

**Nom & prénom** : DAHMANE walid

**Classe** : 4IIR G2

**Site** : CENTRE



# Rapport Java JEE

**Objet** : Spring data, JPA, Spring web, Lombok...

## ❖ Code source :

<https://github.com/DAHMANE-git/GestionDesPatients>

## ❖ Présentation :

	id	birth_date	is_sick	name	score
<input type="checkbox"/> Éditer Copier Supprimer	1	2021-03-30	0	Hassan	2300
<input type="checkbox"/> Éditer Copier Supprimer	2	2021-03-30	0	Farah	1200
<input type="checkbox"/> Éditer Copier Supprimer	3	2021-03-30	0	Janna	7600
<input type="checkbox"/> Éditer Copier Supprimer	4	2021-03-30	0	Imane	8500
<input type="checkbox"/> Éditer Copier Supprimer	6	2021-03-30	0	Hassan	2300

```
▼ tp_jpa [boot] [devtools]
  ▼ src/main/java
    ▼ ma.emsi.tp_jpa
      > PatientsController.java
      > TpJpaApplication.java
    ▼ ma.emsi.tp_jpa.entities
      > Patient.java
    ▼ ma.emsi.tp_jpa.repositories
      > PatientRepository.java
```

**Nom & prénom** : DAHMANE valid

**Classe** : 4IIR G2

**Site** : CENTRE

## ❖ Notes :

Le TP consiste à la compréhension des principes de bases de spring/hibernate ainsi que la liaison de données d'abord avec H2 puis avec MariaDB-MySQL.

Ce TP nous apprend comment faire un résonnement web tout en créant des Controlleurs des repositories ainsi que l'injection des dépendances. Le TP présente une classe Patient qui forme notre seule modele, grace a celle la on lui a créée une interface repository qui va englober nos fonctions d'accès a la base de données ainsi qu'un controlleur qui va faire la correspondance/Mapping entre ces fonctions avec des path URL (ex "/patients" et "/patients/{id}").

## ❖ Modele :

```
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @ToString
public class Patient {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotNull
    @Size(min = 5,max = 15)
    private String name;
    @Temporal(TemporalType.DATE)
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date dateNaissance;
    private boolean malade;
    @DecimalMin("4")
    private int score;
}
```

**Nom & prénom :** DAHMANE walid

**Classe :** 4IIR G2

**Site :** CENTRE

## ❖ Repository :

```
1 package org.sid.spingmvc.dao;
2
3 import java.awt.print.Pageable;
4
5
6
7
8
9 public interface PatientRepository extends JpaRepository<Patient, Long> {
10     public Page<Patient> findByNameContains(String mc,
11         org.springframework.data.domain.Pageable pageable);
12 }
```

L'interface appelée **PatientRepository** hérite de l'interface **JpaRepository** qui est une interface générique qui utilise deux types, le premier c'est le type de l'entité « **Patient** » et le deuxième c'est l'identifiant de l'entité de type **Long** qui représente l'Id. Cette classe va nous fournir toutes les fonctions nécessaires pour accéder à la base.

## ❖ Controller :

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

    @GetMapping("/listPatient")
    public List<Patient> list(Long id) {
        return patientRepository.findAll();
    }

    @GetMapping("/patients/{id}")
    public Patient getOne(@PathVariable Long id) {
        return patientRepository.findById(id).get();
    }
}
```

**Nom & prénom** : DAHMANE walid

**Classe** : 4IIR G2

**Site** : CENTRE

## ❖ Application JPA :

Dans cette classe on teste toutes les fonctionnalités de notre app.

```
public void run(String... args) throws Exception {
    /*patientRepository.save(new Patient(null, "Hassan", new
Date(),2300,false));
    patientRepository.save(new Patient(null, "Farah", new
Date(),1200,false));
    patientRepository.save(new Patient(null, "Janna", new
Date(),7600,false));
    patientRepository.save(new Patient(null, "Imane", new
Date(),8500,false));
    patientRepository.save(new Patient(null, "Yassine", new
Date(),2300,true));*/
    System.out.println("*****ALL*****");
    patientRepository.findAll().forEach(p->{
        System.out.println(p.toString());
    });
    System.out.println("*****By
Id*****");
    Patient p = patientRepository.findById(4L).get();
    System.out.println(p.toString());
    System.out.println("*****by
name*****");
    Page<Patient> p2 = patientRepository.findByNameContains("a",
PageRequest.of(0, 2));
    p2.forEach(pp->{
        System.out.println(pp.toString());
    });
    System.out.println("*****by
sickness*****");
    List<Patient> p3 = patientRepository.findByIsSick(true);
    p3.forEach(ps->{
        System.out.println(ps.toString());
    });
    //patientRepository.deleteById(5L);
    System.out.println("*****by name and
sickness*****");
    List<Patient> p4 =
patientRepository.findByNameContainsAndIsSick("H",true);
    p4.forEach(pp4->{
        System.out.println(pp4.toString());
    });

    System.out.println("*****Test
pages*****");
```

**Nom & prénom** : DAHMANE walid

**Classe** : 4IIR G2

**Site** : CENTRE

```
Page<Patient> pagePatients =
patientRepository.findAll(PageRequest.of(0,2));
System.out.println("Nombre des pages -
>" + pagePatients.getTotalPages());
List<Patient> listeP = pagePatients.getContent();
listeP.forEach(ppp->{
    System.out.println(ppp.toString());
});
}
```

La classe Patient se trouve dans un autre package `ma.emsi.tp_jpa.entities`.

Cette classe sera comme table dans la base de données sous le même nom avec un petit 's' à la fin, et elle va comporter comme champs :

- ❖ **id** : de type Long ; qui indique l'identifiant de chaque patient et qui s'incrémente automatiquement à travers les annotations utilisées `@Id`, `@GeneratedValue (strategy=GenerationType.IDENTITY)`.
- ❖ **nom** : de type String et qui sera enregistré comme colonne nommée NOM et de taille 25 (`@Column(name="NOM",length = 25)`).
- ❖ **datNaissance** : qui représente la date de naissance du patient de type Date (`@Temporal(TemporalType.DATE)`) pour obtenir que la date sans heure. La colonne dans la base de données va prendre le même nom que celui de l'attribut puisqu'on n'a pas spécifié de nom de colonne.
- ❖ **score** : de type Int.
- ❖ **malade** : un attribut boolean qui prend la valeur true si le patient est malade et false dans le cas contraire.

## ❖ Annotations :

**@data** : annotation Lombok qui permet de générer les getters et setters

**@NoArgsConstructor @AllArgsConstructor** : deux annotations permettant de générer les constructeurs de la classe l'un sans argument et l'autre avec des paramètres.

**@toString** : permet d'invoquer la méthode `toString()`.

**@table** : permet de donner un nom différent à la table en base que celui de l'entité.

**Nom & prénom** : DAHMANE walid

**Classe** : 4IIR G2

**Site** : CENTRE

**@Temporal** : permet de définir le format de la date.

**@Column** : nous aide a donner des specifications a la colonne associé a un attribut precis dans notre entite.

## ❖ Liaisons avec la base :

➔ Avec la base H2



```
Patient.java  *TpJpaApplication.java  PatientRepository.java  tp_jpa/pom.xml  *application.properties
1 spring.datasource.url=jdbc:h2:mem:DB_PATIENTS
2 server.port=8082
```

➔ Avec mySQL

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/db_patients?serverTimezone=UTC
2 spring.datasource.username= root
3 spring.datasource.password=
4 spring.jpa.show-sql= true
5 spring.jpa.hibernate.ddl-auto= update
6 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
```