Mapping Objet/Relationnel Suite JPA

I. Mougenot (isabelle.mougenot@umontpellier.fr)

Faculté des Sciences Université Montpellier

2020

Retour sur la vision générale

Architecture à mettre en place

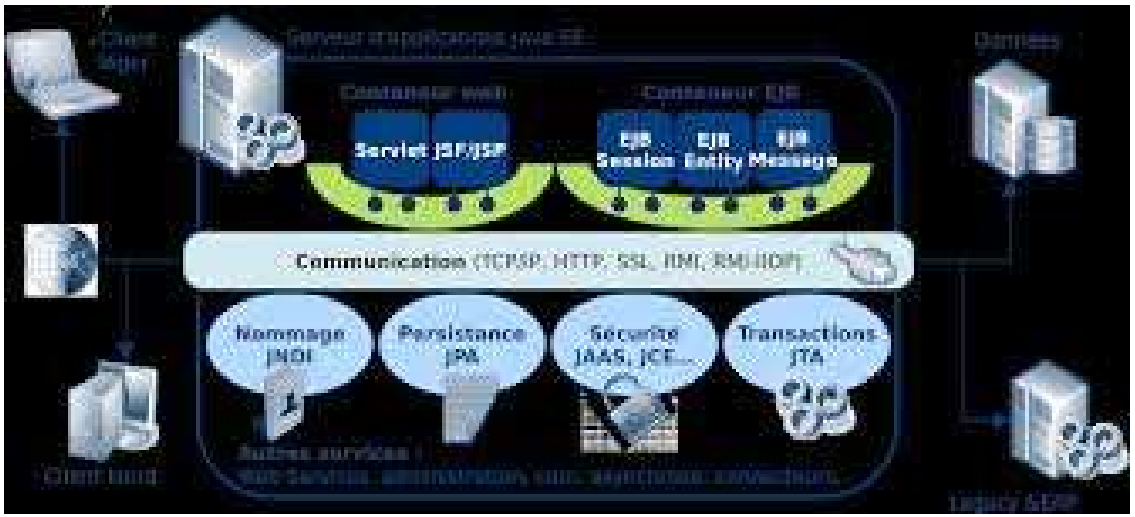

Figure: EJB Entity (extrait de `https://guillaume.piolle.fr/doc/devweb-javaEE.pdf`)

Les différentes activités

- BDR (Exemple avec MySQL) - schéma relationnel Lieu-Monument
- JBDC (Connectivité Java - BDR)
- JPA (gérer la persistance comme si il n'y avait pas de distorsion de modèle)
- Hibernate (une solution de conteneur possible pour aborder JPA)
- JPA API Query - A corréler au jeu de requêtes à construire
- EJB et POJO

Le cas de Monument -AssocieA- Celebrite

Comment traduire au niveau des Entity Java Beans, une relation plusieurs à plusieurs ?

Figure: Diagramme de classes partiel

Plusieurs cas possibles

Ici le plus simple, car AssocieA n'a pas de propriété propre
voir `http://orm.bdpedia.fr/jpamodel.html`

- PAS DE CLASSE AssocieA (n'est pas le reflet d'une entité) : Monument et Celebrite suffisent
- liens bidirectionnels avec des annotations @ManyToMany
- les rôles sont exploités pour définir les propriétés de type Collection
- Spécialiser les types de Collection : par exemple List

Schéma relationnel

- monument (<u>geohash varchar(12)</u>, nom varchar(80), proprietaire varchar(10), typeM varchar(20), longitude float, latitude float, codeInsee varchar(6));
- Celebrite(<u>numCelebrite integer</u>, nom varchar(16), prenom varchar(16), nationalite varchar(10), epoque varchar(6))
- AssocieA(<u>codeM varchar(12), numCelebrite integer</u>)
  avec AssocieA(codeM) $\subseteq$ monument(geohash)
  avec AssocieA(numCelebrite) $\subseteq$ Celebrite(numCelebrite)

Le cas de Acteur1 -Joue1- Film

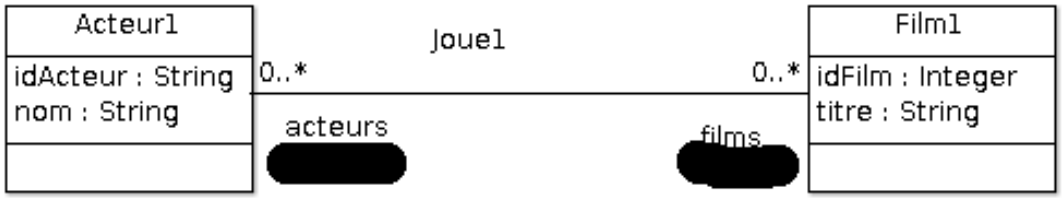

Figure: Diagramme de classes (inspiré de `http://orm.bdpedia.fr/jpamodel.html`)

Schéma relationnel et Joue1 qui traduit une association

- Acteur1(idActeur varchar(4), nom varchar(30))
- Film1(idFilm integer,titre varchar(50))
- Joue1(idFilm integer,idActeur varchar(4)) avec Joue1(idFilm) $\subseteq$ Film1(idFilm)
  avec Joue1(idActeur) $\subseteq$ Acteur1(idActeur)

Le cas de Acteur1 -Joue1- Film

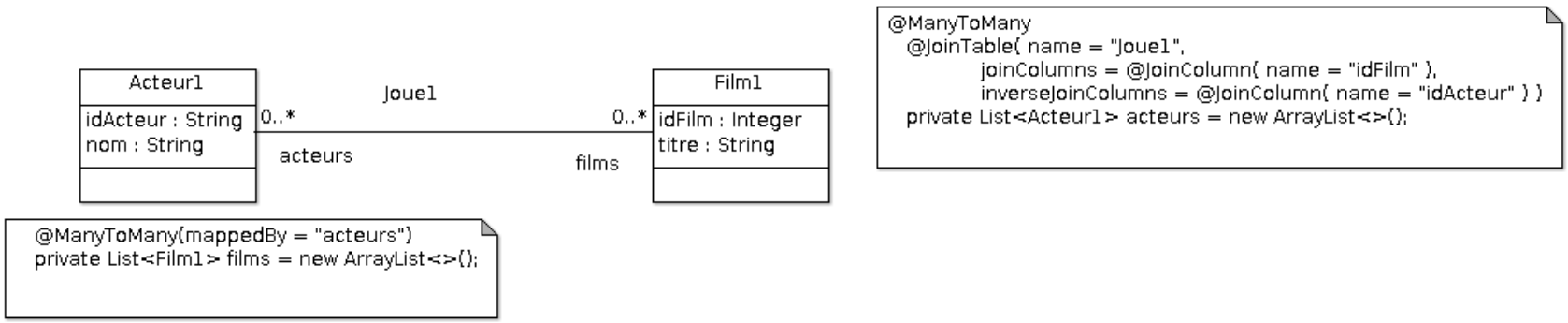

Figure: Diagramme de classes avec annotations

Classe de test (JPA Query)

```
EntityTransaction tx = em.getTransaction();
    tx.begin();
    Query query = em.createQuery("SELECT f FROM Film1 f ");
    System.out.println("Les films");

    List<Film1> films = (List<Film1>)query.getResultList();
    for(Film1 fl : films) {
     System.out.println(fl.getTitre());
     List<Acteur1> c2 = fl.getActeurs();
        for (Acteur1 p : c2)
        { System.out.println(" "+p.toString());}
    }
    tx.commit();
```

Listing 1: Requête sur film et acteurs associés