Amina Dahmouni

Classe : II-BDCC2

[Compte Rendu]

Use case JPA, Hibernate Spring Data, One To Many, One To One

Patient, Medecin, Rendez-vous et Consultation

Architecture JEE et Middlewares

L'Ecole Normale Supérieure de l'Enseignement Technique de Mohammedia (ENSET)

Les entités :

- Classe Patient

```java
package ma.enset.hospital.Entities;
import ...
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Patient {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    @Temporal(TemporalType.DATE)
    private Date dateNaissance;
    private boolean malade;
    @OneToMany(mappedBy = "patient",fetch = FetchType.LAZY)
    private Collection<RendezVous> rendezVous;
}
```

- Classe Medecin

```java
package ma.enset.hospital.Entities;
import ...
@Entity @NoArgsConstructor @AllArgsConstructor
@Data
public class Medecin {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private String email;
    private String specialite;
    @OneToMany(mappedBy = "medecin",fetch = FetchType.LAZY)
    private Collection<RendezVous> rendezVous;
}
```

- Classe Consultation

```java
package ma.enset.hospital.Entities;
import ...
@Entity @NoArgsConstructor @AllArgsConstructor
@Data
public class Consultation {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date dateConsultation;
    private String rapport;
    @OneToOne
    private  RendezVous rendezVous;
}
```

- Classe RendezVous

```java
package ma.enset.hospital.Entities;
import ...
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class RendezVous {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private  Long id;
    private Date date;
    //Type ennuméré
    @Enumerated(EnumType.STRING)//Stocker le type au format string
    private StatusRND status;
    @ManyToOne
    private  Patient patient;
    @ManyToOne
    private  Medecin medecin;
    @OneToOne(mappedBy="rendezVous")
    private  Consultation consultation;
}
```

- Le type énuméré StatusRND pour la classe RendezVous

StatusRND.java

```java
package ma.enset.hospital.Entities;
public enum StatusRND {
    PENDING,
    CANCLED,
    DONE
}
```

HospitalApplication.java :

- Tester la validité du modèle

HospitalApplication.java

```java
import ...
@SpringBootApplication
public class HospitalApplication {
    public static void main(String[] args) {
        SpringApplication.run(HospitalApplication.class, args);
    }
    /* Dans Spring lorsqu'il y'a une méthode qui utilise l'annotation
       @Bean c.a.d cette méthode va etre exécuté au démarage et au meme
       temps cette méthode va retourner un objet et cette objet devient
       un composant spring elle est équivalente à l'annotation @Component.
     */
```

- Création des patients et des médecins

```
@Bean
CommandLineRunner start(PatientRepository patientRepository
, MedecinRepository medecinRepository, RendezVousRepository rendezVousRepository,
                        ConsultationRepository consultationRepository){
 return  args -> {
     //Une boocle pour créer 3 patients
     Stream.of("Mohamed","Imane","Halima").forEach(name->{
         Patient patient = new Patient();
         patient.setNom(name);
         patient.setDateNaissance(new Date());
         patient.setMalade(false);
         patientRepository.save(patient);
     });
    Stream.of("Bouchra","Hamid","Hounayda").forEach(name->{
         Medecin medecin = new Medecin();
         medecin.setNom(name);
         medecin.setEmail(name+"@gmail.com");
         medecin.setSpecialite(Math.random()>0.5?"Cardio":"Dentiste");
         medecinRepository.save(medecin);
     });
```

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM
1	2022-03-13	FALSE	Mohamed
2	2022-03-13	FALSE	Imane
3	2022-03-13	FALSE	Halima

(3 rows, 2 ms)

SELECT * FROM MEDECIN;

ID	EMAIL	NOM	SPECIALITE
1	Bouchra@gmail.com	Bouchra	Cardio
2	Hamid@gmail.com	Hamid	Dentiste
3	Hounayda@gmail.com	Hounayda	Cardio

(3 rows, 3 ms)

- Création d'un rendez-vous

```
   Patient patient1=patientRepository.findById(1L).orElse( other: null);
   Patient patient2=patientRepository.findByNom( name: "Halima");

   Medecin medecin=medecinRepository.findByNom( name: "Bouchra");

    RendezVous rendezVous=new RendezVous();
    rendezVous.setDate(new Date());
    rendezVous.setStatus(StatusRND.PENDING);
    rendezVous.setMedecin(medecin);
    rendezVous.setPatient(patient1);
    rendezVousRepository.save(rendezVous);
```

SELECT * FROM RENDEZ_VOUS;

ID	DATE	STATUS	MEDECIN_ID	PATIENT_ID
1	2022-03-13 10:42:32.536	PENDING	1	1

(1 row, 3 ms)

- Création d'une consultation

```
RendezVous rendezVous1=rendezVousRepository.findById(1L).orElse( other: null);

Consultation consultation=new Consultation();
consultation.setDateConsultation(new Date());
consultation.setRendezVous(rendezVous1);
consultation.setRapport("Rapport de la consultation ...");
consultationRepository.save(consultation);
```

SELECT * FROM CONSULTATION;

ID	DATE_CONSULTATION	RAPPORT	RENDEZ_VOUS_ID
1	2022-03-13 10:42:32.544	Rapport de la consultation ...	1

(1 row, 2 ms)

Avec la couche métier :

- Les méthodes déclarées dans l'interface IHospitalService

```
package ma.enset.hospital.Service;
import ...
public interface IHospitalService {
    Patient savePatient(Patient patient);
    Medecin saveMedecin(Medecin medecin);
    RendezVous saveRDV(RendezVous rendezVous);
    Consultation saveConsultation(Consultation consultation);
}
```

- La classe HospitalServiceImpl pour implémenter l'interface

```java
package ma.enset.hospital.Service;
import ...
@Service
@Transactional//Opérations transactionnels
public class HospitalServiceImpl implements IHospitalService {
    /* Pour l'injection des dépendances soit on fait @Autowired
        pour chaque variable(patientRepository,medecinRepository...)
        ou bien on utilise constructeur avec paramétres.
    */
    private PatientRepository patientRepository;
    private MedecinRepository medecinRepository;
    private RendezVousRepository rendezVousRepository;
    private ConsultationRepository consultationRepository;
    public HospitalServiceImpl(PatientRepository patientRepository,
                               MedecinRepository medecinRepository,
                               RendezVousRepository rendezVousRepository,
                               ConsultationRepository consultationRepository) {
        this.patientRepository = patientRepository;
        this.medecinRepository = medecinRepository;
        this.rendezVousRepository = rendezVousRepository;
        this.consultationRepository = consultationRepository;
    }
```

```java
    @Override
    public Patient savePatient(Patient patient) { return patientRepository.save(patient); }
    @Override
    public Medecin saveMedecin(Medecin medecin) { return medecinRepository.save(medecin); }
    @Override
    public RendezVous saveRDV(RendezVous rendezVous) {
        //Pour générer des chaines de caractères aléatoires uniques
        rendezVous.setId(UUID.randomUUID().toString());
        return rendezVousRepository.save(rendezVous);
    }
    @Override
    public Consultation saveConsultation(Consultation consultation) {
        return consultationRepository.save(consultation);
    }
```

Dans la classe RendezVous on change le type du Id à String afin de travailler avec des Id sous format de chaines de caractères (Aussi dans l'interface RendezVousRepository).

HospitalApplication.java :

```
CommandLineRunner start(IHospitalService hospitalService,PatientRepository patientRepository
        , MedecinRepository medecinRepository, RendezVousRepository rendezVousRepository){
    return  args -> {
        //Une boocle pour créer 3 patients
        Stream.of("Mohamed","Imane","Halima").forEach(name->{
            Patient patient = new Patient();
            patient.setNom(name);
            patient.setDateNaissance(new Date());
            patient.setMalade(false);
            hospitalService.savePatient(patient);
        });
        Stream.of("Bouchra","Hamid","Hounayda").forEach(name->{
            Medecin medecin = new Medecin();
            medecin.setNom(name);
            medecin.setEmail(name+"@gmail.com");
            medecin.setSpecialite(Math.random()>0.5?"Cardio":"Dentiste");
            hospitalService.saveMedecin(medecin);
        });
```

```
Patient patient1=patientRepository.findById(1L).orElse( other: null);
Patient patient2=patientRepository.findByNom( name: "Halima");

Medecin medecin=medecinRepository.findByNom( name: "Bouchra");

RendezVous rendezVous=new RendezVous();
rendezVous.setDate(new Date());
rendezVous.setStatus(StatusRND.PENDING);
rendezVous.setMedecin(medecin);
rendezVous.setPatient(patient1);
hospitalService.saveRDV(rendezVous);
//récuperer tt les RDV et prendre le première findAll().get(0)
//Dans le domaine du Big Data il y'a pas d'auto-increment
RendezVous rendezVous1=rendezVousRepository.findAll().get(0);

Consultation consultation=new Consultation();
consultation.setDateConsultation(new Date());
consultation.setRendezVous(rendezVous1);
consultation.setRapport("Rapport de la consultation ...");
hospitalService.saveConsultation(consultation);
```

SELECT * FROM RENDEZ_VOUS;

ID	DATE	STATUS	MEDECIN_ID	PATIENT_ID
63ea5b5b-26ac-442e-9631-23fa6bc5fcc5	2022-03-13 11:26:14.356	PENDING	1	1

(1 row, 6 ms)

La couche Web :

PatientRestController.java

```java
package ma.enset.hospital.Web;
import ...
@RestController
public class PatientRestController {
    @Autowired
    private PatientRepository patientRepository;
    //Méthode pour consulter la liste des patients
    @GetMapping ("/patients")
        public List<Patient> patientList() { return patientRepository.findAll(); }
        /* à cause de la relation bidiréctionnelle
          on obtient un affichage infini des données
         */
}
```

Pour éviter le probléme cousé par la relation bidiréctionnelle on utilise l'annotation @JsonProperty avec la valeur WRITE_ONLY pour prendre en considération l'attribut seulement dans l'ajout mais dans l'affichage c'est pas l'appel.

Cette annotation est ajoutée dans les classes RendezVous, Medecin et Consultation pour afficher la liste des patients et pour chaque patient on affiche la liste des rendez-vous et pour chaque rendez-vous les informations du médecin et de la consultation.

Afin d'obtenir la hiérarchie suivante :

```
▼ 0:
    id:                     1
    nom:                    "Mohamed"
    dateNaissance:          "2022-03-13"
    malade:                 false
  ▼ rendezVous:
    ▼ 0:
        id:                 "873b070a-0df3-4212-ae4b-44472bb52eb7"
        date:               "2022-03-13T10:44:55.780+00:00"
        status:             "PENDING"
      ▼ medecin:
            id:             1
            nom:            "Bouchra"
            email:          "Bouchra@gmail.com"
            specialite:     "Dentiste"
      ▼ consultation:
            id:             1
            dateConsultation: "2022-03-13T10:44:55.807+00:00"
            rapport:        "Rapport de la consultation ..."
▼ 1:
    id:                     2
    nom:                    "Imane"
    dateNaissance:          "2022-03-13"
    malade:                 false
    rendezVous:             []
▼ 2:
    id:                     3
    nom:                    "Halima"
    dateNaissance:          "2022-03-13"
    malade:                 false
    rendezVous:             []
```