Cave à vin (partie 03)

# TP03 du module 02 – Spring Data

|  |
| --- |
| Ce TP sera un TP fil rouge du cours   * Création d’une association OneToMany unidirectionnelle |

|  |
| --- |
| **Durée estimée** |
| 1 heure |

Solution

# Créer les entités

## Voici le diagramme des entités à réaliser :

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

Voici le diagramme des tables en base de données :

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

* Le package des classes est fr.eni.cave.bo.client
* Utiliser les annotations de JPA
  + Appliquer les annotations pour l’association OneToMany unidirectionnelle
  + Penser à préciser l’annotation @JoinColumn
* Les prix doivent être des float avec 2 décimales
  + Utiliser pour cela le paramètre precision=2 de l’annotation @Column
  + JAVADOC : int precision à (Optional) The precision for a decimal (exact numeric) column
* Utiliser les annotations de Lombok
  + Eviter que la liste soit nulle ; risque de NullPointerException
    - Utiliser l’annotation @Builder.Default
  + Pour les 2 entités, seul l’attribut id est utilisé pour la méthode equals

## Entité LignePanier

**package** fr.eni.cave.bo.client;

**import** jakarta.persistence.\*;

**import** lombok.\*;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

//equals sur l'attribut id

@EqualsAndHashCode(of = { "id" })

@ToString

@Builder

@Entity

@Table(name = "CAV\_LINE")

**public** **class** LignePanier {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "LINE\_ID")

**private** Integer id;

@Column(name = "QUANTITY")

**private** **int** quantite;

//précision à 2 décimales pour le prix

@Column(name = "PRICE", precision = 2)

**private** **float** prix;

}

## Entité Panier

* Avec l’association OneToMany vers LignePanier

**package** fr.eni.cave.bo.client;

**import** java.util.ArrayList;

**import** java.util.List;

**import** jakarta.persistence.\*;

**import** lombok.\*;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

//equals sur l'attribut id

@EqualsAndHashCode(of = { "id" })

@ToString

@Builder

@Entity

@Table(name = "CAV\_SHOPPING\_CART")

**public** **class** Panier {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "SHOPPING\_CART\_ID")

**private** Integer id;

@Column(name = "ORDER\_NUMBER", length = 200)

**private** String numCommande;

//précision à 2 décimales pour le prix

@Column(name = "TOTAL\_PRICE", precision=2)

**private** **float** prixTotal;

@Column(name = "PAID")

**private** **boolean** paye;

@OneToMany(cascade = CascadeType.***ALL***, orphanRemoval = **true**, fetch = FetchType.***EAGER***)

@JoinColumn(name="SHOPPING\_CART\_ID")

**private** @Builder.Default List<LignePanier> lignesPanier = **new** ArrayList<>();

}

* L’association entre Panier et LignePanier est une composition
  + Les 2 entités doivent avoir le même cycle de vie
  + Il faut donc imposer :
    - cascade = CascadeType.ALL
    - orphanRemoval = true,
    - fetch = FetchType.EAGER à pour remonter en 1 passe le panier et ses lignes

# Créer les Repository

**package** fr.eni.cave.dal;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** fr.eni.cave.bo.client.LignePanier;

**public** **interface** LignePanierRepository **extends** JpaRepository<LignePanier, Integer>{

}

**package** fr.eni.cave.dal;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** fr.eni.cave.bo.client.Panier;

**public** **interface** PanierRepository **extends** JpaRepository<Panier, Integer>{

}

# Créer des tests unitaires

* Une classe de tests unitaires appelée TestOneToManyUni vous est fournie
  + Copier la classe dans le package fr.eni.cave.association des tests Java
* Aidez-vous de cette méthode et des tests unitaires des démonstrations
* Créer les tests unitaires pour cette association
  + Créer des prix avec au moins 2 décimales pour bien montrer le comportement
  + L’application peut créer un nouveau panier à partir de sa première ligne
    - Le panier a un identifiant à nul et est vide
    - Seule la ligne a des informations
    - Il est possible d’ajouter une ligne à un panier existant
* Il faut 4 tests :
  + test\_save\_nouvelleLigne\_nouveauPanier
  + test\_save\_nouvelleLigne\_Panier()
  + test\_delete()
  + test\_orphanRemoval()

**package** fr.eni.cave.association;

**import** **static** org.assertj.core.api.Assertions.*assertThat*;

**import** **static** org.junit.jupiter.api.Assertions.*assertNull*;

**import** java.util.List;

**import** java.util.stream.Collectors;

**import** org.junit.jupiter.api.Test;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.test.autoconfigure.orm.jpa.\*;

**import** fr.eni.cave.bo.client.\*;

**import** fr.eni.cave.dal.PanierRepository;

**import** lombok.extern.slf4j.Slf4j;

@Slf4j

@DataJpaTest

**public** **class** TestOneToManyUni {

@Autowired

**private** TestEntityManager entityManager;

@Autowired

PanierRepository repository;

@Test

**public** **void** test\_save\_nouvelleLigne\_nouveauPanier() {

**int** qte = 4;

**final** LignePanier lp = LignePanier

.*builder*()

.quantite(qte)

.prix(qte \* 23.95f)

.build();

**final** Panier panier = **new** Panier();

panier.setPrixTotal(lp.getPrix());

// Association OneToMany

panier.getLignesPanier().add(lp);

// Appel du comportement

**final** Panier panierDB = repository.save(panier);

// Vérification de l'identifiant de l'entreprise

*assertThat*(panierDB.getId()).isGreaterThan(0);

// Vérification de la cascade de l'association

*assertThat*(panierDB.getLignesPanier()).isNotNull();

*assertThat*(panierDB.getLignesPanier()).isNotEmpty();

*assertThat*(panierDB.getLignesPanier().size()).isEqualTo(1);

***log***.info(panierDB.toString());

}

@Test

**public** **void** test\_save\_nouvelleLigne\_Panier() {

**final** Panier panierDB = panierEnDB();

**int** qte = 10;

**final** LignePanier lp = LignePanier

.*builder*()

.quantite(qte)

.prix(qte \* 23.95f)

.build();

panierDB.setPrixTotal(panierDB.getPrixTotal() + lp.getPrix());

// Association OneToMany

panierDB.getLignesPanier().add(lp);

// Appel du comportement

**final** Panier panierDB2 = repository.save(panierDB);

// Vérification de l'identifiant de l'entreprise

*assertThat*(panierDB2.getId()).isEqualTo(panierDB.getId());

// Vérification de la cascade de l'association

*assertThat*(panierDB2.getLignesPanier()).isNotNull();

*assertThat*(panierDB2.getLignesPanier()).isNotEmpty();

*assertThat*(panierDB2.getLignesPanier().size()).isEqualTo(2);

***log***.info(panierDB2.toString());

}

@Test

**public** **void** test\_delete() {

**final** Panier panierDB = panierEnDB();

**final** List<Integer> listeIdLignePanier = panierDB.getLignesPanier()

.stream()

.map(LignePanier::getId)

.collect(Collectors.*toList*());

// Appel du comportement

repository.delete(panierDB);

// Vérification que l'entité a été supprimée

**final** Panier panierDB2 = entityManager.find(Panier.**class**, panierDB.getId());

*assertNull*(panierDB2);

// Vérifier que tous les LignePanier sont supprimés par cascade

*assertThat*(listeIdLignePanier).isNotNull();

*assertThat*(listeIdLignePanier).isNotEmpty();

listeIdLignePanier.forEach(id -> {

*assertThat*(id).isGreaterThan(0);

LignePanier lpDB = entityManager.find(LignePanier.**class**, id);

*assertNull*(lpDB);

});

}

@Test

**public** **void** test\_orphanRemoval() {

**final** Panier panierDB = panierEnDB();

**final** List<Integer> listeIdLignePanier = panierDB.getLignesPanier()

.stream()

.map(LignePanier::getId)

.collect(Collectors.*toList*());

// Détacher les LignePanier

panierDB.getLignesPanier().clear();

// Appel du comportement

repository.delete(panierDB);

// Vérification que l'entité a été supprimée

**final** Panier panierDB2 = entityManager.find(Panier.**class**, panierDB.getId());

*assertNull*(panierDB2);

// Vérifier que tous les LignePanier sont supprimées par orphanRemoval

*assertThat*(listeIdLignePanier).isNotNull();

*assertThat*(listeIdLignePanier).isNotEmpty();

listeIdLignePanier.forEach(id -> {

*assertThat*(id).isGreaterThan(0);

LignePanier lpDB = entityManager.find(LignePanier.**class**, id);

*assertNull*(lpDB);

});

}

**private** Panier panierEnDB() {

**final** Panier panier = **new** Panier();

**final** LignePanier lp = LignePanier

.*builder*()

.quantite(3)

.prix(3 \* 11.45f)

.build();

panier.getLignesPanier().add(lp);

panier.setPrixTotal(lp.getPrix());

entityManager.persist(panier);

entityManager.flush();

*assertThat*(panier.getId()).isGreaterThan(0);

*assertThat*(panier.getId()).isGreaterThan(0);

**return** panier;

}

}

* Les tables créées pour l’association Panier-LignePanier sont :

create table cav\_line (line\_id integer generated by default as identity, price float(7), quantity integer, shopping\_cart\_id integer, primary key (line\_id))

create table cav\_shopping\_cart (paid boolean, shopping\_cart\_id integer generated by default as identity, total\_price float(7), order\_number varchar(200), primary key (shopping\_cart\_id))

* + Voici la clef étrangère (jointure) ajoutée :

alter table if exists cav\_line add constraint FK4n1m9axjgk9sraxy7w5a0noa1 foreign key (shopping\_cart\_id) references cav\_shopping\_cart

* Les traces obtenues sont :
  + Le panier créé par le test test\_save\_nouvelleLigne\_nouveauPanier

Panier(id=1, numCommande=null, prixTotal=95.8, paye=false, lignesPanier=[LignePanier(id=1, quantite=4, prix=95.8)])

* + Le panier mis à jour par le test test\_save\_nouvelleLigne\_Panier
    - Initialement :

Panier(id=2, numCommande=null, prixTotal=34.35, paye=false, lignesPanier=[LignePanier(id=2, quantite=3, prix=34.35)])

* + - Après l’ajout de la nouvelle ligne :

Panier(id=2, numCommande=null, prixTotal=273.85, paye=false, lignesPanier=[LignePanier(id=2, quantite=3, prix=34.35), LignePanier(id=3, quantite=10, prix=239.5)])