Cave à vin (partie 04)

# TP04 du module 02 – Spring Data

|  |
| --- |
| Ce TP sera un TP fil rouge du cours   * Création de 2 associations ManyToOne |

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

Solution

# Créer les entités

## Voici le diagramme des entités à réaliser :

Une image contenant texte, diagramme, ligne, Parallèle

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.vin
* Utiliser les annotations de JPA
  + Appliquer les annotations pour l’association ManyToONe
  + Penser à préciser l’annotation @JoinColumn
  + Réfléchir aux contraintes de cascade
* Ces 3 entités n’ont pas de contraintes pour le toString
* Cependant, il faut exclure les 2 associations du equals de Bouteille

## Créer l’entité Couleur

* C’est une entité assez minimaliste
* Elle contient uniquement
  + Une clef primaire autogénérée par la base
  + Un attribut représentant le nom de la couleur
    - Il doit être unique et une taille maximale de 250
* Il est possible d’utiliser l’annotation @Data pour cette entité

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

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

**import** lombok.\*;

@NoArgsConstructor

@AllArgsConstructor

@Data

@Builder

@Entity

@Table(name = "CAV\_COLOR")

**public** **class** Couleur {

@Id

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

@Column(name = "COLOR\_ID")

**private** Integer id;

@Column(name = "NAME", length = 250, unique = **true**, nullable = **false**)

**private** String nom;

}

## Créer l’entité Region

* Cette entité est sur le même modèle que l’entité Couleur
* Les seules différences sont sur le nom des colonnes et de la table en base

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

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

**import** lombok.\*;

@NoArgsConstructor

@AllArgsConstructor

@Data

@Builder

@Entity

@Table(name = "CAV\_REGION")

**public** **class** Region {

@Id

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

@Column(name = "REGION\_ID")

**private** Integer id;

@Column(name = "NAME", length = 250, unique = **true**, nullable = **false**)

**private** String nom;

}

## Créer l’entité Bouteille

* Elle contient les attributs :
  + nom à représentant le nom de la bouteille
    - Il doit être unique et une taille maximale de 250
  + petillant à booleén
  + millesime
    - Une taille maximale de 100
  + quantite
  + prix à avec une précision de 2
* Pas de contrainte sur le toString

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

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

**import** lombok.\*;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@EqualsAndHashCode

@ToString

@Builder

@Entity

@Table(name = "CAV\_BOTTLE")

**public** **class** Bouteille {

@Id

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

@Column(name = "BOTTLE\_ID")

**private** Integer id;

@Column(name = "NAME", length = 250, unique = **true**, nullable = **false**)

**private** String nom;

@Column(name = "SPARKLING")

**private** **boolean** petillant;

@Column(name = "VINTAGE", length = 100)

**private** String millesime;

@Column(name = "QUANTITY")

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

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

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

@EqualsAndHashCode.Exclude

@ManyToOne

@JoinColumn(name = "REGION\_ID")

**private** Region region;

@EqualsAndHashCode.Exclude

@ManyToOne

@JoinColumn(name = "COLOR\_ID")

**private** Couleur couleur;

}

* Et les 2 associations ManyToOne vers Couleur et Region
  + Les couleurs et les régions doivent exister indépendamment des bouteilles, donc pas de notion de cascade sur les associations
  + Nous ne voulons pas que les 2 associations soient dans le equals à utilisation @EqualsAndHashCode.Exclude

# Créer les Repository

* Aucune contrainte particulière
  + Ils sont tous les 3 sur le modèle de celui-ci :

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

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

**import** fr.eni.cave.bo.vin.Couleur;

**public** **interface** CouleurRepository **extends** JpaRepository<Couleur, Integer>{

}

# Créer des tests unitaires

* Une classe de tests unitaires appelée TestManyToOne vous est fournie
  + Copier la classe dans le package fr.eni.cave.association des tests Java
* Créer 3 tests unitaires pour ces 2 associations :
  + Créer des prix avec au moins 2 décimales pour bien montrer le comportement
  + test\_save à permet de sauver une bouteille avec une couleur et une région
  + test\_save\_bouteilles\_regions\_couleurs()à elle permet d’enregistre plusieurs bouteilles avec des couleurs similaires et des régions similaires
  + test\_delete() à elle permet de vérifier que si on supprime une bouteille, cela ne supprime pas la couleur et la région associées

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

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

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

**import** java.util.\*;

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

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

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

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

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

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

@Slf4j

@DataJpaTest

**public** **class** TestManyToOne {

@Autowired

**private** TestEntityManager entityManager;

@Autowired

BouteilleRepository bouteilleRepository;

@Autowired

CouleurRepository couleurRepository;

@Autowired

RegionRepository regionRepository;

Couleur rouge;

Couleur blanc;

Couleur rose;

Region grandEst;

Region paysDeLaLoire;

Region nouvelleAquitaine;

…

@Test

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

**final** Bouteille bouteille = Bouteille

.*builder*()

.nom("Blanc du DOMAINE ENI Ecole")

.millesime("2022")

.prix(23.95f)

.quantite(1298)

.build();

// Association ManyToOne

bouteille.setRegion(paysDeLaLoire);

bouteille.setCouleur(blanc);

// Appel du comportement

**final** Bouteille bouteilleDB = bouteilleRepository.save(bouteille);

// Vérification de l'identifiant

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

// Vérification des associations

*assertThat*(bouteilleDB.getCouleur()).isNotNull();

*assertThat*(bouteilleDB.getCouleur()).isEqualTo(blanc);

*assertThat*(bouteilleDB.getRegion()).isNotNull();

*assertThat*(bouteilleDB.getRegion()).isEqualTo(paysDeLaLoire);

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

}

@Test

**public** **void** test\_save\_bouteilles\_regions\_couleurs() {

**final** List<Bouteille> bouteilles = jeuDeDonnees();

// sauver le jeu de données en base

bouteilles.forEach(b ->{

bouteilleRepository.save(b);

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

// Vérification des associations

*assertThat*(b.getCouleur()).isNotNull();

*assertThat*(b.getRegion()).isNotNull();

});

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

}

@Test

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

**final** Bouteille bouteille = Bouteille

.*builder*()

.nom("Blanc du DOMAINE ENI Ecole")

.millesime("2022")

.prix(23.95f)

.quantite(1298)

.build();

// Association ManyToOne

bouteille.setRegion(paysDeLaLoire);

bouteille.setCouleur(blanc);

// Appel du comportement

**final** Bouteille bouteilleDB =entityManager.persist(bouteille);

entityManager.flush();

// Vérification de l'identifiant

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

*assertThat*(bouteilleDB.getCouleur()).isNotNull();

*assertThat*(bouteilleDB.getCouleur()).isEqualTo(blanc);

*assertThat*(bouteilleDB.getRegion()).isNotNull();

*assertThat*(bouteilleDB.getRegion()).isEqualTo(paysDeLaLoire);

// Appel du comportement

bouteilleRepository.delete(bouteilleDB);

// Vérification que l'entité a été supprimée

**final** Bouteille bouteilleDB2 = entityManager.find(Bouteille.**class**, bouteilleDB.getId());

*assertNull*(bouteilleDB2);

// Vérifier que les éléments associés sont toujours présents - PAS de cascade

**final** List<Region> regions = regionRepository.findAll();

*assertThat*(regions).isNotNull();

*assertThat*(regions).isNotEmpty();

*assertThat*(regions.size()).isEqualTo(3);

**final** List<Couleur> couleurs = couleurRepository.findAll();

*assertThat*(couleurs).isNotNull();

*assertThat*(couleurs).isNotEmpty();

*assertThat*(couleurs.size()).isEqualTo(3);

}

**…**

}

* Les tables créées pour les associations Bouteille-Couleur et Bouteille-Region sont :

create table cav\_bottle (bottle\_id integer generated by default as identity, color\_id integer, price float(7), quantity integer, region\_id integer, sparkling boolean, vintage varchar(100), name varchar(250) not null unique, primary key (bottle\_id))

create table cav\_color (color\_id integer generated by default as identity, name varchar(250) not null unique, primary key (color\_id))

create table cav\_region (region\_id integer generated by default as identity, name varchar(250) not null unique, primary key (region\_id))

* + Voici les 2 clefs étrangères (jointure) ajoutées :

alter table if exists cav\_bottle add constraint FK950w1rc8dkmptb45fk1nos6ra foreign key (color\_id) references cav\_color

alter table if exists cav\_bottle add constraint FKj2rhorgadcok6128cimp3slci foreign key (region\_id) references cav\_region

* Voici les traces obtenues  sur l’exécution de test\_save\_bouteilles\_regions\_couleurs() :

[

Bouteille(

id=3,

nom=Blanc du DOMAINE ENI Ecole,

petillant=false,

millesime=2022,

quantite=1298,

prix=23.95,

region=Region(id=8, nom=Pays de la Loire),

couleur=Couleur(id=8, nom=Blanc)),

Bouteille(

id=4,

nom=Rouge du DOMAINE ENI Ecole,

petillant=false,

millesime=2018,

quantite=987,

prix=11.45,

region=Region(id=8, nom=Pays de la Loire),

couleur=Couleur(id=7, nom=Rouge)),

Bouteille(

id=5,

nom=Blanc du DOMAINE ENI Service,

petillant=true,

millesime=2022,

quantite=111,

prix=34.0,

region=Region(id=7, nom=Grand Est),

couleur=Couleur(id=8, nom=Blanc)),

Bouteille(

id=6,

nom=Rouge du DOMAINE ENI Service,

petillant=false,

millesime=2012,

quantite=344,

prix=8.15,

region=Region(id=8, nom=Pays de la Loire),

couleur=Couleur(id=7, nom=Rouge)),

Bouteille(

id=7,

nom=Rosé du DOMAINE ENI,

petillant=false,

millesime=2020,

quantite=1987,

prix=33.0,

region=Region(id=9, nom=Nouvelle Aquitaine),

couleur=Couleur(id=9, nom=Rosé))

]