Création d’un projet avec Spring Data MongoDB et son premier Document

# Démonstration 4 du module 3

|  |
| --- |
| L’objectif de cette démonstration :   * Création d’un projet Spring Boot avec Spring Data MongoDB * Création d’un BO et en faire un Document * Tester le Document * Tester le schéma dynamique |

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

# Créer une application DemoNoSQL avec SpringBoot

* Utiliser Spring initializr et sélectionner :
  + Gradle (privilégier avec le langage Groovy)
  + Java en version minimum 17
  + La version courante de Spring Boot (Release)
  + Le packaging est en Jar
  + Métadonnées :

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

Description générée automatiquement

* + Ajouter les dépendances :
    - Lombok
    - Spring Boot DevTools
    - Spring Data MongoDB
* Importer le projet dans votre IDE, en tant que projet Gradle

# Configurer la base de données

* Pour configurer MongoDB, nous avons créé un fichier YAML
* Nous allons aussi utiliser un fichier YAML pour Spring Boot
* Renommer le fichier application.properties en application.yml

#Data Source

spring:

data:

mongodb:

username: sa

password: Passw0rd

host: localhost

port: 27017

database: Demo\_ENI\_Ecole

authentication-database: admin

* Nous allons ajouter des informations pour détailler les traces d’exécution de Spring Data MongoDB
  + Ajouter dans application.yml :

#Log

logging:

level:

org:

springframework:

data:

mongodb:

core:

MongoTemplate: DEBUG

# Déclaration d’un document :

A cette itération, nous allons uniquement ajouter l’objet métier Avis

* Il permet de donner une note sur le cours et la pédagogie entre 0 et 5
* Il permet de mettre un commentaire sur le cours et la pédagogie
* Voici le diagramme de la classe

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

Description générée automatiquement

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

**import** lombok.\*;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Builder

**public** **class** Avis {

@Id

**private** String id;

**private** **int** notePedagogie;

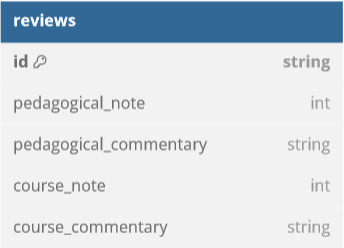
**private** String commentairePedagogie;

**private** **int** noteCours;

**private** String commentaireCours;

}

* Ajouter les annotations de Spring Data MongoDB pour que la classe métier devienne un document
  + Il faut qu’elle corresponde à la Collection :



* id est l’identifiant

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

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

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

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

**import** lombok.\*;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Builder

@Document(collection = "reviews")

**public** **class** Avis {

@Id

**private** String id;

@Field(name = "pedagogical\_note")

**private** **int** notePedagogie;

@Field(name = "pedagogical\_commentary")

**private** String commentairePedagogie;

@Field(name = "course\_note")

**private** **int** noteCours;

@Field(name = "course\_commentary")

**private** String commentaireCours;

}

* @Document(collection = "reviews") 🡪 permet de déclarer la classe comme un document et le paramètre collection permet de préciser le nom de la collection en base
* @Id 🡪 permet de préciser l’attribut qui représente l’identifiant. Par défaut dans MongoDB le type est String
* @Field 🡪 permet de préciser le nom du champ en base.

# Création de AvisRepository

* Créer l’interface AvisRepository qui hérite de MongoRepository<Avis,String>

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

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

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

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

}

# Création de tests unitaires

Dans le module précédent, nous avons utilisé une solution bouchonnée pour la base de données avec @DataJpaTest.

Dans ce module, nous allons privilégier de manipuler directement en base, pour voir le comportement de MongoDB

* Nous allons utiliser le contexte de Spring Boot directement
* @SpringBootTest
* Nous serons donc sur la vraie base de données : MongoDB

Les tests sont réalisés sur la base de données, les données sont donc conservées.

* Il faut donc penser à ordonner les tests :
  + @TestMethodOrder(MethodOrderer.MethodName.class)

Dans les tests unitaires,

* Nous allons vérifier les comportements de base :
  + Création d’un avis complet
  + Trouver tous les avis

Nous allons aussi vérifier le schéma dynamique

* Création d’un avis partiel
* Pour bien mettre en avant la notion de schéma dynamique, notre test de création d’un avis partiel sera réalisé avant le avis avec tous ses attributs

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

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

**import** java.util.List;

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

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

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

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

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

@Slf4j

@SpringBootTest

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

**class** TestDocument {

@Autowired

AvisRepository avisRepository;

@Test

**void** test01\_save\_avis\_sans\_commentaire\_cours() {

Avis avis = Avis

.*builder*()

.notePedagogie(4)

.commentairePedagogie("commentaire sur la pédagogie")

.noteCours(3)

.build();

Avis avisDB = avisRepository.save(avis);

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

*assertThat*(avisDB.getId()).isNotNull();

*assertThat*(avisDB.getId()).isNotBlank();

//Vérifier que l'attribut commentaireCours est nul

*assertThat*(avisDB.getCommentaireCours()).isNull();

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

}

@Test

**void** test02\_save\_avis\_tous\_attributs() {

Avis avis = Avis

.*builder*()

.notePedagogie(4)

.commentairePedagogie("commentaire sur la pédagogie")

.noteCours(3)

.commentaireCours("commentaire sur le cours")

.build();

Avis avisDB = avisRepository.save(avis);

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

*assertThat*(avisDB.getId()).isNotNull();

*assertThat*(avisDB.getId()).isNotBlank();

//Vérifier que tous les attributs sont remplis et non nuls

*assertThat*(avisDB.getNotePedagogie()).isNotNull();

*assertThat*(avisDB.getNotePedagogie()).isEqualTo(4);

*assertThat*(avisDB.getCommentairePedagogie()).isNotNull();

*assertThat*(avisDB.getNoteCours()).isNotNull();

*assertThat*(avisDB.getNoteCours()).isEqualTo(3);

*assertThat*(avisDB.getCommentaireCours()).isNotNull();

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

}

@Test

**void** test03\_findAll\_avis() {

List<Avis> listeAvis = avisRepository.findAll();

*assertThat*(listeAvis).isNotNull();

*assertThat*(listeAvis).isNotEmpty();

*assertThat*(listeAvis.size()).isEqualTo(2);

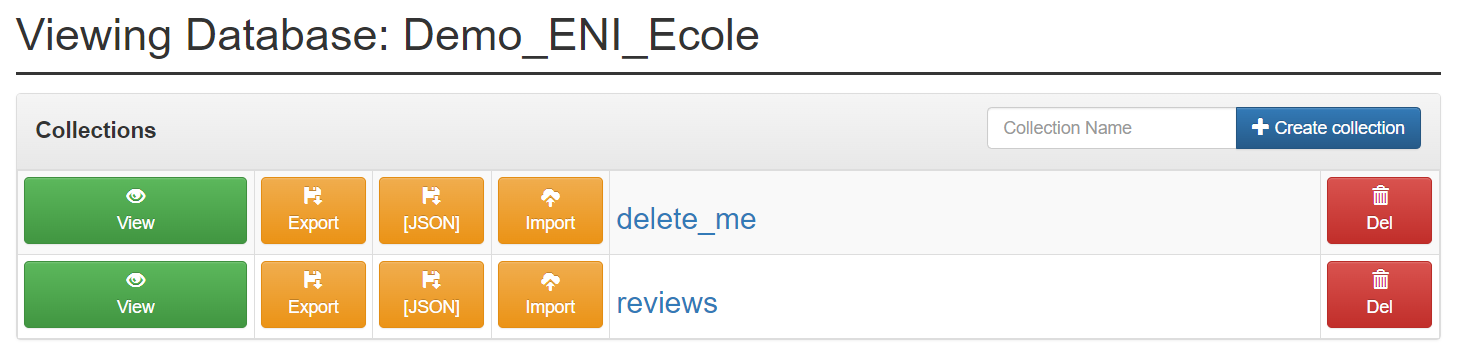
***log***.info(listeAvis.toString());

}

}

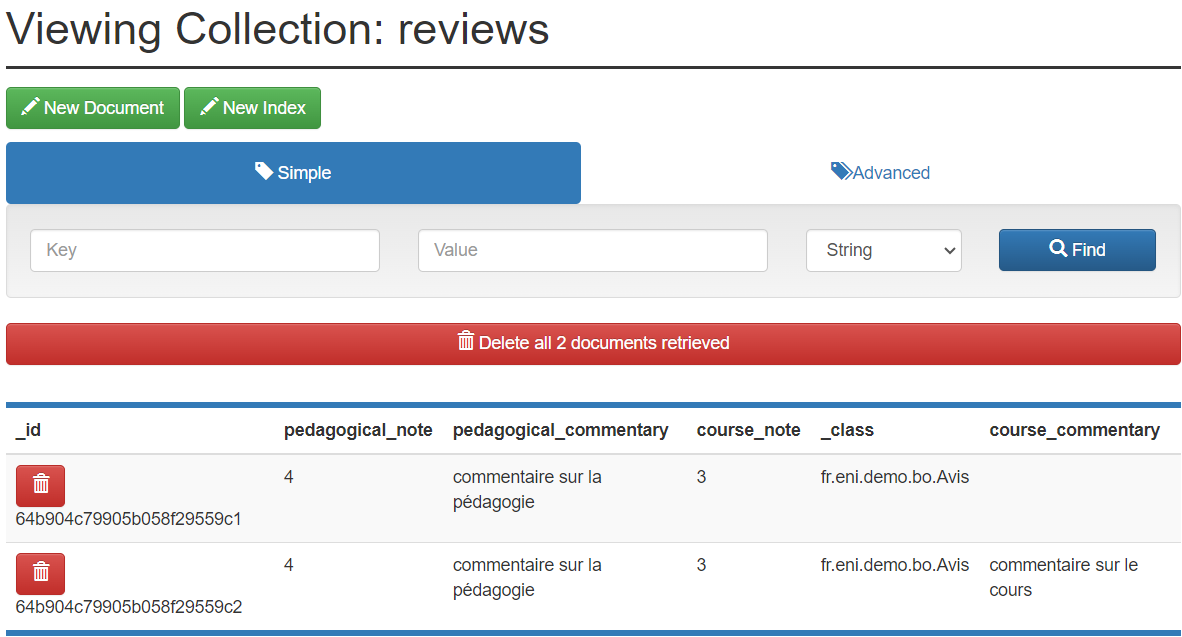
## Exécution :

* Tous les tests sont verts.
* En base, il y a maintenant 2 Collections : reviews et delete\_me



### reviews :

* Cliquer sur « View » pour voir le contenu de la Collection reviews :



1. Pour la création d’un avis qui n’a pas commentaireCours défini
   * Spring Data MongoDB a réalisé la requête :

Inserting Document containing fields: [pedagogical\_note, pedagogical\_commentary, course\_note, \_class] in collection: reviews

* + En une requête, il a inséré le Document et la collection nommée ‘reviews’
* Vérifier en base avec le détail du Document :



* + La structure est bien un JSON
  + Il y a l’identifiant qui est représenté par \_id pour signifier qu’il est auto-généré par la base
  + Il y a \_class qui indique à quelle classe Java ce Document est associé
  + Cependant, il n’y a pas dans le schéma de champs course\_commentary
* Remarquer que dans les traces Java, l’avis a bien l’attribut a nul :

Avis(id=64b904c79905b058f29559c1, notePedagogie=4, commentairePedagogie=commentaire sur la pédagogie, noteCours=3, commentaireCours=null)

* Remarque sur la vue de détail d’un Document dans Mongo-Express
  + Permet aussi de modifier ou supprimer le Document
  + D’ailleurs, elle est appelée Editing Document

1. Pour la création d’un avis qui a tous ses attributs

* Spring Data MongoDB a réalisé la requête :

Inserting Document containing fields: [pedagogical\_note, pedagogical\_commentary, course\_note, course\_commentary, \_class] in collection: reviews

* + Il y a bien course\_commentary
* Vérifier en base avec le détail du Document :



* Et remarquer que dans les traces Java, l’avis a bien un « commentaireCours » :

Avis(id=64b904c79905b058f29559c2, notePedagogie=4, commentairePedagogie=commentaire sur la pédagogie, noteCours=3, commentaireCours=commentaire sur le cours)

* Le schéma a été mis à jour automatiquement, sans que cela pose de problème pour l’ancien Document déjà enregistré
  + Nous avons donc la preuve, que dans une base de données NoSQL le schéma est dynamique

1. L’ensemble des avis en base  (Les indentations et les retours chariots sont là pour aider à la lecture) :

[

Avis(id=64b904c79905b058f29559c1, notePedagogie=4, commentairePedagogie=commentaire sur la pédagogie, noteCours=3, commentaireCours=null),

Avis(id=64b904c79905b058f29559c2, notePedagogie=4, commentairePedagogie=commentaire sur la pédagogie, noteCours=3, commentaireCours=commentaire sur le cours)

]

* Nous pouvons remarquer que Spring Data MongoDB n’a aucun souci à remonter l’ensemble des avis qu’ils aient ou non un « commentaireCours »
  + Les bases de données NoSQL sont plus flexibles que les bases de données relationnelles

Attention, nos données sont enregistrées en base, si vous relancer vos tests vous aurez plusieurs avis identiques en base

## En Base :

### delete\_me

* Vous pouvez maintenant supprimer la Collection delete\_me
  + Revenez sur la base Demo\_ENI\_Ecole
  + Cliquer sur le bouton « Del »
  + Puis saisissez le nom de la Collection à supprimer

Une image contenant texte, logiciel, Logiciel multimédia, capture d’écran

Description générée automatiquement

Attention, il faut toujours qu’il reste une Collection dans la base sinon, il retire la base de données