Clef composite et requête spécifique

# Démonstration 05 du module 5

|  |
| --- |
| L’objectif de cette démonstration est   * Voir la contrainte avec les associations, clefs composites et les requêtes spécifiques |

Contexte

* Poursuivre dans notre deuxième application, qui permettra de gérer les avis des stagiaires.

Déroulement

# Requête sur clef composite

* Une clef composite est un objet complexe
  + Il faudrait passer un JSON dans l’URL
  + Ce n’est pas possible
  + Il n’est donc pas envisageable de travailler sur des clefs de ce type
* Il faut donc repenser son projet :
  1. Soit ; mettre en place un @RestController et un @Service pour manipuler la méthode du Repository en passant les données de la clef composite en plusieurs paramètres dans l’URL
  2. Soit ; repenser les BO pour manipuler des clefs primaires simples
* **Nous allons faire le choix 2.**
  + Nous sommes en mode création d’un nouveau projet, il n’y a pas de base de données en production
  + Il est donc possible de modifier le schéma de la base et des BO

## Suppression de la clef composite

* D’après le métier de l’ENI Ecole ; nous pouvons valider que « reference » est un élément unique
  + Il suffit comme identifiant pour les Cours
  + Voici le diagramme de classe de Cours maintenant

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

Description générée automatiquement

* Modifier la classe Cours comme suit :

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

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.\*;

**import** lombok.\*;

@Data

@AllArgsConstructor

@NoArgsConstructor

@EqualsAndHashCode(of = { "reference", "filiere" })

@ToString

@Builder

@Document(collection = "computer\_course")

**public** **class** Cours {

@Id

**private** String reference;

@Field(name = "computer\_science\_course")

**private** String filiere;

@Field(name = "title")

**private** String titre;

@Field(name = "duration")

**private** **int** duree;

}

* Son identifiant est « reference »
* « filiere » devient un simple attribut
* Ajout des annotations Lombok
  + @EqualsAndHashCode sur « reference » et « filiere »
  + @ToString
  + Pour gérer la même comparaison que précédemment
* Déplacer la classe dans le package fr.eni.demo.bo
  + Supprimer le package fr.eni.demo.bo.clefcomposite
* Modifier CoursRepository, la clef primaire est de type String maintenant

**package** fr.eni.demo.dal;

**import** org.springframework.data.mongodb.repository.MongoRepository;

**import** org.springframework.data.rest.core.annotation.RepositoryRestResource;

**import** fr.eni.demo.bo.Cours;

@RepositoryRestResource(collectionResourceRel = "cours", path = "cours")

**public** **interface** CoursRepository **extends** MongoRepository<Cours, String> {

}

## Recherche par Cours

* Nous devons remplacer dans AvisRepository la méthode « findByCours » qui manipule un objet complexe par findByCoursReference

…

@RepositoryRestResource(collectionResourceRel = "avis", path = "avis")

**public** **interface** AvisRepository **extends** MongoRepository<Avis, String> {

List<Avis> findByNoteCours(@Param("noteCours") **int** noteCours);

List<Avis> findByNoteCoursGreaterThan(@Param("noteCours") **int** noteCours);

List<Avis> findByNoteCoursLessThan(@Param("noteCours") **int** noteCours);

// List<Avis> findByStagiaire(@Param("stagiaire") Stagiaire stagiaire);

List<Avis> findByStagiaireImmatriculation(@Param("immatriculation") String immatriculation);

// List<Avis> findByFormateur(@Param("f") Formateur f);

List<Avis> findByFormateurEmail(@Param("email") String email);

// List<Avis> findByCours(@Param("c") Cours c);

List<Avis> findByCoursReference(@Param("reference") String reference);

}

## Mise à jour des tests unitaires

* Pour respecter l’évolution du code ; il faut revoir les tests unitaires et la base de données de notre application
* Supprimer la base de données et la recrée. Pour repartir d’une base vierge (Demo\_ENI\_Ecole)
* Déplacer la classe TestCoursDocument dans le package fr.eni.demo.bo
  + Supprimer le package fr.eni.demo.bo.clefcomposite
  + Modifier le test pour que la clef soit reference

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

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

**import** java.util.Optional;

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

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

**import** org.springframework.boot.test.context.SpringBootTest;

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

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

@Slf4j

@SpringBootTest

**class** TestCoursDocument {

@Autowired

CoursRepository coursRepository;

@Test

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

String coursId = "M360";

Cours cours = Cours

.*builder*()

.reference(coursId)

.filiere("Développement")

.titre("Java Frameworks - API Web")

.duree(10)

.build();

coursRepository.save(cours);

//Vérification en base

Optional<Cours> opt = coursRepository.findById(coursId);

*assertThat*(opt).isNotNull();

*assertThat*(opt.isPresent()).isTrue();

Cours coursDB = opt.get();

*assertThat*(coursDB).isNotNull();

*assertThat*(coursDB.getTitre()).isNotNull();

*assertThat*(coursDB.getDuree()).isEqualTo(10);

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

}

}

* Modifier la classe de tests : TestAssociationAvisCours
  + Tester comme clef getReference au lieu de getId

…

@Test

**void** test01\_save\_avis\_formateur() {

// Récupération des Cours en base

**…**

// Association avec Cours

…

// Sauver

**…**

// Vérifier que l'identifiant n'est pas nul

*…*

// Vérifier que le Cours est complet

*assertThat*(avisDB.getCours().getReference()).isNotNull();

*assertThat*(avisDB.getCours()).isEqualTo(coursDB);

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

}

* Modifier les tests de la classe TestRequetes
  + Mettre à jour la méthode insertion\_Formateur\_Cours\_DB

…

**void** insertion\_Formateur\_Cours\_DB() {

// Création de Formateur

**final** List<Formateur> listeFormateurs = **new** ArrayList<>();

…

// Création de Cours

**final** List<Cours> listeCours = **new** ArrayList<>();

listeCours.add(Cours

.*builder*()

.reference("M030")

.filiere("Développement")

.titre("Web Client")

.duree(5)

.build());

listeCours.add(Cours

.*builder*()

.reference("M070")

.filiere("Développement")

.titre("POO")

.duree(10)

.build());

// Enregistrement en base

listeCours.forEach(cours -> coursRepository.save(cours));

}

* + Mettre à jour la méthode test06\_findByCours

…

@Test

**void** test06\_findByCours() {

**final** Cours cours = Cours

.*builder*()

.reference("M070")

.filiere("Développement")

.titre("POO")

.duree(10)

.build();

List<Avis> listeAvis = avisRepository.findByCoursReference(cours.getReference());

*assertThat*(listeAvis).isNotNull();

*assertThat*(listeAvis).isNotEmpty();

*assertThat*(listeAvis.size()).isGreaterThan(1);

***log***.info("Nb Avis sur le Cours (" + cours.toString() + ") : " + listeAvis.size());

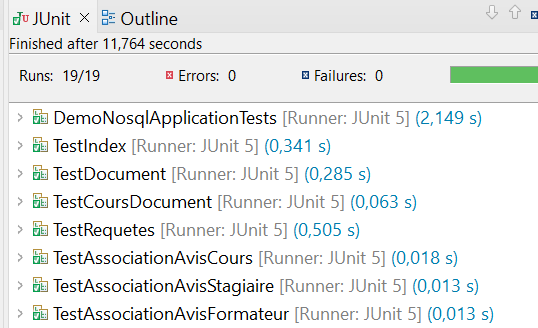
***log***.info(listeAvis.toString());

}

* Toute l’application doit compiler de nouveau

## Exécution de l’ensemble des tests

* Il est possible de lancer en une fois tous les tests unitaires
  + Faire un clic droit sur « src/test/java »
  + Run As 🡪 JUnit Test
* Tous les tests seront vers



## Exécution

* Relancer votre application
* Tester la requête de recherche par cours
* http://localhost:8080/avis/search/findByCoursReference?reference=M070