Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

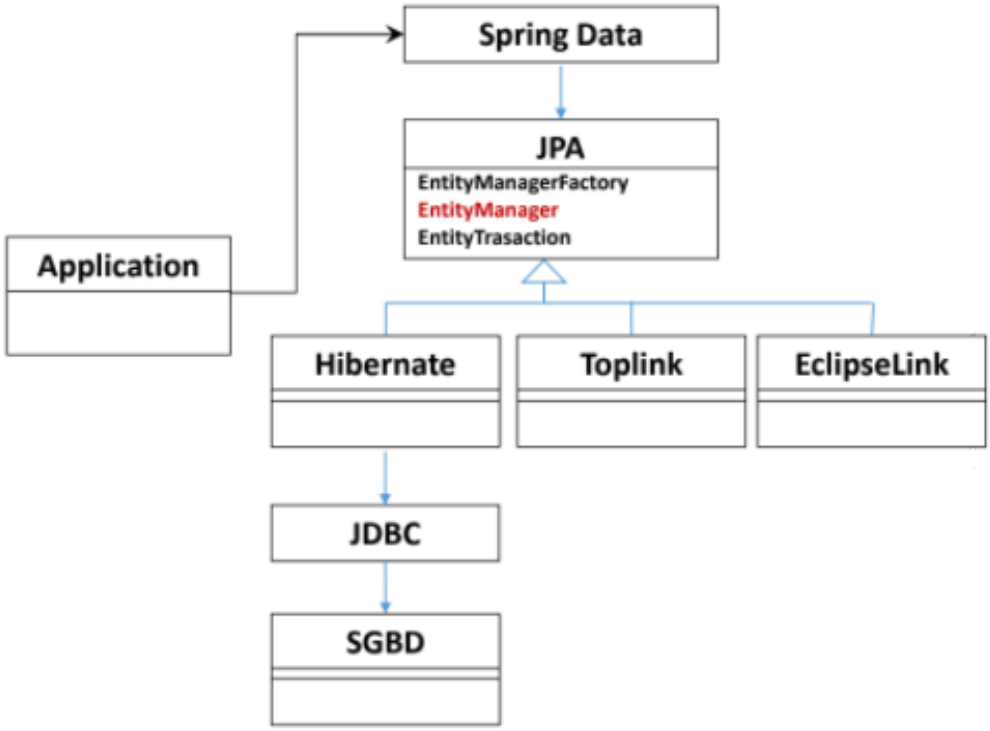
ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

CR DE L'ACTIVITE PRATIQUE N°2 : JPA, HIBERNATE ET SPRING DATA

Filière :« Ingénierie Informatique : Big Data et Cloud Computing » II-BDCC



Réalisé par :

Khadija BENJILALI

Encadré par :

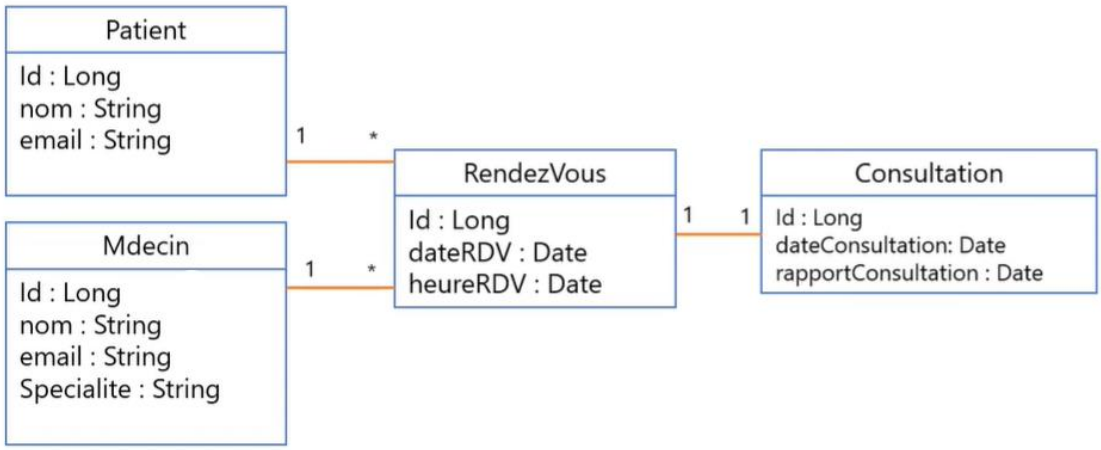
Mohamed YOUSSFI

Année Universitaire : 2021-2022

Mapping objet relationnel avec JPA, Hibernate et Spring Data

Exercice 1 : Association OneToMany, ManyToOne, OneToOne

Exemple : Cas de Patient, Medecin, Rendez-vous, Consultation
On souhaite gérer les rendez-vous des consultations des patients effectuées par des médecins.



- Chaque Rendez-vous concerne un patient et un médecin.
- Pour chaque rendez-vous on associe une seule consultation issue de rendez-vous.
- Un Patient peut prendre plusieurs rendez-vous

➢ Structure du projet

➢ Entities JPA

Patient.java

```java
package ma.enset.hospitalspringdata.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)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Collection<RendezVous> rendezVous;
}
```

Medecin.java

```java
package ma.enset.hospitalspringdata.entities;
import ...

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
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)
    //@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Collection<RendezVous> rendezVous;
}
```

RendezVous.java

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class RendezVous {
    @Id private String id;
    @Temporal(TemporalType.TIMESTAMP)
    private Date date;
    @Enumerated(EnumType.STRING)
    private StatusRDV status;
    @ManyToOne
    // @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Patient patient;
    @ManyToOne
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Medecin medecin;
    @OneToOne(mappedBy = "rendezVous")
    private Consultation consultation;
}
```

Consultation.java

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Consultation {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date dateConsultation;
    private String rapport;

    @OneToOne
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private RendezVous rendezVous;
}
```

➢ Interfaces DAO basées sur Spring Data

ConsultationRepository.java

```java
package ma.enset.hospitalspringdata.repositories;
import ...

public interface ConsultationRepository extends JpaRepository<Consultation, Long> {
}
```

MedecinRepository.java

```java
package ma.enset.hospitalspringdata.repositories;
import ...

public interface MedecinRepository extends JpaRepository<Medecin, Long> {
    List<Medecin> findByNom(String nom);
}
```

PatientRepository.java

```java
package ma.enset.hospitalspringdata.repositories;
import ...

public interface PatientRepository extends JpaRepository<Patient, Long> {
    List<Patient> findByNom(String nom);
}
```

RendezVousRepository.java

```java
package ma.enset.hospitalspringdata.repositories;
import ...

public interface RendezVousRepository extends JpaRepository<RendezVous, String> {
}
```

La couche service (métier)

Interface

IHospitalService.java

```java
package ma.enset.hospitalspringdata.service;
import ...

public interface IHospitalService {
    Patient savePatient(Patient patient);
    Medecin saveMedecin(Medecin medecin);
    RendezVous saveRDV(RendezVous rendezVous);
    Consultation saveConsultation(Consultation consultation);
}
```

Implémentation

HospitalServiceImpl.java

```java
@Service @Transactional
public class HospitalServiceImpl implements IHospitalService {
    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;
    }
    @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) {
        //générer les chaines alaitoires et unique
        rendezVous.setId(UUID.randomUUID().toString());
        return rendezVousRepository.save(rendezVous);
    }
    @Override public Consultation saveConsultation(Consultation consultation) {
        return consultationRepository.save(consultation);
    }
}
```

➢ Application

HospitalSpringDataApplication.java

```java
@Bean
CommandLineRunner start(IHospitalService hospitalService, PatientRepository patientRepository,
                        MedecinRepository medecinRepository, RendezVousRepository rendezVousRepository)
{
    return args -> {
        //liste des patients
        Stream.of("Ahlam","Jamil","Farah").forEach(name->{
            Patient patient = new Patient();
            patient.setNom(name);
            patient.setDateNaissance(new Date());
            patient.setMalade(Math.random()>0.5?true:false);
            hospitalService.savePatient(patient);
        });
        //liste des medecins
        Stream.of("Ali", "Khadija", "Asmaa").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 patient = patientRepository.findById(1L).orElse( other: null);
        Medecin medecin = medecinRepository.findById(1L).orElse( other: null);
        RendezVous rendezVous = new RendezVous();
        rendezVous.setDate(new Date());
        rendezVous.setStatus(StatusRDV.DONE);
        rendezVous.setPatient(patient);
        rendezVous.setMedecin(medecin);
        hospitalService.saveRDV(rendezVous);

        RendezVous rendezVous1 = rendezVousRepository.findAll().get(0);
        Consultation consultation = new Consultation();
        consultation.setDateConsultation(rendezVous1.getDate());
        consultation.setRendezVous(rendezVous1);
        consultation.setRapport("Rapport de la consultation ....");
        hospitalService.saveConsultation(consultation);
    };
  }
}
```

➢ Résultat

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM
1	2022-04-04	FALSE	Ahlam
2	2022-04-04	TRUE	Jamil
3	2022-04-04	FALSE	Farah

SELECT * FROM MEDECIN;

ID	EMAIL	NOM	SPECIALITE
1	Ali@gmail.com	Ali	Cardio
2	Khadija@gmail.com	Khadija	Cardio
3	Asmaa@gmail.com	Asmaa	Cardio

SELECT * FROM RENDEZ_VOUS;

ID	DATE	STATUS	MEDECIN_ID	PATIENT_ID
677f0efd-4cac-47a8-ab9b-32295f98f761	2022-04-04 00:15:28.976	DONE	1	1

SELECT * FROM CONSULTATION;

ID	DATE_CONSULTATION	RAPPORT	RENDEZ_VOUS_ID
1	2022-04-04 00:15:28.976	Rapport de la consultation	677f0efd-4cac-47a8-ab9b-32295f98f761

➢ La couche web

• Le contrôleur Spring MVC

PatientRestController.java

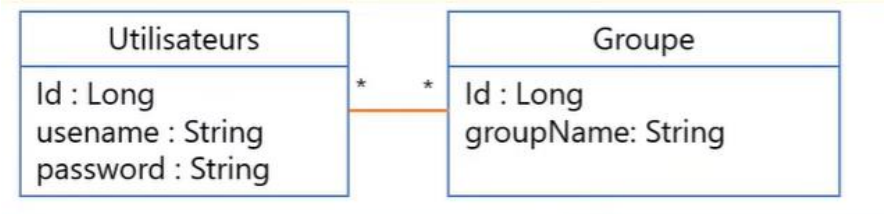
```java
@RestController
public class PatientRestController {

    @Autowired
    private PatientRepository patientRepository;
    @Autowired
    private MedecinRepository medecinRepository;

    @GetMapping("/patients")
    public List<Patient> patientList() { return patientRepository.findAll(); }

    @GetMapping("/medecins")
    public List<Medecin> medecinList() { return medecinRepository.findAll(); }
}
```

localhost:8086/medecins


```
[
  {
    "id": 1,
    "nom": "Ali",
    "email": "Ali@gmail.com",
    "specialite": "Cardio",
    "rendezVous": [
      {
        "id": "677f0efd-4cac-47a8-ab9b-32295f98f761",
        "date": "2022-04-03T22:15:28.976+00:00",
        "status": "DONE",
        "patient": {
          "id": 1,
          "nom": "Ahlam",
          "dateNaissance": "2022-04-04",
          "malade": false
        },
        "consultation": {
          "id": 1,
          "dateConsultation": "2022-04-03T22:15:28.976+00:00",
          "rapport": "Rapport de la consultation ...."
        }
      }
    ]
  },
  {
    "id": 2,
    "nom": "Khadija",
    "email": "Khadija@gmail.com",
    "specialite": "Cardio",
    "rendezVous": []
  },
  {
    "id": 3,
    "nom": "Asmaa",
    "email": "Asmaa@gmail.com",
    "specialite": "Cardio",
    "rendezVous": []
  }
]
```


Exercice 2 : Association ManyToMany

On suppose que l'on souhaite de créer une application qui permet de gérer des Utilisateurs appartenant à des groupes. Chaque Groupe peut contenir plusieurs utilisateurs.



➢ Entities JPA

User.java

```java
@Entity
@Table(name="USERS")
@Data @NoArgsConstructor @AllArgsConstructor
public class User {
    @Id
    private String userId;
    @Column(name = "USER_NAME",unique = true, length = 20)
    private String username;
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private String password;
    @ManyToMany(mappedBy = "users", fetch = FetchType.EAGER)
    private List<Role> roles = new ArrayList<>();
}
```

Role.java

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Role {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "DESCRIPTION")
    private String desc;
    @Column(unique = true, length = 20)
    private String roleName;
    @ManyToMany(fetch = FetchType.EAGER)
    //@JoinTable(name = "USERS_ROLES")
    @ToString.Exclude
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<User> users = new ArrayList<>();
}
```

➢ Interfaces DAO basées sur Spring Data

RoleRepository.java

```java
package ma.enset.jpausers_roles.repositories;
import ...

@Repository
public interface RoleRepository extends JpaRepository<Role, Long> {
    Role findByRoleName(String roleName);
}
```

UserRepository.java

```java
package ma.enset.jpausers_roles.repositories;
import ...

@Repository
public interface UserRepository extends JpaRepository<User, String> {
    User findByUsername(String username);
}
```

➢ La couche service (métier)

➢ Interface

IUserService.java

```java
public interface IUserService {
    User addNewUser(User user);
    Role addNewRole(Role role);
    User findUserByUserName(String username);
    Role findRoleByRoleName(String roleName);
    void addRoleToUser(String username, String roleName);
    User autenticate(String username, String password);
}
```

➢ Implémentation

UserServiceImpl.java

```java
@Service
@Transactional
//pour faire l'injection via le constructeur avec paramètres
@AllArgsConstructor
public class UserServiceImpl implements IUserService {
    private UserRepository userRepository;
    private RoleRepository roleRepository;
    @Override
    public User addNewUser(User user) {
        user.setUserId(UUID.randomUUID().toString());
       // user.setPassword();
        return userRepository.save(user);
    }
    @Override public Role addNewRole(Role role) { return roleRepository.save(role); }
    @Override public User findUserByUserName(String username) { return userRepository.findByUsername(username); }
    @Override public Role findRoleByRoleName(String roleName) { return roleRepository.findByRoleName(roleName); }

    @Override
    public void addRoleToUser(String username, String roleName) {
        User user = findUserByUserName(username);
        Role role = findRoleByRoleName(roleName);
        if(user.getRoles()≠null) {
            //Dans OO ⟹ ManyToMany
            user.getRoles().add(role);  role.getUsers().add(user);
        }
        // car on la Transaction
        //userRepository.save(user);
    }

    @Override
    public User autenticate(String username, String password) {
       User user = findUserByUserName(username);
       if(user==null) throw new RuntimeException("Bad credentials");
       if(user.getPassword().equals(password))
           return user;
       throw new RuntimeException("Bad credentials");
    }
}
```

➢ Application

JpaUsersRolesApplication.java

```java
    @Bean
    CommandLineRunner start(IUserService userService)
    {
        return args → {
            User user = new User();
            user.setUsername("user1");
            user.setPassword("123456");
            userService.addNewUser(user);

            User user2 = new User();
            user2.setUsername("admin");
            user2.setPassword("123456");
            userService.addNewUser(user2);

            Stream.of("STUDENT", "USER", "ADMIN").forEach(role→{
                Role role1 = new Role();
                role1.setRoleName(role);
                userService.addNewRole(role1);
            });

            userService.addRoleToUser( username: "user1",  roleName: "STUDENT");
            userService.addRoleToUser( username: "user1",  roleName: "USER");
            userService.addRoleToUser( username: "admin",  roleName: "USER");
            userService.addRoleToUser( username: "admin",  roleName: "ADMIN");
```

```
45          try {
46              User u = userService.autenticate( username: "user1", password: "123456");
47              System.out.println(u.getUserId());
48              System.out.println(u.getUsername());
49              System.out.println("Roles ⟹ ");
50              u.getRoles().forEach(r→{
51                  System.out.println("Role : "+r);
52              });
53          }catch (Exception e)
54          {
55              e.printStackTrace();
56          }
57      };
58  }
59 }
```

➢ Résultat

user_id	password	user_name
53008f06-a7aa-4efe-bdcf-414d5c14c569	123456	admin
f897f2de-8af7-4c09-8c1f-4a86ebf6cb8d	123456	user1

roles_id	users_user_id
1	f897f2de-8af7-4c09-8c1f-4a86ebf6cb8d
2	f897f2de-8af7-4c09-8c1f-4a86ebf6cb8d
2	53008f06-a7aa-4efe-bdcf-414d5c14c569
3	53008f06-a7aa-4efe-bdcf-414d5c14c569

id	description	role_name
1	*NULL*	STUDENT
2	*NULL*	USER
3	*NULL*	ADMIN

➢ La couche web

• Le contrôleur Spring MVC

UserController.java

```
10  @RestController
11  public class UserController {
12      @Autowired
13      private IUserService userService;
14
15      @GetMapping("/users/{username}")
16      public User user(@PathVariable String username){
17          User user =  userService.findUserByUserName(username);
18          return user;
19      }
20  }
```

localhost:8081/users/admin


```
{
  "userId": "53008f06-a7aa-4efe-bdcf-414d5c14c569",
  "username": "admin",
  "roles": [
    {
      "id": 2,
      "desc": null,
      "roleName": "USER"
    },
    {
      "id": 3,
      "desc": null,
      "roleName": "ADMIN"
    }
  ]
}
```