Association – Modèle de données dénormalisé

# Démonstration 7 du module 3

|  |
| --- |
| Les objectifs de cette démonstration :   * Création d’une référence embarquée 🡪 composition |

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 gérer les associations entre les classes de manière simple avec MongoDB.

* Nous allons utiliser le fait que la structure des données est flexible.
* Les attributs d’associations deviendront des sous objet JSON du Document
* Le Document principal contient les objets associés (même cycle de vie)

Pour respecter le RGPD, nous allons représenter les stagiaires que par leur identifiant et leur promotion.

# Suppression des données existantes

Nos tests sont réalisés directement en base. Pour manipuler les données facilement ; nous allons supprimer certaines collections.

* Suppression de la collection reviews

# Référence embarquée – Association

A cette étape, il faut que le stagiaire qui a donné un avis ; soit enregistré dans son avis

* Voici le schéma des classes que nous voulons gérer

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

Description générée automatiquement

## Création de la classe BO : Stagiaire

* Création de la classe Stagiaire
  + Elle doit respecter le Design Pattern POJO
  + Elle doit redéfinir equals et hashCode
* Avec l’annotation @Data de Lombok, nous avons bien la redéfinition des 2 méthodes
  + Equivalent à :
    - @Getter
    - @Setter
    - @RequiredArgsConstructor
    - @ToString
    - @EqualsAndHashCode
* Il manque uniquement le constructeur par défaut pour POJO : @NoArgsConstructor
  + Nous ajoutons comme d’habitude @Builder et @AllArgsConstructor
  + Pour gérer la création des instances de manière dynamique

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

**import** lombok.\*;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Builder

**public** **class** Stagiaire {

**private** String immatriculation;

**private** String promotion;

}

## Association

* Dans la classe Avis :
  + Ajout d’un attribut de type Stagiaire
  + Préciser le nom du champ en base : student

**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;

//Association

@Field("student")

**private** Stagiaire stagiaire;

}

Pour une association manuelle, avec une référence embarquée, il n’y a besoin de rien d’autre.

* Si vous voulez que le nom des champs soient spécifiques en base, vous pouvez ajouter @Field dans Stagiaire
  + Dans notre cas, nous voulons une base en Anglais
  + Donc, il faut renommer les 2 attributs de Avis

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

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

**import** lombok.\*;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Builder

**public** **class** Stagiaire {

@Field(name = "student\_registration")

**private** String immatriculation;

@Field(name = "class\_id")

**private** String promotion;

}

## Création d’un test pour valider l’association

* Essayons d’insérer un Avis et son Stagiaire

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

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

**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.dal.AvisRepository;

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

@Slf4j

@SpringBootTest

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

**class** TestAssociationAvisStagiaire {

@Autowired

AvisRepository avisRepository;

@Test

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

**final** Stagiaire stagiaire = Stagiaire

.*builder*()

.immatriculation("ENI\_CAMPUS\_202311987")

.promotion("EDMW0001")

.build();

**final** Avis avis = Avis

.*builder*()

.notePedagogie(4)

.commentairePedagogie("commentaire sur la pédagogie")

.noteCours(3)

.commentaireCours("commentaire sur le cours")

.build();

//Association

avis.setStagiaire(stagiaire);

**final** Avis avisDB = avisRepository.save(avis);

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

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

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

// Vérifier que la référence embarquée existe

**final** Stagiaire stagiaireDB = avisDB.getStagiaire();

*assertThat*(stagiaireDB).isNotNull();

*assertThat*(stagiaireDB).isEqualTo(stagiaire);

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

}

}

### Traces dans la console

* Spring Data MongoDB a bien inséré l’Avis et son Stagiaire dans la collection reviews :

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

* Et l’objet stocké en base permet bien de remonter une instance d’Avis et une instance de Stagiaire

Avis(id=64b92e305352be301430a856, notePedagogie=4, commentairePedagogie=commentaire sur la pédagogie, noteCours=3, commentaireCours=commentaire sur le cours, stagiaire=Stagiaire(immatriculation=ENI\_CAMPUS\_202311987, promotion=EDMW0001))

### En base :



* Il y a bien un Document Avis qui contient
  + Un objet JSON appelé student qui représente le Stagiaire
  + student est embarqué dans le Document,
  + Il est enregistré dans le même fichier

# Conclusion

#### + Avantage :

* Schéma simple
  + Ajout d’un attribut qui représente un objet JSON pour chaque association
* 1 seule requête pour récupérer le stagiaire et ses avis

#### - Inconvénients :

* Plus il y aura d’avis plus le Document Stagiaire sera lourd en base de données.
  + Il faut donc faire attention avec ces références embarquées que le nombre ne soit pas exponentiel.
* Il faut aussi penser que dès que le Document, est supprimé ; l’objet JSON associé l’est aussi.

MongoDB préconise cette solution pour que le nombre de requête et la rapidité d’accès à l’information soit conservée.