Index

# Démonstration 5 du module 3

|  |
| --- |
| Les objectifs de cette démonstration :   * Constater que l’identifiant est vu comme un index par défaut comme dans un SGBDR * Mettre en place des index uniques |

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.

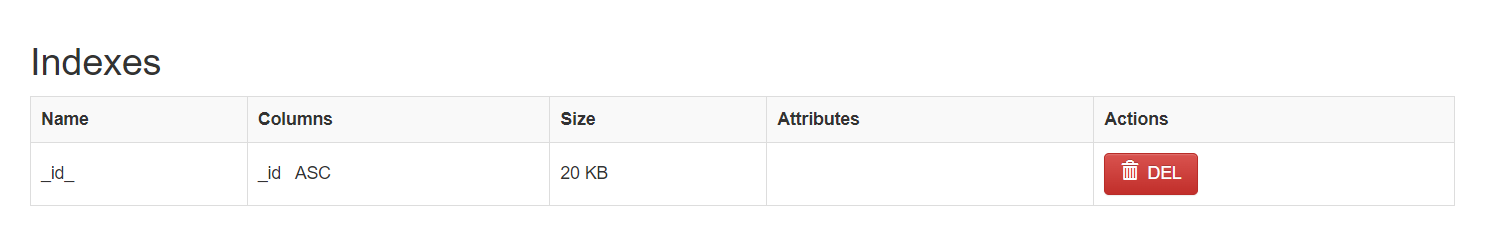
Notre application va se compléter au fil des démonstrations de ce module.

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

# Constater que l’identifiant est un index

* Dans MongoExpress, sélectionner la Collection reviews en cliquant sur « View »:
* Descendre en bas de la page
  + Il y a un tableau précisant les index de cette Collection
  + Constater \_id est bien référencé



# Activation de création des index dans application.yml

* Ajout de la configuration de création des index dans le fichier de configuration de Spring Boot :
  + auto-index-creation: true

#Data Source

spring:

data:

mongodb:

username: sa

password: Passw0rd

host: localhost

port: 27017

database: Demo\_ENI\_Ecole

authentication-database: admin

auto-index-creation: true

…

# Ajout d’index

Dans Avis, nous avons une note et un commentaire sur la pédagogie. Ces données seront liées à un Formateur. Dans cette itération, nous allons commencer à créer la classe BO : Formateur

* Voici le diagramme de classe pour Formateur :

Une image contenant texte, Police, blanc

Description générée automatiquement

MongoDB utilise une chaine de caractère comme identifiant par défaut

* Dans cette classe, email sera l’identifiant
* Nous voulons imposer que nom et prenom soient des index uniques
* Voici la collection désirée en base :

Une image contenant texte, reçu, algèbre

Description générée automatiquement

* Création de Formateur :

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

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

**import** org.springframework.data.mongodb.core.index.Indexed;

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

**import** lombok.\*;

@NoArgsConstructor

@AllArgsConstructor

@Data

@Builder

@Document(collection="trainers")

**public** **class** Formateur {

@Id

**private** String email;

@Field(name = "last\_name")

@Indexed(unique = **true**)

**private** String nom;

@Field(name = "first\_name")

@Indexed(unique = **true**)

**private** String prenom;

}

* @Document et précision du nom de la collection
* @Id sur reference pour qu’il soit l’identifiant du Document
* @Field pour préciser le nom des champs en base
* Ajout de l’annotation @Indexed sur les 2 attributs
  + Et ajout du paramètre unique = true

# Création de FormateurRepository

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

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

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

**public** **interface** FormateurRepository **extends** MongoRepository<Formateur, String> {

}

# Création de tests pour valider l’unicité des 2 index

* Création d’une nouvelle classe de tests : TestIndex

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

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

**import** **static** org.junit.jupiter.api.Assertions.*assertThrows*;

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

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

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

**import** org.springframework.dao.DuplicateKeyException;

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

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

@Slf4j

@SpringBootTest

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

**class** TestIndex {

@Autowired

FormateurRepository formateurRepository;

@Test

**void** test01\_formateur\_OK() {

**final** Formateur formateur = Formateur

.*builder*()

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

.nom("BAILLE")

.prenom("Anne-Lise")

.build();

**final** Formateur formateurDB = formateurRepository.save(formateur);

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

*assertThat*(formateurDB.getEmail()).isNotNull();

*assertThat*(formateurDB.getEmail()).isNotBlank();

*assertThat*(formateurDB.getEmail()).isEqualTo(formateur.getEmail());

// Vérifier que tous les attributs ne sont pas nuls

*assertThat*(formateurDB.getNom()).isNotNull();

*assertThat*(formateurDB.getPrenom()).isNotNull();

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

}

@Test

**void** test02\_unicite\_nom() {

**final** Formateur formateur = Formateur

.*builder*()

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

.nom("BAILLE")

.prenom("Marcel")

.build();

*assertThrows*(DuplicateKeyException.**class**, () -> formateurRepository.save(formateur));

}

@Test

**void** test03\_unicite\_prenom() {

**final** Formateur formateur = Formateur

.*builder*()

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

.nom("TOTO")

.prenom("Anne-Lise")

.build();

*assertThrows*(DuplicateKeyException.**class**, () -> formateurRepository.save(formateur));

}

@Test

**void** test04\_non\_nul\_KO() {

**final** Formateur formateur = Formateur

.*builder*()

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

.build();

**final** Formateur formateurDB = formateurRepository.save(formateur);

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

*assertThat*(formateurDB.getEmail()).isNotNull();

*assertThat*(formateurDB.getEmail()).isNotBlank();

// Vérifier que les attributs nom et prenom SONT NULS

*assertThat*(formateurDB.getNom()).isNull();

*assertThat*(formateurDB.getPrenom()).isNull();

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

//Supprimer ce Formateur incorrect

formateurRepository.delete(formateurDB);

}

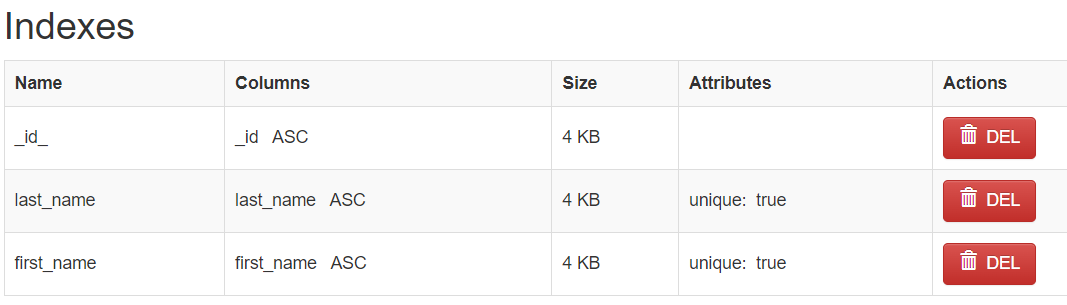
}

* Exécuter, ils seront tous verts
  + Attention, à la classe d’exception ; prenez bien :
    - org.springframework.dao.DuplicateKeyException
  + Le premier teste permet de créer un nouveau Formateur
  + Le second teste valide l’unicité du nom
  + Le troisième teste l’unicité du prenom
  + Le quatrième teste prouve qu’unique ne veut pas dire non nul pour MongoDB
    - Si nous ne l’avions pas supprimé, il y aurait un Formateur avec un nom et prenom à nuls en base
    - Ce n’est pas correcte par rapport au besoin métier

Pour éviter cela. Nous verrons par la suite qu’il est possible de faire des vérifications en BLL pour bloquer ce type de valeurs

## En base :

* Dans MongoExpress, sélectionner la Collection trainers en cliquant sur « View »:
* Descendre en bas de la page
  + Dans le tableau précisant les index de cette Collection
  + Constater qu’il y a maintenant 3 index :
    - \_id
    - last\_name
    - first\_name



* Remarquer que les deux index spécifiés ; sont indiqués
  + En unique
  + Leur sens de classement est ASC pour ascendant. C’est le mode par défaut
  + Si nous voulions en changer il faudrait ajouter le paramètre :

direction = IndexDirection.***DESCENDING***