Clefs primaires composites (méthode 2)

# Démonstration 4 du module 2

|  |
| --- |
| L’objectif de cette démonstration :   * Déclaration d’une clef primaire composite |

Déroulement

# Contexte

* Continuer l’application précédente
* Dans cette itération, nous allons ajouter la notion de stagiaire :
  + Voici la table en base de données désirées

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

Description générée automatiquement

* + Un stagiaire a
    - Une clef primaire composée de son email ENI et son numéro d’étudiant ENI
    - Un nom et un prénom obligatoires
    - Un numéro de téléphone de domicile et/ou de portable
    - Un email personnel obligatoire

# Modification de la classe gérant la clé primaire :

* Dupliquer le package fr.eni.demo.bo.pk en fr.en.demo.bo.pk2
* Renommer les 2 classes en EtudiantPK2 et Etudiant2
  + Pour permettre de conserver les 2 versions
* Elle déclare les 2 attributs de la clef primaire
  + Elle a la charge de préciser les colonnes et contraintes en base avec l’annotation @Column
* Et ajout de l’annotation @Embeddable
* Elle doit implémenter Serializable

**package** fr.eni.demo.bo.pk2;

**import** java.io.Serializable;

**import** jakarta.persistence.\*;

**import** lombok.\*;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Embeddable

**public** **class** EtudiantPK2 **implements** Serializable{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Column(nullable = **false**, unique = **true**, length = 255)

**private** String email;

@Column(name = "STUDENT\_REGISTRATION", nullable = **false**, unique = **true**, length = 100)

**private** String immatriculation;

}

# Modification de la classe d’entité (BO) :

* Renommer le nom de la table en STUDENT2
* Supprimer les 2 attributs représentant la clef composite
* Supprimer l’annotation @IdClass
* Ajout d’un attribut du type de la classe de la clef composite
  + Et l’annotation @EmbeddedId

**package** fr.eni.demo.bo.pk2;

**import** jakarta.persistence.\*;

**import** lombok.\*;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

/\*\*

\*

\* **@author** Eni Ecole

\*

\* **@category** Voici une entité avec une clef composite correspondant à l'email et

\* l'immatriculation

\*/

@Entity

@Table(name = "STUDENT2")

**public** **class** Etudiant2 {

//Définir l'attribut de la clef composite

@EmbeddedId

**private** EtudiantPK2 pk;

@Column(name = "LAST\_NAME", nullable = **false**, length = 90)

**private** String nom;

@Column(name = "FIRST\_NAME", nullable = **false**, length = 150)

**private** String prenom;

@Column(name = "HOME\_PHONE\_NUMBER", length = 12)

**private** String numDom;

@Column(name = "CELL\_NUMBER", length = 12)

**private** String numPortable;

@Column(name="PERSONAL\_EMAIL", nullable = **false**, unique = **true**, length = 255)

**private** String emailPersonnel;

}

# Création de Etudiant2Repository

* Créer l’interface Etudiant2Repository qui hérite de JpaRepository<Etudiant2, EtudiantPK2>
* La complexité est de bien faire attention d’utiliser la classe de la clef composite comme classe pour l’identifiant

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

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** fr.eni.demo.bo.pk2.Etudiant2;

**import** fr.eni.demo.bo.pk2.EtudiantPK2;

**public** **interface** Etudiant2Repository **extends** JpaRepository<Etudiant2, EtudiantPK2>{

}

# Création d’un test unitaire pour valider la clef composite

* Le test est très similaire au précédent
* Il met en avant la contrainte que les attributs de la clef composite ne sont plus dans la classe entité principale

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

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

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

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

**import** org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

**import** org.springframework.boot.test.autoconfigure.orm.jpa.TestEntityManager;

**import** fr.eni.demo.bo.pk2.Etudiant2;

**import** fr.eni.demo.bo.pk2.EtudiantPK2;

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

@Slf4j

@DataJpaTest

**public** **class** TestEtudiant2Repository {

@Autowired

**private** TestEntityManager entityManager;

@Autowired

Etudiant2Repository repository;

@Test

**public** **void** test\_save() {

**final** EtudiantPK2 pk = EtudiantPK2

.*builder*()

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

.immatriculation("ENI\_CAMPUS\_202311872")

.build();

**final** Etudiant2 entiteASauver = Etudiant2

.*builder*()

.pk(pk)

.nom("SOPRANO")

.prenom("Camille")

.numDom("02XXXXXXXX")

.emailPersonnel("camille.soprano@perso.fr")

.build();

// Appel du comportement

**final** Etudiant2 entite = repository.save(entiteASauver);

*assertThat*(entite).isNotNull();

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

**final** Etudiant2 entiteDB = entityManager.find(Etudiant2.**class**, pk);

*assertThat*(entiteDB).isNotNull();

*assertThat*(entite).isEqualTo(entiteDB);

}

}

## Exécution :

* Spring Data JPA a créé le script SQL suivant pour la génération de la table :

create table student2 (

cell\_number varchar(12),

home\_phone\_number varchar(12),

last\_name varchar(90) not null,

student\_registration varchar(100) not null unique,

first\_name varchar(150) not null,

email varchar(255) not null unique,

personal\_email varchar(255) not null unique,

primary key (student\_registration, email))

* + Il est identique au précédent sauf le nom de la table
* Au niveau des traces :

Etudiant2(pk=EtudiantPK2(email=csoprano@campus-eni.fr, immatriculation=ENI\_CAMPUS\_202311872), nom=SOPRANO, prenom=Camille, numDom=02XXXXXXXX, numPortable=null, emailPersonnel=camille.soprano@perso.fr)

* + C’est là que nous constatons toujours la complexité de la clef composite externe
  + Les attributs sont sous la classe EtudiantPK2

Cette deuxième solution est souvent présente. Car c’est la première implémentée.

Mais, il vaut mieux l’éviter car la structure déportée des attributs n’est pas souhaitable