Cave à vin (partie 05)

# TP05 du module 02 – Spring Data

|  |
| --- |
| Ce TP sera un TP fil rouge du cours   * Création d’une hiérarchie d’héritage par jointure |

|  |
| --- |
| **Durée estimée** |
| 1 heure |

Solution

## Voici le diagramme des entités à réaliser :

Une image contenant texte, diagramme, ligne, Parallèle

Description générée automatiquement

Voici le diagramme des tables en base de données :

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

Description générée automatiquement

# Créer et modifier les entités

* Les packages des classes sont fr.eni.cave.bo.client et fr.eni.cave.bo
* Utiliser les annotations de JPA
  + Appliquer les annotations pour la hiérarchie d’héritage par jointure
* Les attributs : pseudo, nom, prenom et password sont remontés au niveau de la classe mère
* Les contraintes sur equals et toString sont aussi remontées sur la classe mère
  + Pour l’entité Client le toString de la classe mère suffit. Tous les attributs tracés sont remontés sur la classe mère
  + Penser à utiliser le paramètre callSuper pour l’annotation @ToString sur Proprio
* Penser à utiliser l’annotation @SuperBuilder sur les 3 entités à la place de @Builder

## Créer l’entité Utilisateur

**package** fr.eni.cave.bo;

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

**import** lombok.\*;

**import** lombok.experimental.SuperBuilder;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@EqualsAndHashCode(of = { "pseudo" })

@ToString(of = { "pseudo", "nom", "prenom" })

@SuperBuilder

@Entity

@Table(name = "CAV\_USER")

@Inheritance(strategy = InheritanceType.***JOINED***)

**public** **class** Utilisateur {

@Id

@Column(name = "LOGIN", nullable = **false**, length = 255)

**private** String pseudo;

@Column(name = "PASSWPORD", nullable = **false**, length = 68)

**private** String password;

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

**private** String nom;

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

**private** String prenom;

}

## Créer l’entité Proprio

* Elle hérite de l’Utilisateur
* Contient un attribut spécifique : siret avec une taille de 14
* Redéfinition de la méthode toString pour cumuler avec l’attribut spécifique
  + Utilisation de @ToString(callSuper = true)
  + Utilisation de @SuperBuilder pour avoir l’héritage de la méthode builder

**package** fr.eni.cave.bo;

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

**import** lombok.\*;

**import** lombok.experimental.SuperBuilder;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString(callSuper = **true**)

@SuperBuilder

@Entity

@Table(name = "CAV\_OWNER")

**public** **class** Proprio **extends** Utilisateur {

@Column(name = "CLIENT\_NUMBER", length = 14)

**private** String siret;

}

## Modifier l’entité Client

* Elle hérite de l’Utilisateur
* Tous ses attributs directs sont dans la classe mère
* Il reste uniquement l’association : Client-Adresse (OneToOne)
* Utilisation de @SuperBuilder pour avoir l’héritage de la méthode builder

**package** fr.eni.cave.bo.client;

**import** fr.eni.cave.bo.Utilisateur;

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

**import** lombok.\*;

**import** lombok.experimental.SuperBuilder;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@SuperBuilder

@Entity

@Table(name = "CAV\_CLIENT")

**public** **class** Client **extends** Utilisateur{

@OneToOne(cascade = CascadeType.***ALL***, orphanRemoval = **true**, fetch = FetchType.***EAGER***)

@JoinColumn(name = "ADDRESS\_ID")

**private** Adresse adresse;

}

# Créer les Repository des entités Utilisateur et Proprio

* Les 2 Repository sont sur le même modèle que celui de ClientRepository
  + Ils héritent de JpaRepository
  + Avec la classe d’entité en paramètre et String pour la clef primaire

# Créer des tests unitaires

* Une classe de tests unitaires appelée TestHeritage vous est fournie
  + Copier la classe dans le package fr.eni.cave.bo des tests Java
* Elle contient une méthode initDB annotée @BeforeEach
  + Elle injecte en base 3 utilisateurs :
    - 1 Utilisateur
    - 1 Propio
    - 1 Client
* En vous aidant des tests unitaires des démonstrations et des TP précédents
* Créer 3 tests unitaires :
  + Test\_findAll\_Utilisateur à doit retourner les 3 utilisateurs créés
  + Test\_findAll\_Proprio à doit retourner uniquement le propriétaire créé
  + Test\_findAll\_Client à doit retourner uniquement le client créé

**package** fr.eni.cave.bo;

**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.cave.bo.\*;

**import** fr.eni.cave.bo.client.Client;

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

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

@Slf4j

@DataJpaTest

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

@Autowired

**private** TestEntityManager entityManager;

@Autowired

UtilisateurRepository utilisateurRepository;

@Autowired

ProprioRepository proprioRepository;

@Autowired

ClientRepository clientRepository;

…

@Test

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

**final** List<Utilisateur> utilisateurs = utilisateurRepository.findAll();

// Vérification

*assertThat*(utilisateurs).isNotNull();

*assertThat*(utilisateurs).isNotEmpty();

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

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

}

@Test

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

**final** List<Proprio> proprios = proprioRepository.findAll();

// Vérification

*assertThat*(proprios).isNotNull();

*assertThat*(proprios).isNotEmpty();

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

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

}

@Test

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

**final** List<Client> clients = clientRepository.findAll();

// Vérification

*assertThat*(clients).isNotNull();

*assertThat*(clients).isNotEmpty();

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

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

}

}

* Voici le genre de traces obtenu (les retours chariots et décalages sont là pour simplifier la lecture, ils ne font pas parti des traces d’origine)
  + Les utilisateurs :

[ Utilisateur(pseudo=harrisonford@email.fr, nom=Ford, prenom=Harrison),

Proprio(super=Utilisateur(pseudo=georgelucas@email.fr, nom=Lucas, prenom=George), siret=12345678901234),

Utilisateur(pseudo=natalieportman@email.fr, nom=Portman, prenom=Natalie)]

* + Les propriétaires :

[Proprio(super=Utilisateur(pseudo=georgelucas@email.fr, nom=Lucas, prenom=George), siret=12345678901234)]

* + Les clients :

[Utilisateur(pseudo=natalieportman@email.fr, nom=Portman, prenom=Natalie)]

* Les tables créées pour l’héritage :

create table cav\_client (address\_id integer unique, login varchar(255) not null, primary key (login))

create table cav\_owner (client\_number varchar(14), login varchar(255) not null, primary key (login))

create table cav\_user (passwpord varchar(68) not null, last\_name varchar(90) not null, first\_name varchar(150) not null, login varchar(255) not null, primary key (login))

* les clefs de jointures

alter table if exists cav\_client add constraint FKjituqnpbm10ehaiprp4a3062m foreign key (login) references cav\_user

alter table if exists cav\_owner add constraint FKnuhnrsqlm1mtpxjdnw7kmpyuk foreign key (login) references cav\_user