Ecole Normale Supérieure de
l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Compte rendu TP

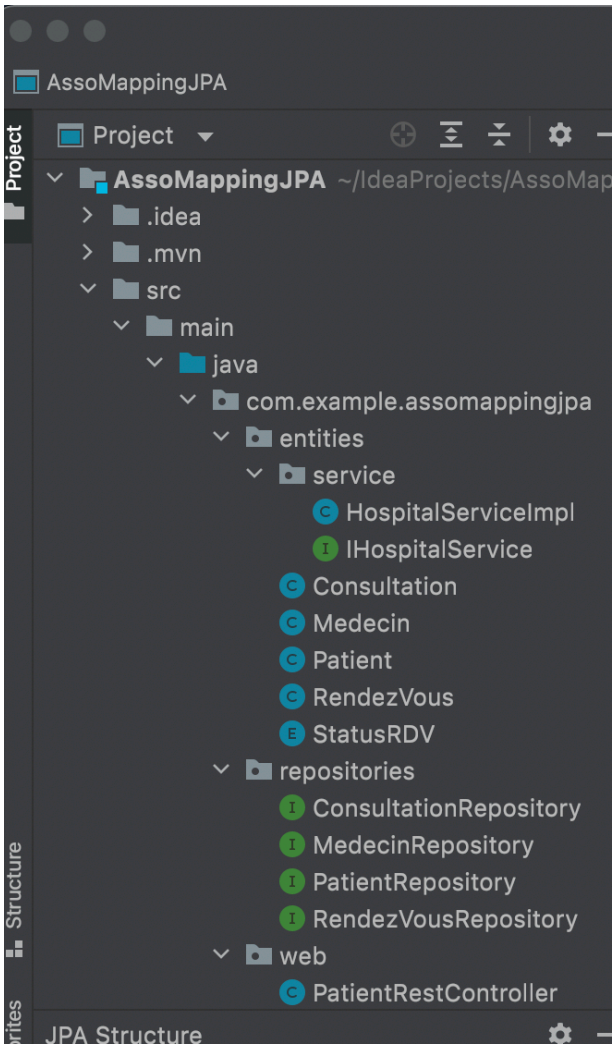
Gérer les associations entre les entités

Réalisé par :
SABRI Wissale

Introduction

Ce TP sert à appliquer nos connaissance en mapping object relationnel.

Structure d'application



1- Les dépendances dans pom

pom.xml (AssoMappingJPA)

```
17          <java.version>1.8</java.version>
18      </properties>
19      <dependencies>
20          <dependency>
21              <groupId>org.springframework.boot</groupId>
22              <artifactId>spring-boot-starter-data-jpa</arti
23          </dependency>
24          <dependency>
25              <groupId>org.springframework.boot</groupId>
26              <artifactId>spring-boot-starter-web</artifactI
27          </dependency>
28
29          <dependency>
30              <groupId>com.h2database</groupId>
31              <artifactId>h2</artifactId>
32              <scope>runtime</scope>
33          </dependency>
34          <dependency>
35              <groupId>org.projectlombok</groupId>
36              <artifactId>lombok</artifactId>
37              <optional>true</optional>
38          </dependency>
39          <dependency>
40              <groupId>org.springframework.boot</groupId>
41              <artifactId>spring-boot-starter-test</artifact
42              <scope>test</scope>
43          </dependency>
44      </dependencies>
```

Recommended plug
'Hibernate Core'.
Configure plugins...

On utilisera :

- Lombok
- Spring data JPA
- H2 database
- Spring web

2- Les entités

Consultation.java | Medecin.java | Patient.java | RendezVous.ja

```java
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.Collection;
import java.util.Date;

@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;
}
```

ation.properties | Consultation.java | Patient.java | RendezVous.java

```java
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.Collection;
import java.util.Date;

@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;

}
```

ation.properties | Consultation.java | RendezVous.java | StatusRDV.java

```java
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.Date;

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

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Temporal(TemporalType.DATE)
    private Date dateConsultation;
    private String rapport;
    @OneToOne
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private RendezVous rendezVous;
}
```

ation.properties | RendezVous.java | StatusRDV.java

```java
@Data
@NoArgsConstructor
@AllArgsConstructor
public class RendezVous {

    @Id

    private String  id;
    @Temporal(TemporalType.DATE)
    private Date date;
    @Enumerated(EnumType.STRING)
    private StatusRDV status;

    @ManyToOne
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Patient patient;
    @ManyToOne
    private Medecin medecin;

    @OneToOne(mappedBy = "rendezVous") // je veux avoir dans consultation
    private Consultation consultation;

}
```

3- Les repositories

Pour chacune des entités, on crée un repo comme ci-dessous qui hérite de la classe générique JpaRepository en spécifiant la classe et le type de l'id

ltationRepository.java | MedecinRepository.java | PatientRepository.java | RendezVousRepository.java

```java
package com.example.assomappingjpa.repositories;

import com.example.assomappingjpa.entities.Patient;
import com.example.assomappingjpa.entities.RendezVous;
import org.springframework.data.jpa.repository.JpaRepository;

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

4- Services

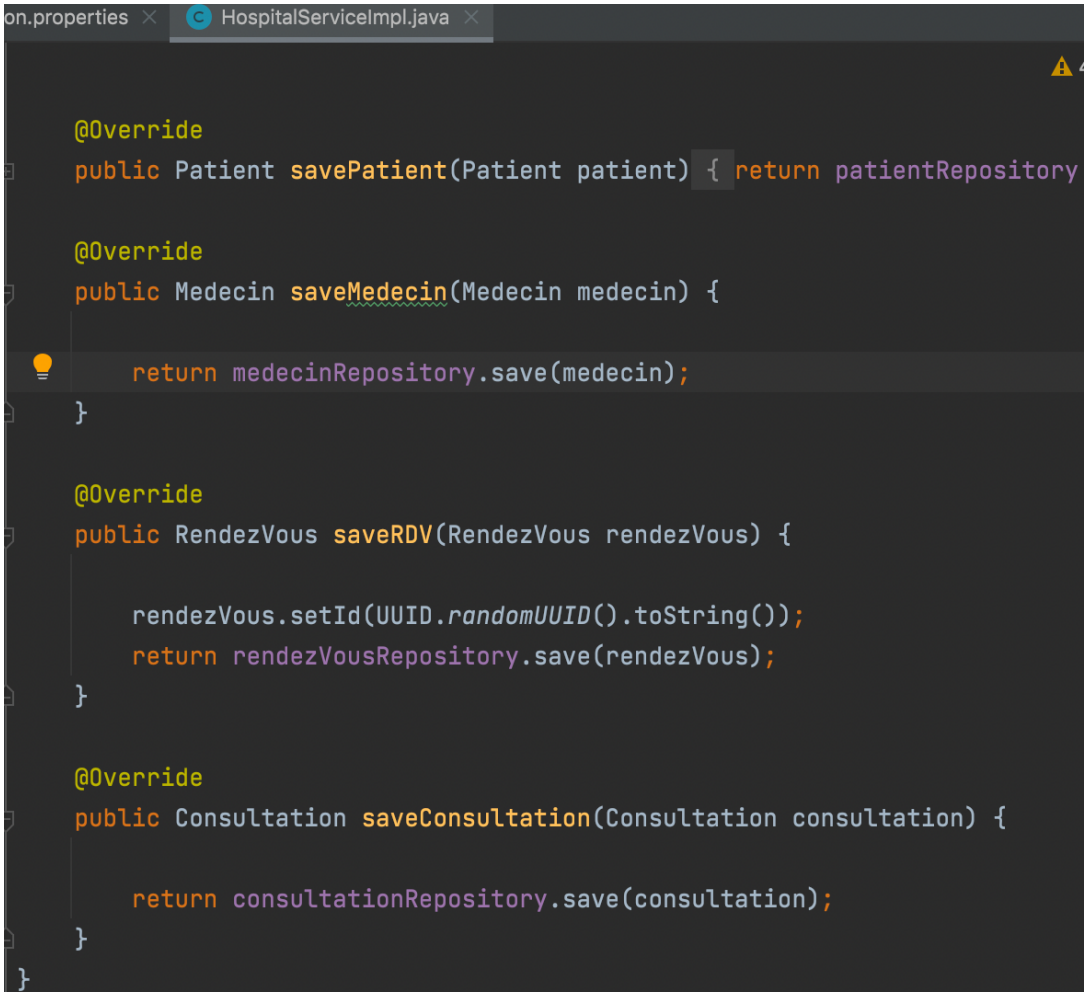
L'interface service sert à declarer les méthodes de manipulation des objets

application.properties | HospitalServiceImpl.java | IHospitalService.java

```java
package com.example.assomappingjpa.service;

import com.example.assomappingjpa.entities.Consultation;
import com.example.assomappingjpa.entities.Medecin;
import com.example.assomappingjpa.entities.Patient;
import com.example.assomappingjpa.entities.RendezVous;

public interface IHospitalService {

    Patient savePatient(Patient patient);
    Medecin saveMedecin(Medecin medecin);
    RendezVous saveRDV(RendezVous rendezVous);
    Consultation saveConsultation(Consultation consultation);

```

On implémente ensuite cette interface, et on se base sur des instances des repositories pour executer les différentes requetes (ajout, recherche,

modification, ..) Cette implementation va séparer l'aspect métier de la gestion des objets de la base de données .



5- Controlleur :

ation.properties | PatientRestController.java | AssoMappingJpaApplication.java

```java
package com.example.assomappingjpa.web;

import com.example.assomappingjpa.entities.Patient;
import com.example.assomappingjpa.repositories.PatientRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class PatientRestController {
    @Autowired
    private PatientRepository patientRepository;


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

6- propriétes de l'application

AssoMappingJpaApplication.java | application.properties

```
spring.datasource.url=jdbc:h2:mem:hospital
server.port=8082
spring.h2.console.enabled=true
spring.jpa.show-sql=true
```

7- Base de données H2