

Pour créer notre application nous avons créé les classes suivantes et qui correspondent à des tables à une base de données qui sera créée automatiquement lors de l'exécution.

Dans toutes les classes, Les annotations `@NoArgsConstructor` et `@AllArgsConstructor` permet de générer automatiquement des constructeurs sans et avec paramètres.

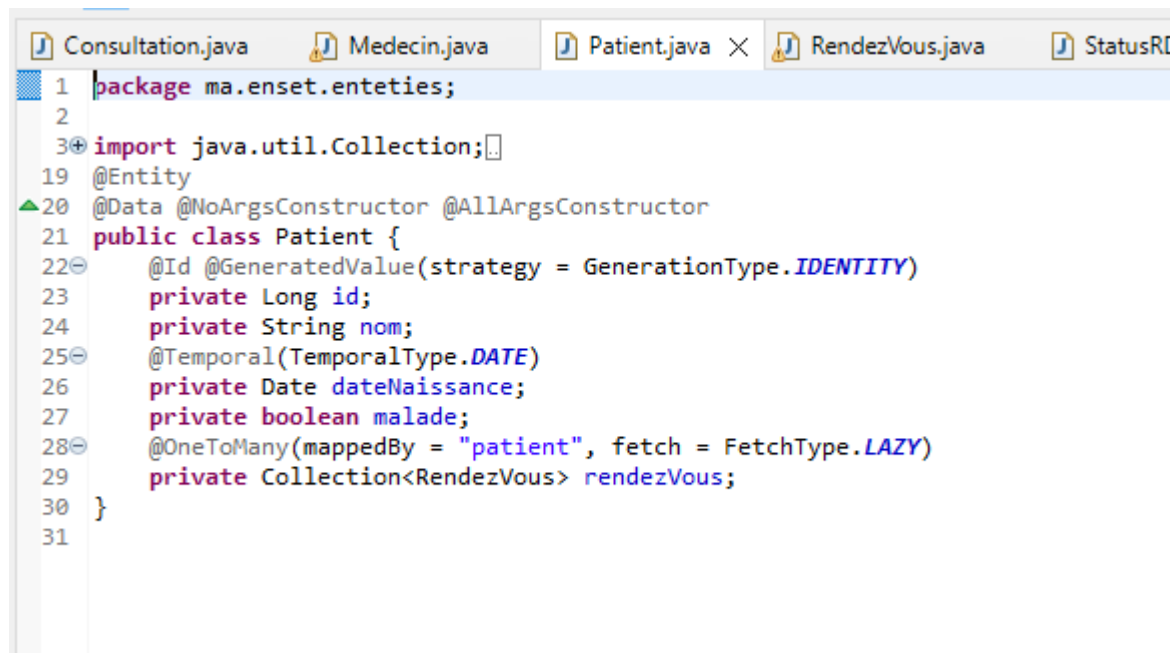
L'annotation `@Id` permet de déclarer de l'attribue est un clé primère de la table de la base de données;

La classe Patient :

Un patient peut avoir plusieurs rendez-vous.

L'annotation temporel permet de spécifier le format de la date dans la base de données.

L'option LAZY permet à ne recharger les informations des rendez-vous dans la mémoire que si l'utilisateur a besoin.



```
1 package ma.enset.enteties;
2
3 import java.util.Collection;
4
5 @Entity
6 @Data @NoArgsConstructor @AllArgsConstructor
7 public class Patient {
8     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
9     private Long id;
10    private String nom;
11    @Temporal(TemporalType.DATE)
12    private Date dateNaissance;
13    private boolean malade;
14    @OneToMany(mappedBy = "patient", fetch = FetchType.LAZY)
15    private Collection<RendezVous> rendezVous;
16 }
17
```

La classe médecin :

Un médecin peut également avoir plusieurs rendez-vous

L'option READ.ONLY permet a ne peut recharger les informations relatives aux rendez-vous lors d'un affichage JSON par exemple.

```
Consultation.java  Medecin.java  Patient.java  RendezVous.java
1 package ma.enset.enteties;
2
3 import java.util.Collection;
18
19 @Entity
20 @Data @NoArgsConstructor @AllArgsConstructor
21 public class Medecin {
22     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
23     private Long id;
24     private String nom;
25     private String email;
26     private String specialite;
27     @OneToMany(mappedBy = "medecin", fetch = FetchType.LAZY)
28     @JsonProperty(access = JsonProperty.Access.READ_ONLY)
29     private Collection<RendezVous> rendezVous;
30
31 }
32
```

La classe consultation : une consultation concerne un rendez-vous.

```
Consultation.java  Medecin.java  Patient.java  RendezVous.java
1 package ma.enset.enteties;
2
3 import java.util.Date;
16 @Entity
17 @Data @NoArgsConstructor @AllArgsConstructor
18 public class Consultation {
19     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
20     private Long id;
21     private Date dateConsultation;
22     private String rapport;
23     @OneToOne
24     @JsonProperty(access = JsonProperty.Access.READ_ONLY)
25     private RendezVous rendezVous;
26
27 }
28
```

La classe rendez-vous : un rendez-vous est effectué par un seul patient, et par un seul médecin, et concerne une et une seule consultation.

```
Consultation.java  Medecin.java  Patient.java  RendezVous.java  StatusRDV.java
1 package ma.enset.enteties;
2
3 import java.util.Date;
21 @Entity
22 @Data @NoArgsConstructor @AllArgsConstructor
23 public class RendezVous {
24     @Id @Column(length=50)
25     private String id;
26     private Date date;
27     @Enumerated(EnumType.STRING) //pour qu'on trouve done ou pending.. sinon on trouve des chiffres
28     private StatusRDV status;
29     @ManyToOne
30     @JsonProperty(access= JsonProperty.Access.READ_ONLY)
31     private Patient patient;
32     @ManyToOne
33     private Medecin medecin;
34     @OneToOne(mappedBy = "rendezVous", fetch = FetchType.LAZY)
35     private Consultation consultation;
36
37 }
38
```

L'énumération StatusRDV : permet de retourner un ensemble de valeur que peut prendre un statut de rendez-vous.

```
Consultation.java  Medecin.java  Patient.java  RendezVous.java  StatusRDV.java
1 package ma.enset.enteties;
2
3 public enum StatusRDV {
4     PENDING,
5     CANCELED,
6     DONE
7 }
8
```

Pour chacune de ces classe, nous avons créé des repository, qui vont hériter de JpaRepository pour gérer qu'on puisse utiliser toutes les méthodes classiques qui permettent de gérer les entités JPA.

```
Consultation.java  Medecin.java  Patient.java  RendezVous.java  StatusRDV.java
1 package ma.enset.Repositories;
2
3+ import org.springframework.data.jpa.repository.JpaRepository;
6
7 public interface ConsultationRepository extends JpaRepository<Consultation, Long> {
8
9 }
10
```

```
1 package ma.enset.Repositories;
2
3+ import java.util.List;
3
4
5 public interface MedecinRepository extends JpaRepository<Medecin, Long> {
6     List<Medecin> findByNom(String nom);
7 }
8
```

```
1 package ma.enset.Repositories;
2
3+ import java.util.List;
4
5 public interface PatientRepository extends JpaRepository<Patient, Long> {
6     List<Patient> findByNom(String nom);
7 }
8
```

```

1 package ma.enset.Repositories;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6
7
8 public interface RendezVousRepository extends JpaRepository<RendezVous, String> {
9
10 }
11

```

De plus nous avons créé une interface qui sert à rendre les services dont nous serons besoin d'appeler et d'utiliser dans notre application.

```

1 package ma.enset.service;
2
3 import ma.enset.enteties.Consultation;
4
5
6
7
8 public interface IHospitalService {
9     Patient savePatient(Patient patient);
10    Medecin saveMedecin(Medecin medecin);
11    RendezVous saveRendezVous(RendezVous rendezVous);
12    Consultation saveConsultation(Consultation consultation);
13 }
14

```

Dire une interface, c'est dire également une implémentation de cette interface :

Puisque le Id de la classe rendez-vous n'est pas de type long, alors nous avons générer un type string, UUID. Et c'est la raison pour la quelle nous avons préciser la longueur du champ id.

```

3+ import java.util.UUID;
7 @Service
8 @Transactional
9 public class HospitalServiceImpl implements IHospitalService{
0     private PatientRepository patientRepository;
1     private MedecinRepository medecinRepository;
2     private RendezVousRepository rendezVousRepository;
3     private ConsultationRepository consultationRepository;
4
5-     public HospitalServiceImpl(PatientRepository patientRepository, MedecinRepository medecinRepository,
6         RendezVousRepository rendezVousRepository, ConsultationRepository consultationRepository) {
7         this.patientRepository = patientRepository;
8         this.medecinRepository = medecinRepository;
9         this.rendezVousRepository = rendezVousRepository;
0         this.consultationRepository = consultationRepository;
1     }
2
3-     @Override
4     public Patient savePatient(Patient patient) {
5         return patientRepository.save(patient);
6     }
7
8-     @Override
9     public Medecin saveMedecin(Medecin medecin) {
0         return medecinRepository.save(medecin);
1     }
2
3-     @Override
4     public RendezVous saveRendezVous(RendezVous rendezVous) {
5         rendezVous.setId(UUID.randomUUID().toString());
6         return rendezVousRepository.save(rendezVous);
7     }
8
9-     @Override
0     public Consultation saveConsultation(Consultation consultation) {
1         return consultationRepository.save(consultation);
2     }
3
4 }
5

```

Donc pour voir le fonctionnement de ces méthodes, nous avons créé la classe suivante, pour créer des objets dans la base de données :

```

public class Seance3Application2 {

    public static void main(String[] args) {
        SpringApplication.run(Seance3Application2.class, args);
    }

    @Bean //execution au demarrage
    CommandLineRunner start(
        PatientRepository patientRepository,
        MedecinRepository medecinRepository,
        RendezVousRepository rendezVousRepository,
        ConsultationRepository consultationRepository) {
        return args -> {
            //pour les patients
            Stream.of("Ahlam", "Aya").forEach(nom ->{
                Patient patient= new Patient();
                patient.setDateNaissance(new Date());
                patient.setMalade(false);
                patient.setNom(nom);
                patientRepository.save(patient);
            });
            //les medecins
            Stream.of("Aicha", "Anas").forEach(nom ->{
                Medecin medecin= new Medecin();
                medecin.setSpecialite(Math.random()>0.5?"Cardio":"Dentiste");
                medecin.setEmail(nom+"@gmail.com");
                medecin.setNom(nom);
                medecinRepository.save(medecin);
            });
            //les rendez-vous
            Stream.of("Aicha", "Anas").forEach(nom ->{
                Medecin medecin= new Medecin();
                medecin.setSpecialite(Math.random()>0.5?"Cardio":"Dentiste");
                medecin.setEmail(nom+"@gmail.com");
                medecin.setNom(nom);
                medecinRepository.save(medecin);
            });
            Patient patient=patientRepository.findById(1L).orElse(null);
            List<Patient> patients=patientRepository.findByNom("ahlam");
            List<Medecin> medecins=medecinRepository.findByNom("Anas");
            RendezVous rendezVous= new RendezVous();

```

```

        medecinRepository.save(medecin);
    });
    Patient patient=patientRepository.findById(1L).orElse(null);
    List<Patient> patients=patientRepository.findByNom("ahlam");
    List<Medecin> medecins=medecinRepository.findByNom("Anas");
    RendezVous rendezVous= new RendezVous();
    Consultation consultation = new Consultation();

    rendezVous.setDate(new Date());
    rendezVous.setMedecin(medecinRepository.findAll().get(0));
    rendezVous.setPatient(patientRepository.findAll().get(0));
    rendezVous.setId(UUID.randomUUID().toString());
    rendezVous.setStatus(StatusRDV.DONE);
    rendezVousRepository.save(rendezVous);

    //RendezVous rendezVous1=rendezVousRepository.findById(1L).orElse(null);
    RendezVous rendezVous1=rendezVousRepository.findAll().get(0);
    consultation.setDateConsultation(new Date());
    consultation.setRapport("Rapport de consultation");
    consultation.setRendezVous(rendezVous1);
    consultationRepository.save(consultation);

};

```

De plus, il est indispensable de configurer le fichier application.properties pour mentionner le nom de la base de données, le dialectSQL, le port d'écout ... :

```

1 spring.datasource.url=jdbc:mysql://localhost:3306/DBA?createDatabaseIfNotExist=true
2 spring.datasource.username=root
3 spring.datasource.password=
4 server.port=8080
5 spring.jpa.hibernate.ddl-auto = create
6 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
7 spring.jpa.show = true
8
9
10

```

Voici alors les tables de la base de données :

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
consultation		1	MyISAM	utf8mb4_0900_ai_ci	3.1 KiB	-
medecin		4	MyISAM	utf8mb4_0900_ai_ci	2.2 KiB	-
patient		2	MyISAM	utf8mb4_0900_ai_ci	2.0 KiB	-
rendez_vous		1	MyISAM	utf8mb4_0900_ai_ci	4.1 KiB	-
4 tables	Sum	8	MyISAM	utf8mb4_0900_ai_ci	11.4 KiB	0 B

☐ Check all With selected:


```
SELECT * FROM `consultation`
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 ▼ Filter rows:

Options

		id	date_consultation	rapport	rendez_vous_id		
<input type="checkbox"/>	Edit	Copy	Delete	1	2022-03-10 23:39:35	Rapport de consultation	4f59754e-ae33-4784-b859-dc5f930e3e37

▲ —

```
SELECT * FROM `medecin`
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 ▼ Filter rows:

+ Options

			id	email	nom	specialite	
<input type="checkbox"/>	Edit	Copy	Delete	1	Aicha@gmail.com	Aicha	Cardio
<input type="checkbox"/>	Edit	Copy	Delete	2	Anas@gmail.com	Anas	Dentiste
<input type="checkbox"/>	Edit	Copy	Delete	3	Aicha@gmail.com	Aicha	Dentiste
<input type="checkbox"/>	Edit	Copy	Delete	4	Anas@gmail.com	Anas	Dentiste

```
SELECT * FROM `patient`
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 ▼ Filter rows:

+ Options

			id	date_naissance	malade	nom	
<input type="checkbox"/>	Edit	Copy	Delete	1	2022-03-10	0	Ahlam
<input type="checkbox"/>	Edit	Copy	Delete	2	2022-03-10	0	Aya

SELECT * FROM `rendez_vous`					
<input type="checkbox"/> Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]					
<input type="checkbox"/> Show all Number of rows: 25 ▼ Filter rows: <input type="text" value="Search this table"/>					
+ Options					
<div> <div>← T →</div> <div>▼ id</div> <div>date</div> <div>status</div> <div>medecin_id</div> <div>patient_id</div> </div>					
<input type="checkbox"/> Edit Copy Delete 4f59754e-ae33-4784-b859-dc5f930e3e37 2022-03-10 23:39:35 DONE 1 1					

Et pour voir la liste des patients on utilisant les requêtes HTTP, nous avons créé un contrôleur, dans le quel nous avons mentionner la methode get pour qu'elle nous permet de retourner tous les patients.

```
package ma.enset.web;

import java.util.List;

@RestController
public class PatientController {
    @Autowired
    private PatientRepository patientRepository;
    @GetMapping("/patients")
    public List<Patient> patientList(){
        return patientRepository.findAll();
    }
}
```

Donc le résultat de cette requête est le suivant :

```
{ "id":1,"nom":"Ahlam","dateNaissance":"2022-03-10","malade":false,"re
10T22:39:35.000+00:00","status":"DONE","patient":{"id":1,"nom":"Ahlam'
dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","status":"DONE",'
b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","status":"D
4784-b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","statu:
ae33-4784-b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","
[{"id":"4f59754e-aee33-4784-b859-dc5f930e3e37","date":"2022-03-10T22:3
10","malade":false,"rendezVous":[{"id":"4f59754e-aee33-4784-b859-dc5f9
{"id":1,"nom":"Ahlam","dateNaissance":"2022-03-10","malade":false,"re
10T22:39:35.000+00:00","status":"DONE","patient":{"id":1,"nom":"Ahlam'
dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","status":"DONE",'
b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","status":"D
4784-b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","statu:
ae33-4784-b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","
[{"id":"4f59754e-aee33-4784-b859-dc5f930e3e37","date":"2022-03-10T22:3
10","malade":false,"rendezVous":[{"id":"4f59754e-aee33-4784-b859-dc5f9
{"id":1,"nom":"Ahlam","dateNaissance":"2022-03-10","malade":false,"re
10T22:39:35.000+00:00","status":"DONE","patient":{"id":1,"nom":"Ahlam'
dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","status":"DONE",'
b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","status":"D
4784-b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","statu:
ae33-4784-b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","
[{"id":"4f59754e-aee33-4784-b859-dc5f930e3e37","date":"2022-03-10T22:3
10","malade":false,"rendezVous":[{"id":"4f59754e-aee33-4784-b859-dc5f9
{"id":1,"nom":"Ahlam","dateNaissance":"2022-03-10","malade":false,"re
10T22:39:35.000+00:00","status":"DONE","patient":{"id":1,"nom":"Ahlam'
dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","status":"DONE",'
b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","status":"D
4784-b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","statu:
ae33-4784-b859-dc5f930e3e37","date":"2022-03-10T22:39:35.000+00:00","
[{"id":"4f59754e-aee33-4784-b859-dc5f930e3e37","date":"2022-03-10T22:3
10","malade":false,"rendezVous":
```