Requêtage

# Démonstration 10 du module 3

|  |
| --- |
| Les objectifs de cette démonstration :   * Utilisation des méthodes avec mots clefs de Spring Data MongoDB * Création de plusieurs requêtes spécifiques avec mots clefs sur les 3 stratégies d’associations |

Déroulement

# Contexte

Nous voulons une deuxième application, qui permettra de gérer les avis des stagiaires.

Elle permettra plus tard, de réaliser des statistiques sur les cours.

Notre application va se compléter au fil des démonstrations de ce module.

Elle est indépendante de l’application précédente.

Comme dans l’application précédente, les informations en base sont en anglais pour différencier les BO et les Document

A cette itération, nous allons ajouter des méthodes pour réaliser des requêtes spécifiques pour filtrer les Avis :

* Des requêtes pour mettre en avant les Avis :
  + Avec une note de Cours à 3
  + Avec une note de Cours > 3
  + Avec une note de Cours < 3
* Des requêtes pour mettre en avant les 3 stratégies d’associations :
  + Remonter les Avis d’un Stagiaire
  + Remonter les Avis associé à un Formateur
  + Remonter les Avis associé à un Cours

# Création des méthodes dans AvisRepository

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

**import** java.util.List;

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

**import** org.springframework.data.repository.query.Param;

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

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

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

@RepositoryRestResource(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> findByFormateur(@Param("f") Formateur f);

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

}

# Création d’une classe de tests

Il faut créer un test unitaire par méthodes spécifiques d’AvisRepository. Les tests correspondent :

* Des requêtes pour mettre en avant les Avis :
  + Avec une note de Cours à 3
  + Avec une note de Cours > 3
  + Avec une note de Cours < 3
* Des requêtes pour mettre en avant les 3 stratégies d’associations :
  + Remonter les Avis d’un Stagiaire
  + Remonter les Avis associé à un Formateur
  + Remonter les Avis associé à un Cours

Ajout d’une première méthode de test pour insérer plus de données en base.

* Si vous voulez par la suite relancer cette classe de tests
* Il faudra mettre en commentaire @Test sur la méthode test00\_insertion\_DB
  + Pour que les données ne soient pas recréées à chaque fois

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

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

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

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

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

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

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

**import** fr.eni.demo.bo.clefcomposite.\*;

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

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

@Slf4j

@SpringBootTest

@TestMethodOrder(MethodOrderer.MethodName.**class**)

**class** TestRequetes {

@Autowired

CoursRepository coursRepository;

@Autowired

FormateurRepository formateurRepository;

@Autowired

AvisRepository avisRepository;

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

// Création de Formateur

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

listeFormateurs.add(Formateur

.*builder*()

.email("pmontembault@campus-eni.fr")

.nom("MONTEMBAULT")

.prenom("Philippe")

.build());

listeFormateurs.add(Formateur

.*builder*()

.email("fdelachesnais@campus-eni.frr")

.nom("DELACHESNAIS")

.prenom("Frédéric")

.build());

// Enregistrement en base

listeFormateurs.forEach(formateur -> formateurRepository.save(formateur));

// Création de Cours

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

listeCours.add(Cours

.*builder*()

.id(CoursId

.*builder*()

.reference("M030")

.filiere("Développement")

.build())

.titre("Web Client")

.duree(5)

.build());

listeCours.add(Cours

.*builder*()

.id(CoursId

.*builder*()

.reference("M070")

.filiere("Développement")

.build())

.titre("POO")

.duree(10)

.build());

// Enregistrement en base

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

}

**void** insertion\_Avis\_DB() {

// Récupération depuis la base des Formateur et des Cours

**final** List<Formateur> listeFormateurs = formateurRepository.findAll();

*assertThat*(listeFormateurs).isNotNull();

*assertThat*(listeFormateurs).isNotEmpty();

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

**final** List<Cours> listeCours = coursRepository.findAll();

*assertThat*(listeCours).isNotNull();

*assertThat*(listeCours).isNotEmpty();

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

// Ajout d'Avis pour chaque Formateur avec chaque Cours

**for** (**int** i = 0; i < listeFormateurs.size(); i++) {

// Faire varier la note

**int** note = 2;

**final** Formateur f = listeFormateurs.get(i);

**for** (**int** j = 0; j < listeCours.size(); j++) {

**final** Cours c = listeCours.get(i);

**final** Avis avis = Avis

.*builder*()

.notePedagogie(note)

.commentairePedagogie("Commentaire sur la pédagogie (" + note + ")")

.noteCours(note)

.commentaireCours("Commentaire du cours (" + note + ")")

.cours(c)

.formateur(f)

.stagiaire(Stagiaire

.*builder*()

.immatriculation("ENI\_1253" + j)

.promotion("CDA1234" + j)

.build())

.build();

// Sauvegarde de Avis

avisRepository.save(avis);

// incrémenter la note

note++;

}

}

}

@Test

**void** test00\_insertion\_DB() {

insertion\_Formateur\_Cours\_DB();

insertion\_Avis\_DB();

}

@Test

**void** test01\_findByNoteCours\_3() {

List<Avis> listeAvis = avisRepository.findByNoteCours(3);

*assertThat*(listeAvis).isNotNull();

*assertThat*(listeAvis).isNotEmpty();

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

***log***.info("Le nombre d'Avis avec une note = 3 est de : " + listeAvis.size());

}

@Test

**void** test02\_findByNoteCoursGreaterThan\_3() {

List<Avis> listeAvis = avisRepository.findByNoteCoursGreaterThan(3);

*assertThat*(listeAvis).isNotNull();

*assertThat*(listeAvis).isNotEmpty();

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

***log***.info("Le nombre d'Avis avec une note > 3 est de : " + listeAvis.size());

}

@Test

**void** test03\_findByNoteCoursLessThan\_3() {

List<Avis> listeAvis = avisRepository.findByNoteCoursLessThan(3);

*assertThat*(listeAvis).isNotNull();

*assertThat*(listeAvis).isNotEmpty();

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

***log***.info("Le nombre d'Avis avec une note < 3 est de : " + listeAvis.size());

}

@Test

**void** test04\_findByStagiaire() {

**final** Stagiaire stagiaire = Stagiaire

.*builder*()

.immatriculation("ENI\_12531")

.promotion("CDA12341")

.build();

List<Avis> listeAvis = avisRepository.findByStagiaire(stagiaire);

*assertThat*(listeAvis).isNotNull();

*assertThat*(listeAvis).isNotEmpty();

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

***log***.info("Nb Avis du Stagiaire (" + stagiaire.toString() + ") : " + listeAvis.size());

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

}

@Test

**void** test05\_findByFormateur() {

**final** Formateur philippe = Formateur

.*builder*()

.email("pmontembault@campus-eni.fr")

.nom("MONTEMBAULT")

.prenom("Philippe")

.build();

List<Avis> listeAvis = avisRepository.findByFormateur(philippe);

*assertThat*(listeAvis).isNotNull();

*assertThat*(listeAvis).isNotEmpty();

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

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

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

}

@Test

**void** test06\_findByCours() {

**final** Cours cours = Cours

.*builder*()

.id(CoursId

.*builder*()

.reference("M070")

.filiere("Développement")

.build())

.titre("POO")

.duree(10)

.build();

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

*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());

}

}

# Exécution

* Tous les tests sont verts
  + La méthode d’insertion à rajouter :
  + 2 Formateur
  + 2 Cours
  + 9 Avis pour 3 Stagiaire (différents)

## Traces dans la console

Des requêtes pour mettre en avant les Avis :

* Avec une note de Cours à 3

find using query: { "course\_note" : 3} fields: Document{{}} for class: class fr.eni.demo.bo.Avis in collection: reviews

Le nombre d'Avis avec une note = 3 est de : 6

* + Il y a eu 3 nouveaux avis avec une note à 3
  + + les 3 qui existaient déjà
  + Remarquez que les paramètres passés à la requête sont des objets JSON
    - MongoDB travaille sur ce type d’objet uniquement
    - Et la recherche se fait par les champs d’un Document associé à une classe Java
    - Pour une Collection donnée
* Avec une note de Cours > 3

find using query: { "course\_note" : { "$gt" : 3}} fields: Document{{}} for class: class fr.eni.demo.bo.Avis in collection: reviews

Le nombre d'Avis avec une note > 3 est de : 4

* + Il y a eu 3 nouveaux avis avec une note > 3
  + + 1 qui existait déjà
  + Remarquez dans le paramètre en JSON il y a un sous objet JSON créé { "$gt" : 3}
    - Il correspond au nom du mot clef GreaterThan = $gt et la valeur 3
* Avec une note de Cours < 3

find using query: { "course\_note" : { "$lt" : 3}} fields: Document{{}} for class: class fr.eni.demo.bo.Avis in collection: reviews

Le nombre d'Avis avec une note < 3 est de : 3

* + Correspondent aux 3 nouveaux avis avec une note < 3
  + Remarquez dans le paramètre en JSON il y a un sous objet JSON créé { "$lt" : 3}
    - Il correspond au nom du mot clef LessThan = $lt et la valeur 3

Des requêtes pour mettre en avant les 3 stratégies d’associations :

* Les résultats corrsespondent uniquement aux 9 nouveaux Avis ajoutés
  + Car nous utilisons les nouveaux Formateur, Cours et Stagiaire ; insérés par cette classe de tests
* Remonter les Avis d’un Stagiaire

find using query: { "student" : { "student\_registration" : "ENI\_12531", "class\_id" : "CDA12341"}} fields: Document{{}} for class: class fr.eni.demo.bo.Avis in collection: reviews

Nb Avis du Stagiaire (Stagiaire(immatriculation=ENI\_12531, promotion=CDA12341)) : 3

* + Sur la même logique que dans les requêtes précédentes, le paramètre est un objet JSON
    - Il correspond à l’objet en paramètre ; dans cette requête un Formateur
* Remonter les Avis associé à un Formateur

find using query: { "trainer\_id" : "pmontembault@campus-eni.fr"} fields: Document{{}} for class: class fr.eni.demo.bo.Avis in collection: reviews

Nb Avis sur le Formateur (Formateur(email=pmontembault@campus-eni.fr, nom=MONTEMBAULT, prenom=Philippe)) : 3

* Remonter les Avis associé à un Cours

find using query: { "computer\_course\_id" : { "$java" : { "$ref" : "computer\_course", "$id" : "Document{{reference=M070, computer\_science\_course=Développement}}" } } } fields: Document{{}} for class: class fr.eni.demo.bo.Avis in collection: reviews

Nb Avis sur le Cours (Cours(id=CoursId(reference=M070, filiere=Développement), titre=POO, duree=10)) : 3