Association OneToMany unidirectionnelle

# Démonstration 7 du module 2

|  |
| --- |
| Les objectifs de cette démonstration :   * Déclaration d’une association OneToMany * Manipulation des paramètres (cascade, orphanRemoval et fetch) * Utilisation de l’annotation @JoinColumn |

Déroulement

# Contexte

* Continuer l’application précédente
* Dans cette itération, nous allons ajouter la relation (1:N) entre Promo et EtudiantEni :
  + Voici le diagrammes des entités

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

Description générée automatiquement

# Création de l’entité Promo :

* Utilisation des annotations de JPA pour en faire une entité
  + Il faut que les EtudiantEni soient enregistrés, mis à jour et supprimés en même temps que la Promo associée : cascade = CascadeType.ALL
  + Si la liste des EtudiantEni est setter à null ; il faut que tous les EtudiantEni orphelins soient supprimés : orphanRemove = true
  + Dans l’idéal, nous ne voulons pas charger par défaut la liste des EtudiantEni tant qu’on n’en a pas besoin : fetch = FetchType.LAZY
* @JoinColumn pour préciser le nom en base
* Nous ne voulons pas que la liste soit considérée dans la méthode equals : @EqualsAndHashCode.Exclude
* Nous voulons qu’elle soit initialisée à une liste vide par défaut, pour ne pas tomber sur un NullPointerException avec l’appel d’un simple add()
  + Avec Lombok, il faut utiliser l’annotation @Builder.Default et mettre une valeur par défaut : = new ArrayList<>() ;

**package** fr.eni.demo.bo.stagiaire;

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

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

**import** lombok.\*;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Entity

@Table(name = "STUDENT\_CLASS")

**public** **class** Promo {

@Id

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

**private** **int** id;

@Column(name = "NAME", nullable = **false**, length = 12)

**private** String nom;

@EqualsAndHashCode.Exclude

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

@JoinColumn(name="STUDENT\_ID")

**private** @Builder.Default List<EtudiantEni> etudiants = **new** ArrayList<>();

}

Aucune modification a apporté à l’entité EtudiantEni

# Création de PromoRepository :

* Créer l’interface PromoRepository qui hérite de JpaRepository<Promo, Integer>

# Utilisation des tests unitaires

* Vérifions le bon fonctionnement de nos annotations et de nos paramètres de OneToMany
* Créer une classe TestOneToManyUni
  + Se faire injecter TestEntityManager et PromoRepository
* Créer 3 tests :
  + Persister une Promo et ses EtudiantEni
  + Supprimer une Promo et par cascade tous ses EtudiantEni
  + Détacher les EtudiantEni d’une Promo, supprimer celle-ci et valider que les EtudiantEni orphelins sont supprimés

**package** fr.eni.demo.association;

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

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

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

**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.demo.bo.stagiaire.\*;

**import** fr.eni.demo.dal.PromoRepository;

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

@Slf4j

@DataJpaTest

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

@Autowired

**private** TestEntityManager entityManager;

@Autowired

PromoRepository repository;

@Test

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

**final** Promo promo = Promo

.*builder*()

.nom("EDWM\_TEST")

.build();

// Association OneToMany

promo.setEtudiants(jeuDeDonnees());

// Appel du comportement

**final** Promo promoDB = repository.save(promo);

// Vérification de l'identifiant

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

// Vérification de la cascade de l'association

*assertThat*(promoDB.getEtudiants()).isNotNull();

*assertThat*(promoDB.getEtudiants()).isNotEmpty();

*assertThat*(promoDB.getEtudiants().size()).isEqualTo(3);

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

}

@Test

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

**final** Promo promo = Promo

.*builder*()

.nom("EDWM\_TEST")

.build();

// Association OneToMany

promo.setEtudiants(jeuDeDonnees());

// Contexte de la DB

**final** Promo promoDB = entityManager.persist(promo);

entityManager.flush();

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

*assertThat*(promoDB.getEtudiants()).isNotNull();

*assertThat*(promoDB.getEtudiants()).isNotEmpty();

List<EtudiantEni> etudiantsDB = promoDB.getEtudiants();

List<String> listeIdEtudiantEniDB = etudiantsDB

.stream()

.map(EtudiantEni::getImmatriculation)

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

// Appel du comportement

repository.delete(promoDB);

// Vérification que l'entité a été supprimée

**final** Promo promoDB2 = entityManager.find(Promo.**class**, promo.getId());

*assertNull*(promoDB2);

// Vérifier que tous les EtudiantEni sont supprimés par cascade

*assertThat*(listeIdEtudiantEniDB).isNotNull();

*assertThat*(listeIdEtudiantEniDB).isNotEmpty();

listeIdEtudiantEniDB.forEach(etudiantId -> {

*assertThat*(etudiantId).isNotNull();

EtudiantEni empDB = entityManager.find(EtudiantEni.**class**, etudiantId);

*assertNull*(empDB);

});

}

@Test

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

**final** Promo promo = Promo

.*builder*()

.nom("EDWM\_TEST")

.build();

// Association OneToMany

promo.setEtudiants(jeuDeDonnees());

// Contexte de la DB

**final** Promo promoDB = entityManager.persist(promo);

entityManager.flush();

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

*assertThat*(promoDB.getEtudiants()).isNotNull();

*assertThat*(promoDB.getEtudiants()).isNotEmpty();

List<EtudiantEni> etudiantsDB = promoDB.getEtudiants();

List<String> listeIdEtudiantEniDB = etudiantsDB

.stream()

.map(EtudiantEni::getImmatriculation)

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

// Détacher les EtudiantEni de leur Promo

promo.getEtudiants().clear();

// Appel du comportement

repository.delete(promo);

// Vérification que l'entité a été supprimée

**final** Promo promoDB2 = entityManager.find(Promo.**class**, promo.getId());

*assertNull*(promoDB2);

// Vérifier que tous les EtudiantEni sont supprimés par orphanRemoval

*assertThat*(listeIdEtudiantEniDB).isNotNull();

*assertThat*(listeIdEtudiantEniDB).isNotEmpty();

listeIdEtudiantEniDB.forEach(etudiantId -> {

*assertThat*(etudiantId).isNotNull();

EtudiantEni empDB = entityManager.find(EtudiantEni.**class**, etudiantId);

*assertNull*(empDB);

});

}

**private** List<EtudiantEni> jeuDeDonnees() {

**final** List<EtudiantEni> etudiants = **new** ArrayList<>();

String immatriculation = "ENI\_CAMPUS\_20231187";

**for** (**int** i = 1; i < 4; i++) {

**final** DonneesPerso donneesPerso = DonneesPerso

.*builder*()

.nom("Nom"+i)

.prenom("Prenom"+i)

.numDom("02XXXXXXXX")

.numPortable("07XXXXXXXX")

.emailPersonnel("Nom"+i+".Prenom"+i+"@perso.fr")

.build();

**final** EtudiantEni etudiant = EtudiantEni

.*builder*()

.immatriculation(immatriculation+i)

.email("pnom"+i+"@campus-eni.fr")

.build();

etudiant.setDonneesPerso(donneesPerso);

donneesPerso.setEtudiantEni(etudiant);

etudiants.add(etudiant);

}

**return** etudiants;

}

}

* Tous les tests doivent être verts