have one to one, one to many, many to one, and many to many relationships with other classes.
- A relationship can be bidirectional or unidirectional, e.g.
  - in a bidirectional relationship both classes store a reference to each other while in an unidirectional case only one class has a reference to the other class. Within a bidirectional relationship you need to specify the owning side of this relationship in the other class with the attribute "mappedBy", e.g. @ManyToMany(mappedBy="attributeOfTheOwningClass".

Mapping Relationnel

Relationship annotations:

- @OneToOne
- @OneToMany
- @ManyToOne
- @ManyToMany

Persistence units

- The persistence unit is described via the persistence.xml file in the META-INF directory of the source folder.
- A set of entities which are logical connected will be grouped via a persistence unit.
- The persistence.xml file defines the connection data to the database, e.g. the driver, the user and the password.

Entity Manager

- The EntityManager is created by the EntityManagerFactory which is configured by the persistence unit.
- The entity manager javax.persistence.EntityManager provides the operations from and to the database, e.g. find objects, persists them, remove objects from the database, etc.
- In a JavaEE application the entity manager is automatically inserted in the web application. Outside JavaEE you need to manage the entity manager yourself.
- Entities which are managed by an Entity Manager will automatically propagate these changes to the database (if this happens within a commit statement).
- If the Entity Manager is closed (via close()) then the managed entities are in a detached state. If synchronize them again with the database a Entity Manager provides the merge() method.