Héritage – SINGLE\_TABLE

# Démonstration 11 du module 2

|  |
| --- |
| Les objectifs de cette démonstration sont :   * Déclaration d’une hiérarchie d’héritage avec SINGLE\_TABLE * Mise à plat de l’héritage avec Lombok |

Déroulement

# Contexte

* Continuer l’application précédente
* Dans cette itération, notre application va mettre en place la notion d’héritage :
  + Un formateur hérite d’un employé
  + Nous ajoutons une nouvelle catégorie d’employé les chargés de relations Ecole-Entreprise
* Nous allons utiliser SINGLE\_TABLE pour gérer cette hiérarchie d’employés
* Voici le diagrammes des entités de l’application actuelle :

Une image contenant texte, diagramme, ligne, Parallèle

Description générée automatiquement

* Nous devons gérer l’héritage entre Formateur et Employe et entre ChargeRelationsEcoleEntreprises et Employe

# Modification de l’entité Employe :

* Ajout de l’annotation @Inheritance
  + Avec la stratégie SINGLE\_TABLE
* Ajout de l’annotation @DiscriminatorColumn pour définir le nom de la colonne discriminante
* Ajout de l’annotation @DiscriminatorValue pour définir la valeur par défaut quand il s’agit d’une instance d’Employe

…

@Entity

@Table(name = "EMPLOYEE")

@Inheritance(strategy = InheritanceType.***SINGLE\_TABLE***)

@DiscriminatorColumn(name = "DISCR")

@DiscriminatorValue(value = "E")

**public** **class** Employe {

…

}

# Modification de l’entité Formateur :

* Il faut retirer tous les attributs qui appartiennent à l’entité Employe :
  + id
  + nom
  + prenom
  + email
  + immatriculation
  + numDom
  + numPortable
* Il restera :
  + L’attribut filiere
  + Et l’association avec les cours dispensés

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

**import** java.util.List ;

**import** jakarta.persistence.\* ;

**import** lombok.\* ;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@EqualsAndHashCode(of = { "immatriculation" })

@ToString

@Builder

/\*\*

\*

\* **@author** Eni Ecole

\*/

@Entity

@Table(name = "TRAINER")

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

@Column(name = "COMPUTER\_SCIENCE\_COURSE", length = 150)

**private** String filiere;

//Association ManyToMany

@ManyToMany(fetch = FetchType.***LAZY***)

@JoinTable(name = "COMPUTER\_COURSES\_PROVIDED",

joinColumns = { @JoinColumn(name = "TRAINER\_ID") },

inverseJoinColumns = {@JoinColumn(name = "COMPUTER\_COURSE\_ID") })

@ToString.Exclude

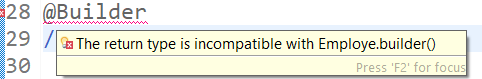
**private** List<Cours> coursDispenses;

}

* Il faut ajouter l’héritage vers l’entité Employe
  + Lors de l’ajout de « extends Employe »

**public** **class** Formateur **extends** Employe {

* + L’IDE émet une erreur sur l’annotation @Builder de Lombok



## Héritage et Lombok

Dès que nous gérons une hiérarchie d’héritage avec Lombok, il faut réfléchir à toutes les méthodes qui y sont liées :

* toString
* equals
* builder

Dans le cas des méthode toString et equals :

* Il faut soit vouloir les redéfinir localement en intégrant l’appel du comportement de la super classe
  + Exemple : @ToString(callSuper = true)
* Soit laisser le comportement définit sur la super classe
* Il est toujours possible d’exclure des attributs

L’annotation @Builder autogénère une méthode builder()

* Pour Lombok, il ne peut pas exister dans une hiérarchie d’héritage de méthode builder dans les filles, sans appeler celle de la mère
* Lombok, permet d’utiliser l’annotation @SuperBuilder pour gérer l’héritage de cette méthode
  + Cette annotation doit être placée sur la classe mère et les classes filles à la place de l’annotation @Builder

Remarque : cette annotation est dans un package d’évolution du framework : Lombok.experimental

Il faudra faire attention, qu’elle devienne le standard. Ou la remplacer sur une évolution du framework.

C’est le type de contrainte des frameworks.

### Dans l’entité Employe :

* Remplacer @Builder par @SuperBuilder

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

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

**import** lombok\*;

**import** lombok.experimental.SuperBuilder;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@EqualsAndHashCode(of = { "immatriculation" })

@ToString

@SuperBuilder

/\*\*

\*

\* **@author** Eni Ecole

\*

\*/

@Entity

@Table(name = "EMPLOYEE")

@Inheritance(strategy = InheritanceType.***SINGLE\_TABLE***)

@DiscriminatorColumn(name = "DISCR")

@DiscriminatorValue(value = "E")

**public** **class** Employe {

### Dans l’entité Formateur :

* Remplacer @Builder par @SuperBuilder
* Pour le toString
  + Nous voulons les informations de la super classe et l’attribut filiere
  + Ajouter l’appel de la super classe à l’annotation @ToString

@ToString(callSuper = **true**)

* Il reste à ajouter l’annotation pour préciser la valeur de la colonne discriminante

@DiscriminatorValue(value = "F")

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

**import** java.util.\*;

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

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

**import** lombok.\*;

**import** lombok.experimental.SuperBuilder;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString(callSuper = **true**)

@SuperBuilder

/\*\*

\*

\* **@author** Eni Ecole

\*/

@Entity

@Table(name = "TRAINER")

@DiscriminatorValue(value = "F")

**public** **class** Formateur **extends** Employe {

@Column(name = "COMPUTER\_SCIENCE\_COURSE", length = 150)

@ToString.Include

**private** String filiere;

// Association ManyToMany

@ManyToMany(fetch = FetchType.***LAZY***)

@JoinTable(name = "COMPUTER\_COURSES\_PROVIDED",

joinColumns = {@JoinColumn(name = "TRAINER\_ID") },

inverseJoinColumns = { @JoinColumn(name = "COMPUTER\_COURSE\_ID") })

@ToString.Exclude

@Builder.Default

**private** List<Cours> coursDispenses = **new** ArrayList<>();

}

## Héritage, Lombok et tests unitaires

Nous avons modifié l’entité Formateur ; il faut relancer la classe de tests associées TestManyToManyUni

* Relancer les tests, ils sont de nouveau vert

# Création de la classe ChargeRelationsEcoleEntreprises :

* Création d’un sous package fr.eni.demo.bo.accompagement
* Elle doit hériter de la classe Employe
* Utilisation des annotations de JPA pour en faire une entité
  + Positionner l’annotation @DiscriminatorValue(value = "C") pour préciser la valeur de la colonne discriminante
* Elle a un attribut numeroBureau
* Ajout de l’appel de la méthode toString de la super classe @ToString(callSuper = true)
* Il faut placer l’annotation @SuperBuilder à la place de @Builder
* Voici le code de la classe :

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

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

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

**import** lombok.\*;

**import** lombok.experimental.SuperBuilder;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString(callSuper = **true**)

@SuperBuilder

/\*\*

\*

\* **@author** Eni Ecole

\*/

@Entity

@Table(name = "SCHOOL\_BUSINESS\_OFFICER")

@DiscriminatorValue(value = "C")

**public** **class** ChargeRelationsEcoleEntreprises **extends** Employe {

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

**private** String numeroBureau;

}

# Création de ChargerReeRepository :

* Créer l’interface ChargerReeRepository qui hérite de
  + JpaRepository<ChargeRelationsEcoleEntreprises, Integer>

# Utilisation des tests unitaires

* Vérifions le bon fonctionnement de notre héritage
* Créer une classe TestHeritage
  + Se faire injecter TestEntityManager, EmployeRepository, FormateurRepository et ChargeReeRepository
* Créer une méthode de pré-chargement de 3 employés :
  + 1 Employe
  + 1 Formateur
  + 1 ChargeRelationsEcoleEntreprises
* Créer 3 tests :
  + Remonter tous les employés (= 3)
  + Remonter tous les formateurs (= 1)
  + Remonter tous les chargés de relations Ecole-Entreprises (= 1)

**package** fr.eni.demo.heritage;

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

**import** java.util.\*;

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

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

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

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

**import** fr.eni.demo.bo.accompagement.ChargeRelationsEcoleEntreprises;

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

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

**import** fr.eni.demo.dal.accompagement.ChargeReeRepository;

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

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

@Slf4j

@DataJpaTest

**public** **class** TestHeritage {

@Autowired

**private** TestEntityManager entityManager;

@Autowired

EmployeRepository employeRepository;

@Autowired

FormateurRepository formateurRepository;

@Autowired

ChargeReeRepository chargeReeRepository;

@BeforeEach

**public** **void** jeuDeDonnees() {

List<Employe> employes = **new** ArrayList<>();

employes.add(Employe

.*builder*()

.nom("NICOLAS")

.prenom("Cédric")

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

.immatriculation("ENI\_ECOLE\_10100")

.numDom("02XXXXXXXX")

.numPortable("06XXXXXXXX")

.build());

employes.add(Formateur

.*builder*()

.nom("BAILLE")

.prenom("Anne-Lise")

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

.immatriculation("ENI\_ECOLE\_12398")

.numDom("02XXXXXXXX")

.filiere("Développement")

.build());

employes.add(ChargeRelationsEcoleEntreprises

.*builder*()

.nom("PAGEOT")

.prenom("Emma")

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

.immatriculation("ENI\_ECOLE\_13398")

.numDom("02XXXXXXXX")

.numPortable("06XXXXXXXX")

.numeroBureau("04XXXXXXXX")

.build());

// Contexte de la DB

employes.forEach(e -> {

entityManager.persist(e);

// Vérification de l'identifiant

*assertThat*(e.getId()).isGreaterThan(0);

});

}

@Test

**public** **void** test\_findAll\_Employe() {

**final** List<Employe> employes = employeRepository.findAll();

// Vérification

*assertThat*(employes).isNotNull();

*assertThat*(employes).isNotEmpty();

*assertThat*(employes.size()).isEqualTo(3);

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

}

@Test

**public** **void** test\_findAll\_Formateur() {

**final** List<Formateur> formateurs = formateurRepository.findAll();

// Vérification

*assertThat*(formateurs).isNotNull();

*assertThat*(formateurs).isNotEmpty();

*assertThat*(formateurs.size()).isEqualTo(1);

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

}

@Test

**public** **void** test\_findAll\_ChargeRee() {

**final** List<ChargeRelationsEcoleEntreprises> chargesRee = chargeReeRepository.findAll();

// Vérification

*assertThat*(chargesRee).isNotNull();

*assertThat*(chargesRee).isNotEmpty();

*assertThat*(chargesRee.size()).isEqualTo(1);

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

}

}

* Tous les tests doivent être verts
* Au niveau base de données :
  + Création d’une table Employee

create table employee (

address\_id integer unique,

civility\_id varchar(3),

employee\_id integer generated by default as identity,

cell\_number varchar(12),

home\_phone\_number varchar(12),

office\_phone\_number varchar(12),

discr varchar(31) not null,

last\_name varchar(90) not null,

employee\_registration varchar(100) not null unique,

computer\_science\_course varchar(150),

first\_name varchar(150) not null,

email varchar(255) not null unique,

primary key (employee\_id)

)

* + Avec une colonne discriminante
  + Et 2 colonnes associées aux attributs supplémentaires de Formateur et ChargeRelationsEcoleEntreprises
* Les employés enregistrés sont :

[

Employe(

id=1,

civilite=null,

nom=NICOLAS,

prenom=Cédric,

email=cnicolas@campus-eni.fr,

immatriculation=ENI\_ECOLE\_10100,

numDom=02XXXXXXXX,

numPortable=06XXXXXXXX,

adresse=null),

Formateur(

super=Employe(

id=2,

civilite=null,

nom=BAILLE,

prenom=Anne-Lise,

email=abaille@campus-eni.fr,

immatriculation=ENI\_ECOLE\_12398,

numDom=02XXXXXXXX,

numPortable=null,

adresse=null),

filiere=Développement),

ChargeRelationsEcoleEntreprises(

super=Employe(

id=3,

civilite=null,

nom=PAGEOT,

prenom=Emma,

email=epageot@campus-eni.fr,

immatriculation=ENI\_ECOLE\_13398,

numDom=02XXXXXXXX,

numPortable=06XXXXXXXX,

adresse=null),

numeroBureau=04XXXXXXXX)

]