Gestion des avis (partie 04)

# TP04 du module 03 – Spring Data MongoDB

|  |
| --- |
| Ce TP sera un TP fil rouge du cours   * Création d’une association DBRef (stratégie 3) * Utilisation de tests unitaires pour valider |

|  |
| --- |
| **Durée estimée** |
| 30 minutes |

Contexte

En vous appuyant sur l’ensemble des concepts abordés dans le cours, mettez en place une application Spring Boot permettant la gestion des avis d’une cave à vin.

L’application de cave à vin sera une application d’API Web sécurisée à la fin des itérations.

L’application de gestion des avis sera aussi une application d’API Web :

* Elle doit permettre de gérer les avis des clients sur le vin commandé.
  + La date de l’avis
  + L’identifiant de la bouteille commandée, ainsi que de la Région et couleur
  + La quantité commandée par le client
  + Une note (entre 0 et 5), un commentaire et le pseudo du client (permettre de protéger les informations du client RGPD)
* Le but étant de faire des statistiques pour le propriétaire du Vignoble

Dans les itérations de ce module ; nous réaliserons uniquement du back de l’application. En particulier, la couche DAL et les BO pour le moment.

Cette application se fera avec Spring Boot.

Déclarer la structure des données en Anglais

* Voici le diagramme de classes pour cette itération

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

Description générée automatiquement

* Voici un schéma de l’objet JSON représentant un Document de la Collection reviews en base :

{

\_id: ObjectId('64be51d78b1b8f684cc35df9'),

note: 5,

commentary: 'Doux. A déguster frais',

date: ISODate('2023-07-24T10:26:31.325Z'),

client: {

login: 'bobeponge@email.fr',

quantity\_ordered: 11

},

bottle\_id: DBRef('undefined', '[object Object]'),

\_class: 'fr.eni.gestionavis.bo.Avis'

}

Solution

# Créer la classe Client et l’association avec Avis

* Ajouter dans Avis un attribut de type Bouteille
* Lui associé l’annotation @DBRef pour créer l’association entre les 2 Document
  + Utilisation de la stratégie 3 ; car l’identifiant de Bouteille est une clef composite
  + Il faut se rappeler que les Bouteille doivent exister avant d’être associées aux Avis
* Lui associé @Field pour préciser le nom du champ en base : bottle\_id

**package** fr.eni.gestionavis.bo;

**import** java.time.LocalDateTime;

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

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

**import** lombok.\*;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Builder

@Document(collection = "reviews")

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

@Id

**private** String id;

**private** **int** note;

@Field(name = "commentary")

**private** String commentaire;

**private** LocalDateTime date;

**private** Client client;

@DBRef

@Field("bottle\_id")

**private** Bouteille bouteille;

}

# Créer des tests unitaires

* En vous inspirant des tests des démonstrations.
* Créer une classe de tests unitaires qui s’appuie sur le contexte de Spring
  + @Slf4j
  + @SpringBootTest
  + @TestMethodOrder(MethodOrderer.MethodName.**class**)
* Créer un test :
  + Sauver un Avis avec son Client et sa Bouteille
  + Les 2 associations doivent être présentes et validées
  + Tracer l’Avis

**package** fr.eni.gestionavis.association;

**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.gestionavis.bo.\*;

**import** fr.eni.gestionavis.bo.vin.\*;

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

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

@Slf4j

@SpringBootTest

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

**class** TestAssociationAvisBouteille {

@Autowired

AvisRepository avisRepository;

@Autowired

BouteilleRepository bouteilleRepository;

@Test

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

// Récupération des Bouteille en base

**final** List<Bouteille> listeBouteilleDB = bouteilleRepository.findAll();

*assertThat*(listeBouteilleDB).isNotNull();

*assertThat*(listeBouteilleDB).isNotEmpty();

**final** Bouteille bouteilleDB = listeBouteilleDB.get(0);

**final** Client client = Client

.*builder*()

.pseudo("bobeponge@email.fr")

.quantiteCommandee(11)

.build();

**final** Avis avis = Avis

.*builder*()

.note(5)

.commentaire("Doux. A déguster frais")

.client(client)

.build();

// Association avec Bouteille

avis.setBouteille(bouteilleDB);

// Sauver

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

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

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

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

// Vérifier que le Client n'est pas nul

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

*assertThat*(avisDB.getClient()).isEqualTo(client);

// Vérifier que le Bouteille est complet

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

*assertThat*(avisDB.getBouteille()).isEqualTo(bouteilleDB);

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

}

}

* Le test est à vert
* Voici le type de trace obtenu :

find using query: {} fields: Document{{}} for class: class fr.eni.gestionavis.bo.vin.Bouteille in collection: bottles

Inserting Document containing fields: [note, commentary, date, client, bottle\_id, \_class] in collection: reviews

Avis(id=64be51d78b1b8f684cc35df9, note=5, commentaire=Doux. A déguster frais, date=2023-07-24T12:26:31.325589800,

client=Client(pseudo=bobeponge@email.fr, quantiteCommandee=11),

bouteille=Bouteille(id=BouteilleId(idBouteille=2298, idRegion=5, idCouleur=1),

nom=Vin Blanc ENI))