

- 1- Pour réussir notre application, nous avons coché au départ un ensemble d'outils, la chose qui a généré un ensemble de dépendances.



```
7      <artifactId>spring-boot-starter-parent</artifactId>
8      <version>2.6.4</version>
9      <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>ma.enset</groupId>
12     <artifactId>Seance2</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <name>Seance2</name>
15     <description>Demo project for Spring Boot</description>
16     <properties>
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</artifactId>
23       </dependency>
24       <dependency>
25         <groupId>org.springframework.boot</groupId>
26         <artifactId>spring-boot-starter-web</artifactId>
27       </dependency>
28
29       <dependency>
30         <groupId>com.h2database</groupId>
31         <artifactId>h2</artifactId>
32         <scope>runtime</scope>
33       </dependency>
34       <dependency>
35         <groupId>mysql</groupId>
36         <artifactId>mysql-connector-java</artifactId>
37       </dependency>
38
39       <dependency>
40         <groupId>org.projectlombok</groupId>
41         <artifactId>lombok</artifactId>
42         <optional>true</optional>
43       </dependency>
44     </dependencies>
```

- 2- La Creation de la classe Patient qui est une entité JPA qui va correspondre à une table de la base de données.

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;

Et pour renommer le nom de la colone 'nom' à la base de données en peut utiliser l'annotation `@Column` en spécifiant le nom de la colone, et si nous voudrions ajouter d'autres spécifications comme la taille de cet attribue on peut préciser le `length`.

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

```
Seance2Application.j... Patient.java PatientRepository.java
1 package ma.enset.enteties;
2
3 import java.util.Date;
16 @Entity
17 @Data @NoArgsConstructor @AllArgsConstructor
18 public class Patient {
19     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
20     private Long id;
21     @Column(name="NOM", length=50)
22     private String nom;
23     @Temporal(TemporalType.DATE)
24     // JJ MM YYYY
25     private Date dateNaissance;
26     private boolean malade;
27     private int score;
28 }
29
```

- 3- Dans le PatientRepository qui étends de JpaRepository, on peut déclarer l'ensemble des méthodes que nous allons appeler. Quelques méthodes sont déjà prêtes à utiliser, et pour spécifier d'autres qui ne le sont pas, on peut utiliser les requêtes SQL qui vont définir nos méthodes créer.

```
Seance2Application.java Patient.java PatientRepository.java Seance2/pom.xml application.properties
1 package ma.enset.Repositories;
2
3 import java.util.Date;
4 public interface PatientRepository extends JpaRepository<Patient, Long> {
5     List<Patient> findByMalade(boolean malade);
6     Page<Patient> findByMalade(boolean malade, Pageable pageable);
7     List<Patient> findByMaladeAndScoreLessThan(boolean malade, int score);
8     List<Patient> findByMaladeIsTrueAndScoreLessThan(int score);
9     List<Patient> findByDateNaissanceBetweenAndMaladeIsTrueOrNomLike(Date date1, Date date2, String nom);
10    @Query("select p from Patient p where p.dateNaissance between :x and :y or p.nom like :z")
11    List<Patient> chercherPatients(@Param("x") Date date1, @Param("y") Date date2, @Param("z") String nom);
12    @Query("select p from Patient p where p.nom like :x and p.score < :y")
13    List<Patient> chercherPatients2(@Param("x") String nom, @Param("y") int score);
14
15
16
17 }
18
```

4- Dans cette classe nous pouvons exécuter les méthodes créées.

```
1
2
3 @SpringBootApplication
4 public class Seance2Application implements CommandLineRunner {
5     @Autowired
6     private PatientRepository patientRepository;
7
8     public static void main(String[] args) {
9         SpringApplication.run(Seance2Application.class, args);
10    }
11
12    @Override
13    public void run(String... args) throws Exception {
14        for(int i=0; i<100; i++) {
15            patientRepository.save(
16                new Patient(null, "ahlam", new Date(), Math.random()>0.5?true:false, 5)
17            );
18        }
19        //obtenir que 5 => pagination
20        //charger 0 par 1 pour passer à la page suivante
21        Page<Patient> patients=patientRepository.findAll(PageRequest.of(0, 5));
22        System.out.println("Total des pages"+patients.getTotalPages());
23        System.out.println("Total des elements"+patients.getTotalElements());
24        System.out.println("Numero de La page"+patients.getNumber());
25        List<Patient> byMalade= patientRepository.findByMalade(false);
26        List<Patient> chercher= patientRepository.chercherPatients2("%h%", 30);
27        Page<Patient> byMalade2= patientRepository.findByMalade(false,PageRequest.of(2, 3));
28        patients.forEach(patient->{
29            System.out.println(patient.getId());
30            System.out.println(patient.getNom());
31            System.out.println(patient.getScore());
32            System.out.println(patient.getDateNaissance());
33            System.err.println(patient.isMalade());
34        });
35        System.out.println();
36        Patient patient=patientRepository.findById(1L).orElse(null);
37        if(patient != null) {
38            System.out.println(patient.getNom());
39            System.err.println(patient.isMalade());
40        }
41        patient.setScore(12);
42    }
43}
```

5- Pour déclarer un ensemble de configurations, on a besoins de fichiers application.properties pour mentionner en titre d'exemple le nom de la base de données, le port, le dialectSQL ...




















































```
Seance2Application.java Patient.java PatientRepository.java Seance2/pom.xml application.properties
1 spring.datasource.url=jdbc:h2:mem:patient_db
2 spring.h2.console.enabled=true
3
4 <!-- pour my sql -->
5 <!-- spring.datasource.url=jdbc:mysql://localhost:3306/DBA?createDatabaseIfNotExist=true -->
6 <!-- spring.datasource.username=root-->
7
8 <!-- spring.datasource.password= -->
9 <!--server.port=8080 -->
10 <!-- spring.jpa.hibernate.ddl-auto = update -->
11 <!-- spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect -->
12 <!-- spring.jpa.show = true -->
13 <!-- masquer la ligne de h2 -->
14
```

6- Voici le résultat des données stockées dans la base de données.

Table	Action	Rows	Type	Collation	Size	Overhead
patient	★ Browse Structure Search Insert Empty Drop	99	MyISAM	utf8mb4_0900_ai_ci	6.7 KiB	28 B
1 table	Sum	99	MyISAM	utf8mb4_0900_ai_ci	6.7 KiB	28 B

☐ Check all / Check tables having overhead With selected: ▼

+ Options

<div>← T →</div>				id	date_naissance	malade	nom	score
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	2022-03-10	0	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	2022-03-10	1	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	2022-03-10	0	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	2022-03-10	1	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	2022-03-10	1	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	7	2022-03-10	0	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	8	2022-03-10	0	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	9	2022-03-10	0	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	10	2022-03-10	1	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	11	2022-03-10	1	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	12	2022-03-10	1	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	13	2022-03-10	0	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	14	2022-03-10	0	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	15	2022-03-10	1	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	16	2022-03-10	1	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	17	2022-03-10	0	ahlam	5
<input type="checkbox"/>	 Edit	 Copy	 Delete	18	2022-03-10	0	ahlam	5