# LAZY

Artiste artiste = **new** Artiste ("puma", "guerin", "really good artiste");

Peinture p = **new** Peinture (dimension, **false**, artiste, 2010, "", "LOL", "bla", "", "", SupportOeuvre.*BOIS*, Realisation.*ACRYLIQUE*);

*oeuvreDAO*.createOeuvre(p);

produit les requêtes suivantes :

Hibernate:

insert

into

Oeuvre

(annee, artiste\_id, caracteristique, commentaire, hauteur, largeur, longueur, hasBeenReproduced, resume, tag, titre, realisation, support, DTYPE)

values

(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 'Peinture')

Hibernate:

insert

into

Artiste

(nom, prenom, bibliographie, id)

values

(?, ?, ?, ?)

Hibernate:

update

Oeuvre

set

annee=?,

artiste\_id=?,

caracteristique=?,

commentaire=?,

hauteur=?,

largeur=?,

longueur=?,

hasBeenReproduced=?,

resume=?,

tag=?,

titre=?,

realisation=?,

support=?

where

id=?

🡺 L’œuvre est insérée. Etant donné qu’on est en cascade.Persist, l’artiste est persisté également. Etant donné que l’œuvre a été persistée avant l’artiste et qu’elle possède une clé étrangère dessus (à ce moment là l’artiste n’a pas d’id), il faut mettre à jour (update) l’œuvre avec la clé étrangère de l’artiste.

Ensuite on fait un test , pour rappel la classe œuvre a un attribut :

@ManyToOne (cascade=CascadeType.*PERSIST*, fetch=FetchType.*LAZY*)

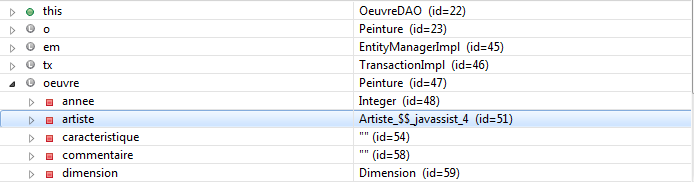
@JoinColumn(name="artiste\_id")

**private** Artiste artiste;

* Nous allons voir l’effet de FetchType.Lazy

En mode debug, regardons l’effet de :

Oeuvre oeuvre = em.find(Oeuvre.**class**,o.getId());



On a chargé tous les attributs de l’œuvre sauf l’artiste (qui contient lui-même une liste d’œuvre)

🡺Il ne serait pas judicieux de télécharger toutes les autres œuvres de l’artiste afin d’avoir le graphe complet.

On voit artiste : Artiste\_$$\_javassist\_4 (id=51) : il s’agit d’un proxy.

Ainsi quand on voudra accéder à l’artiste de l’œuvre par :

*LOG*.info("info : "+ oeuvre.getArtiste().getNom());

Les informations sont correctement récupérées depuis la base de données.

Hibernate:

select

artiste0\_.id as id22\_0\_,

artiste0\_.nom as nom22\_0\_,

artiste0\_.prenom as prenom22\_0\_,

artiste0\_.bibliographie as bibliogr1\_25\_0\_

from

Artiste artiste0\_

where

artiste0\_.id=?

Les données sont téléchargées de manière transparente. A chaque fois qu’on voudra récupérer un élément de la liste une requête est générée sur la base de données.

🡺Mettre toutes les relations ManyToOne en LAZY

La relation ToMany est en LAZY par défaut

# EAGER

Prenons en exemple :

@OneToMany(mappedBy="artiste", orphanRemoval=**true**, cascade={CascadeType.*PERSIST*, CascadeType.*REMOVE*}, fetch=FetchType.*EAGER*)

**private** List<Oeuvre> oeuvres = **new** ArrayList<Oeuvre>();

Si on fait :

Artiste a = em.find(Artiste.**class**,artiste.getId());

Hibernate:

select

artiste0\_.id as id45\_1\_,

artiste0\_.nom as nom45\_1\_,

artiste0\_.prenom as prenom45\_1\_,

artiste0\_.bibliographie as bibliogr1\_48\_1\_,

oeuvres1\_.artiste\_id as artiste16\_45\_3\_,

oeuvres1\_.id as id3\_,

oeuvres1\_.id as id44\_0\_,

oeuvres1\_.annee as annee44\_0\_,

oeuvres1\_.artiste\_id as artiste16\_44\_0\_,

oeuvres1\_.caracteristique as caracter4\_44\_0\_,

oeuvres1\_.commentaire as commenta5\_44\_0\_,

oeuvres1\_.hauteur as hauteur44\_0\_,

oeuvres1\_.largeur as largeur44\_0\_,

oeuvres1\_.longueur as longueur44\_0\_,

oeuvres1\_.hasBeenReproduced as hasBeenR9\_44\_0\_,

oeuvres1\_.resume as resume44\_0\_,

oeuvres1\_.tag as tag44\_0\_,

oeuvres1\_.titre as titre44\_0\_,

oeuvres1\_.support as support44\_0\_,

oeuvres1\_.materiaux as materiaux44\_0\_,

oeuvres1\_.realisation as realisa15\_44\_0\_,

oeuvres1\_.DTYPE as DTYPE44\_0\_

from

Artiste artiste0\_

left outer join

Oeuvre oeuvres1\_

on artiste0\_.id=oeuvres1\_.artiste\_id

where

artiste0\_.id=?

L’entity Artiste est récupérée ainsi que ses œuvres associées = Left Outer Join

Par contre quand on voudra accéder à une information relative à ses œuvres on n’aura plus besoin d’interroger la base car les données on déjà été rapatriées.

🡺Pas de requête supplémentaire générée.

*LOG*.info("info : "+ a.getOeuvres().get(0).getTitre());