Learn coding

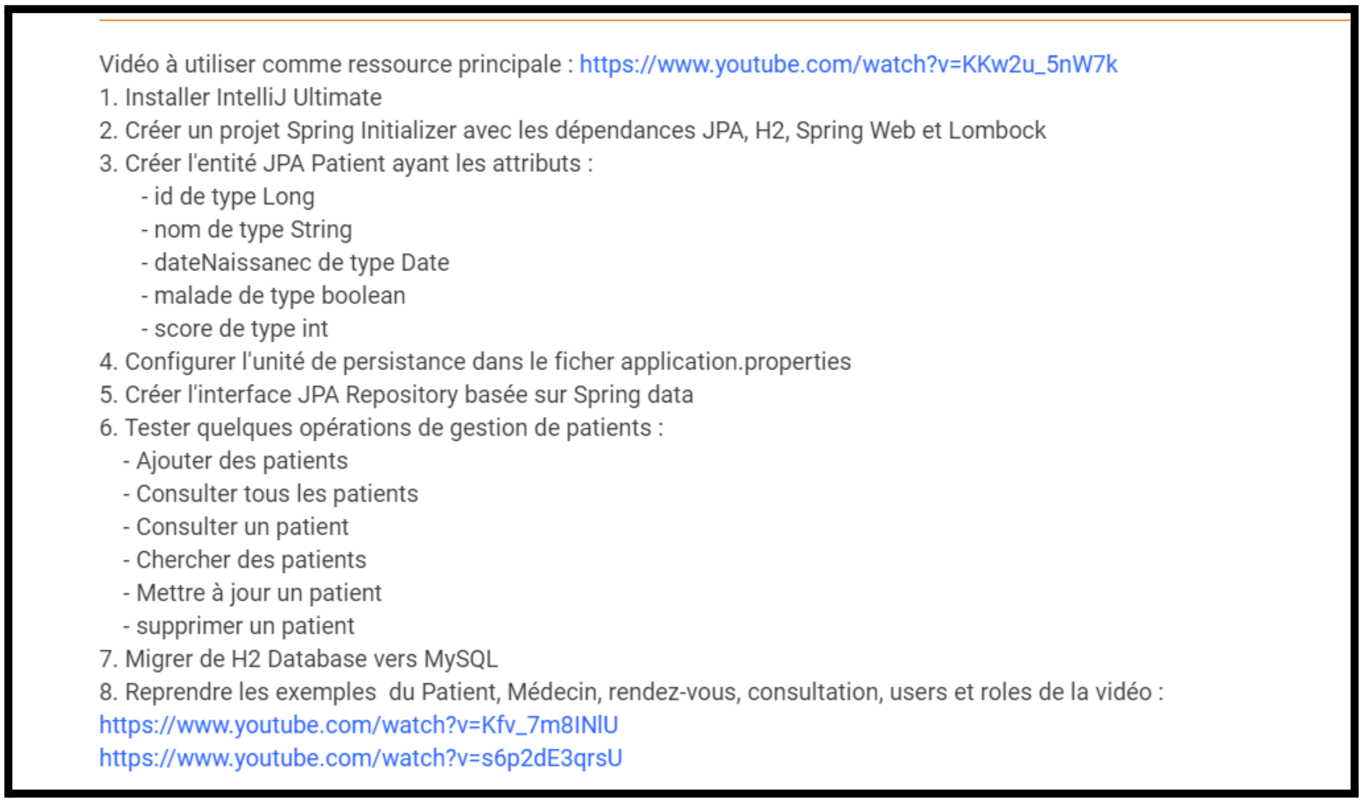
Java jee

Distributed Architectures and Middlewares

HAMZA BRAIMI

Activité Pratique N°3 : JPA Hibernate Spring Data

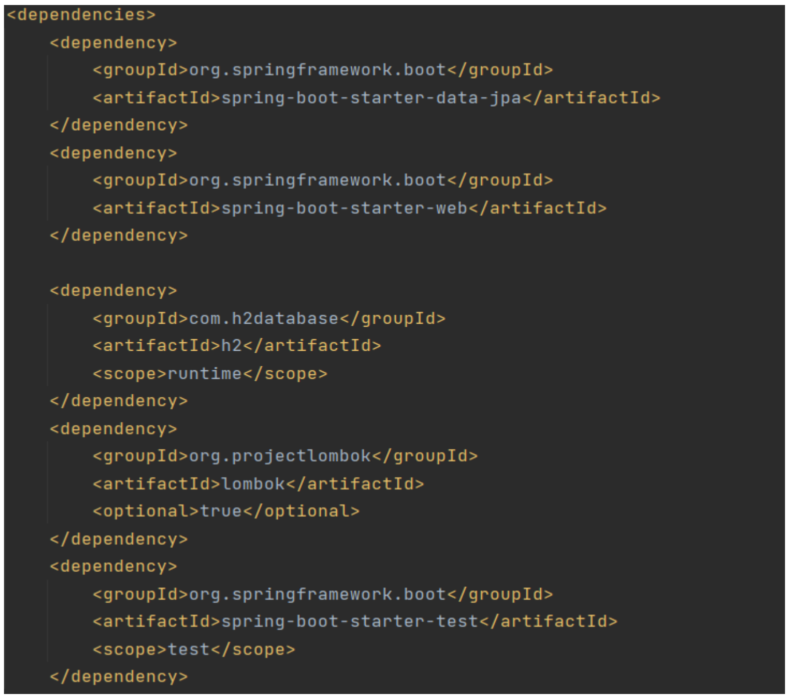
First Step :



Vidéo à utiliser comme ressource principale : https://www.youtube.com/watch?v=KKw2u_5nW7k
1. Installer IntelliJ Ultimate
2. Créer un projet Spring Initializer avec les dépendances JPA, H2, Spring Web et Lombock
3. Créer l'entité JPA Patient ayant les attributs :
    - id de type Long
    - nom de type String
    - dateNaissanec de type Date
    - malade de type boolean
    - score de type int
4. Configurer l'unité de persistance dans le ficher application.properties
5. Créer l'interface JPA Repository basée sur Spring data
6. Tester quelques opérations de gestion de patients :
    - Ajouter des patients
    - Consulter tous les patients
    - Consulter un patient
    - Chercher des patients
    - Mettre à jour un patient
    - supprimer un patient
7. Migrer de H2 Database vers MySQL
8. Reprendre les exemples du Patient, Médecin, rendez-vous, consultation, users et roles de la vidéo :
https://www.youtube.com/watch?v=Kfv_7m8INlU
https://www.youtube.com/watch?v=s6p2dE3qrsU

The first step is to try installing the dependencies using spring initializer:

- ✔ H2 database
- ✔ Jpa
- ✔ Spring Web
- ✔ Lombock



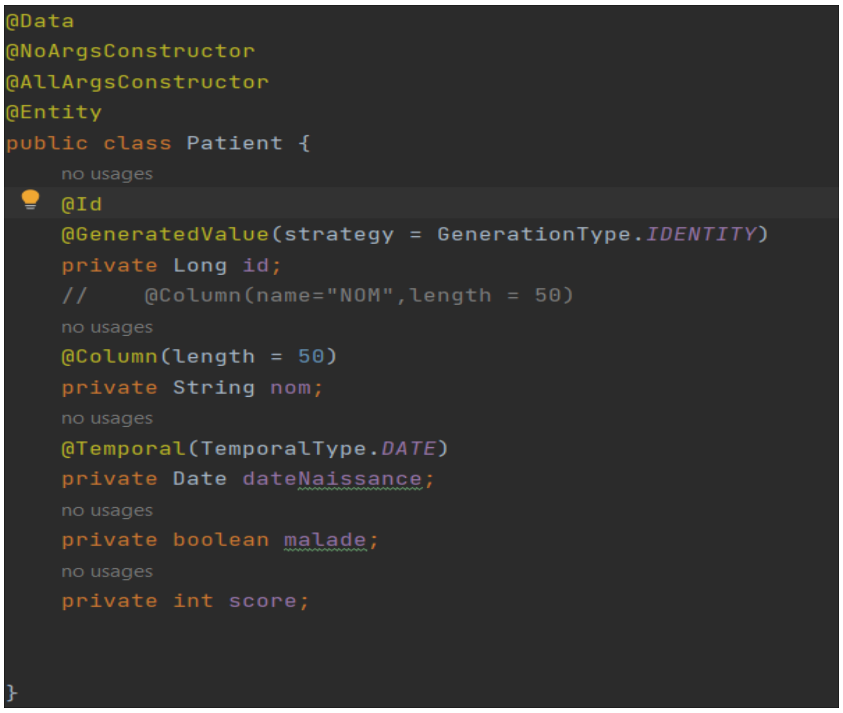
```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
```

Second Step :

Create the JPA Patient entity with the attributes:

- ✔ long type ID
- ✔ String type name
- ✔ DateBirth of type Date
- ✔ boolean patient
- ✔ int score

That is the implementation of this entity (name=” patient”)

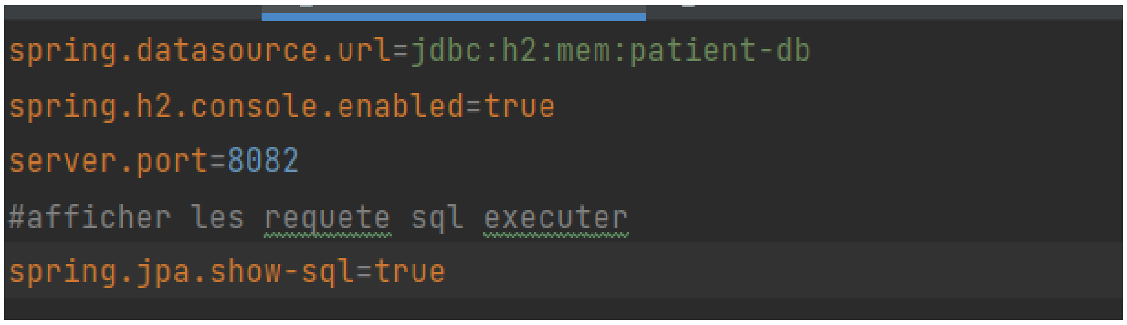


```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Patient {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    //    @Column(name="NOM",length = 50)
    no usages
    @Column(length = 50)
    private String nom;
    no usages
    @Temporal(TemporalType.DATE)
    private Date dateNaissance;
    no usages
    private boolean malade;
    no usages
    private int score;


}
```

@Data=@Data is a convenient shortcut annotation that bundles the features of @ToString, @EqualsAndHashCode, @Getter / @Setter
@NoArgsConstructor=Is used to generate the no-argument constructor for a class
@AllArgsConstructor=Generates a constructor with one parameter for every field in the class
@Entity=indicates that this class is a persistent class.

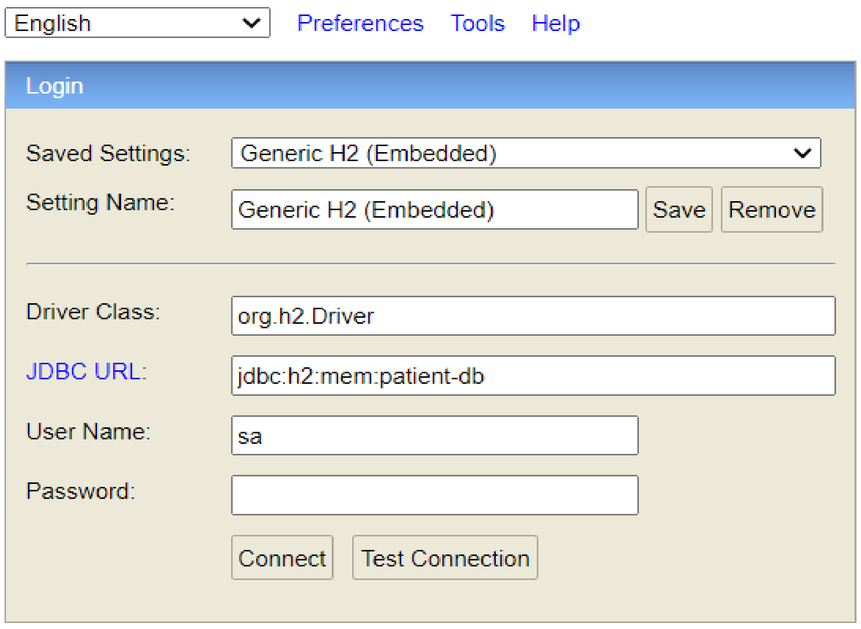
▯ Third Step :

In this level we should add some configuration in application.properties :



```
spring.datasource.url=jdbc:h2:mem:patient-db
spring.h2.console.enabled=true
server.port=8082
#afficher les requete sql executer
spring.jpa.show-sql=true
```

spring.datasource.url= is used to specify the URL of the database that the application should connect to. This URL typically contains information about the database server, port, and database name.
spring.h2.console.enabled=used to enable the H2 console, which is a web-based database management tool that is built into H2, a lightweight relational database that can be used for development and testing purposes.
server.port=to specify =Is used to specify the port on which the application should listen for incoming requests. By default, Spring Boot applications use port 8080.
spring.jpa.show-sql=Is used to enable the display of SQL statements generated by JPA/Hibernate

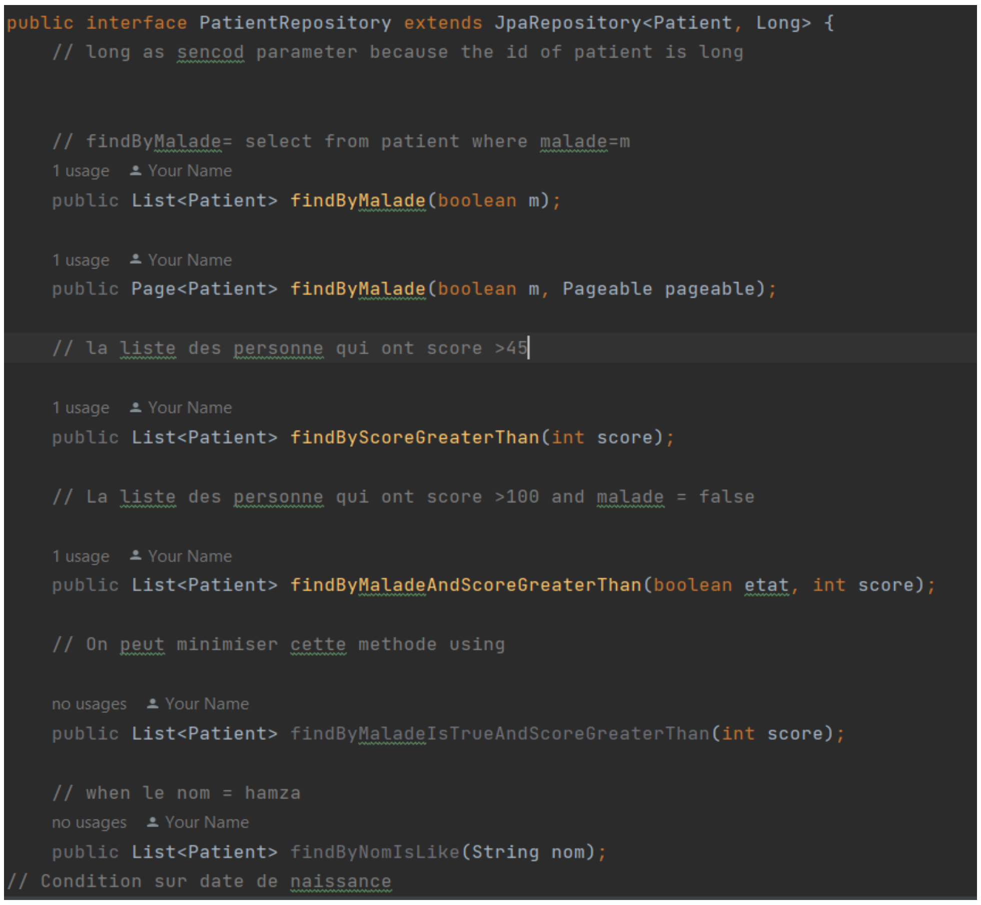


Saved Settings:	Generic H2 (Embedded)
Setting Name:	Generic H2 (Embedded)
Driver Class:	org.h2.Driver
JDBC URL:	jdbc:h2:mem:patient-db
User Name:	sa
Password:	

❖ That is the interface of h2 database

Fourth Step :

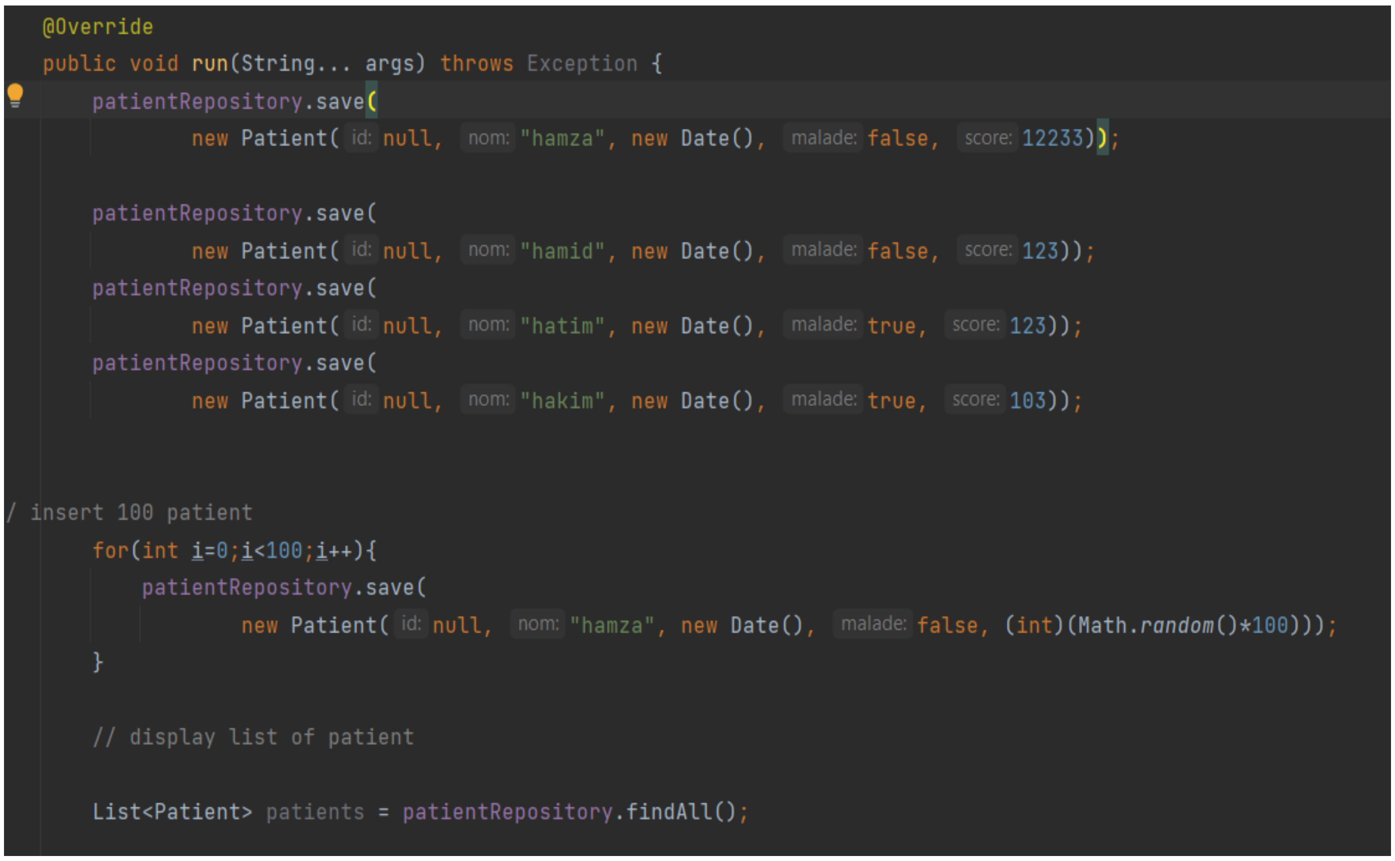
In this step we will create an interface patientRepository that extendsJpaRepository.**JpaRepository** is a JPA (Java Persistence API) specific extension of Repository. It contains the full API of CrudRepository and PagingAndSortingRepository. So it contains API for basic CRUD operations and also an API for pagination and sorting.



```
public interface PatientRepository extends JpaRepository<Patient, Long> {
    // long as sencod parameter because the id of patient is long


    // findByMalade= select from patient where malade=m
    1 usage   Your Name
    public List<Patient> findByMalade(boolean m);

    1 usage   Your Name
    public Page<Patient> findByMalade(boolean m, Pageable pageable);

    // la liste des personne qui ont score >45

    1 usage   Your Name
    public List<Patient> findByScoreGreaterThan(int score);

    // La liste des personne qui ont score >100 and malade = false

    1 usage   Your Name
    public List<Patient> findByMaladeAndScoreGreaterThan(boolean etat, int score);

    // On peut minimiser cette methode using

    no usages   Your Name
    public List<Patient> findByMaladeIsTrueAndScoreGreaterThan(int score);

    // when le nom = hamza
    no usages   Your Name
    public List<Patient> findByNomIsLike(String nom);
// Condition sur date de naissance
```

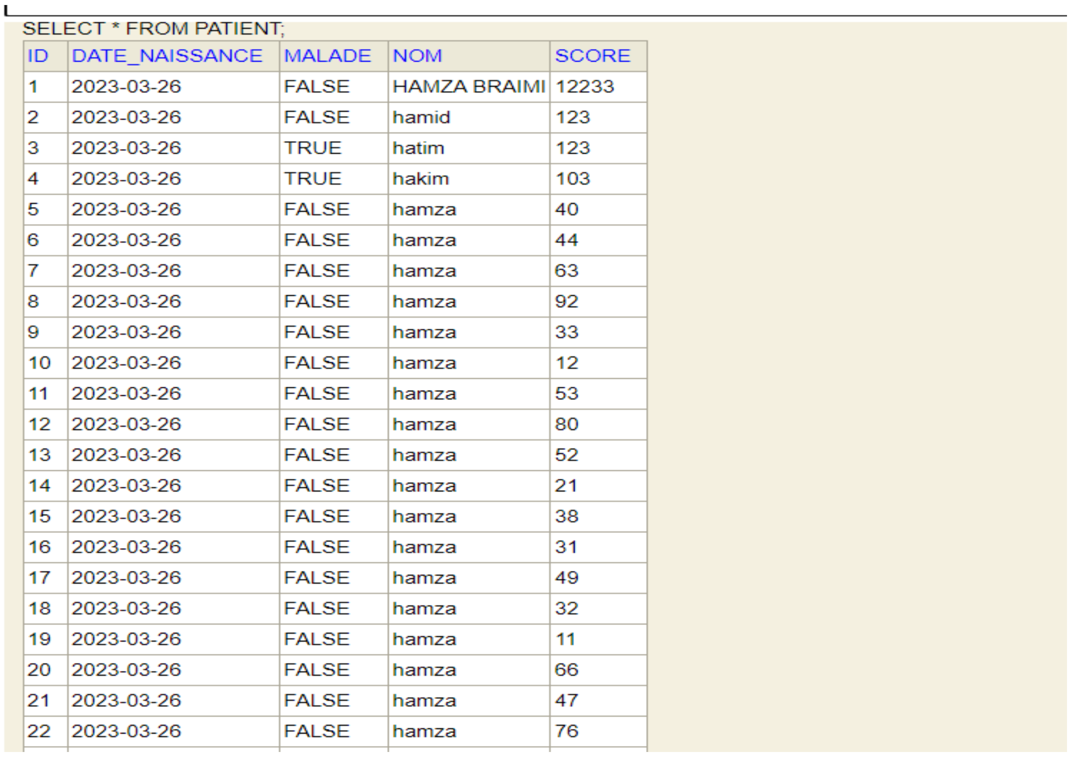
- ## Last Step :

Main class extends CommandLinner(CommandLineRunner is a simple Spring Boot interface with a run method. Spring Boot will automatically call the run method of all beans implementing this interface after the application context has been loaded.)



```
@Override
public void run(String... args) throws Exception {
    patientRepository.save(
            new Patient( id: null,  nom: "hamza", new Date(),  malade: false,  score: 12233));

    patientRepository.save(
            new Patient( id: null,  nom: "hamid", new Date(),  malade: false,  score: 123));
    patientRepository.save(
            new Patient( id: null,  nom: "hatim", new Date(),  malade: true,  score: 123));
    patientRepository.save(
            new Patient( id: null,  nom: "hakim", new Date(),  malade: true,  score: 103));


/ insert 100 patient
    for(int i=0;i<100;i++){
        patientRepository.save(
                new Patient( id: null,  nom: "hamza", new Date(),  malade: false, (int)(Math.random()*100)));
    }

    // display list of patient

    List<Patient> patients = patientRepository.findAll();
```

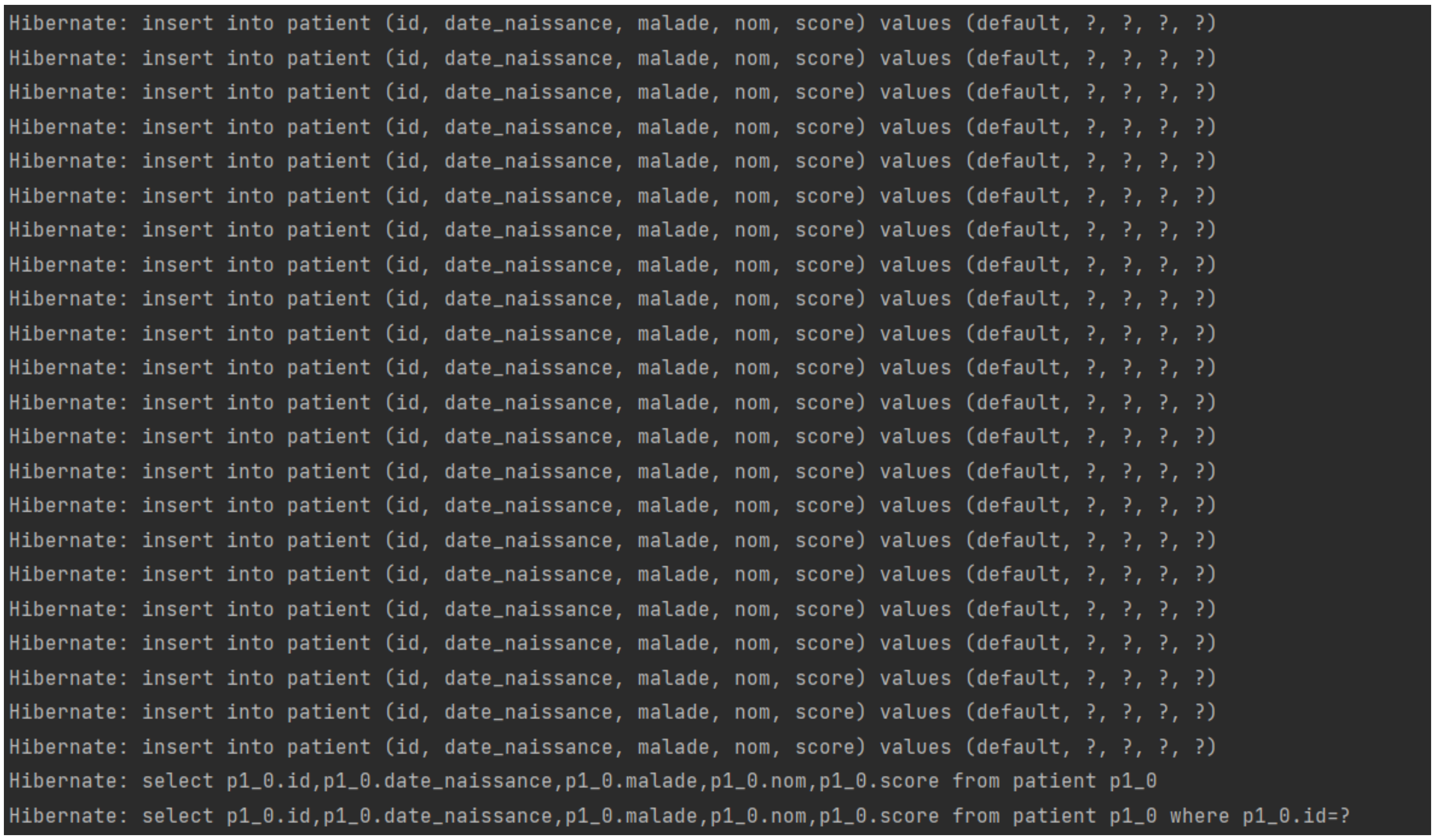
- Save Method is to insert patients in h2 database in that is the result:



SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM	SCORE
1	2023-03-26	FALSE	HAMZA BRAIMI	12233
2	2023-03-26	FALSE	hamid	123
3	2023-03-26	TRUE	hatim	123
4	2023-03-26	TRUE	hakim	103
5	2023-03-26	FALSE	hamza	40
6	2023-03-26	FALSE	hamza	44
7	2023-03-26	FALSE	hamza	63
8	2023-03-26	FALSE	hamza	92
9	2023-03-26	FALSE	hamza	33
10	2023-03-26	FALSE	hamza	12
11	2023-03-26	FALSE	hamza	53
12	2023-03-26	FALSE	hamza	80
13	2023-03-26	FALSE	hamza	52
14	2023-03-26	FALSE	hamza	21
15	2023-03-26	FALSE	hamza	38
16	2023-03-26	FALSE	hamza	31
17	2023-03-26	FALSE	hamza	49
18	2023-03-26	FALSE	hamza	32
19	2023-03-26	FALSE	hamza	11
20	2023-03-26	FALSE	hamza	66
21	2023-03-26	FALSE	hamza	47
22	2023-03-26	FALSE	hamza	76

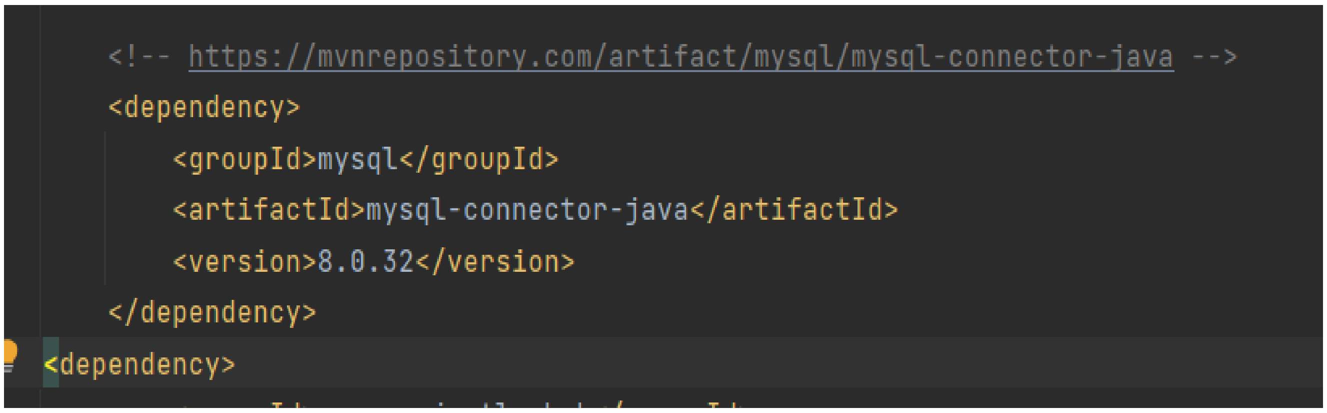
Executes SQl Queries



```
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: insert into patient (id, date_naissance, malade, nom, score) values (default, ?, ?, ?, ?)
Hibernate: select p1_0.id,p1_0.date_naissance,p1_0.malade,p1_0.nom,p1_0.score from patient p1_0
Hibernate: select p1_0.id,p1_0.date_naissance,p1_0.malade,p1_0.nom,p1_0.score from patient p1_0 where p1_0.id=?
```

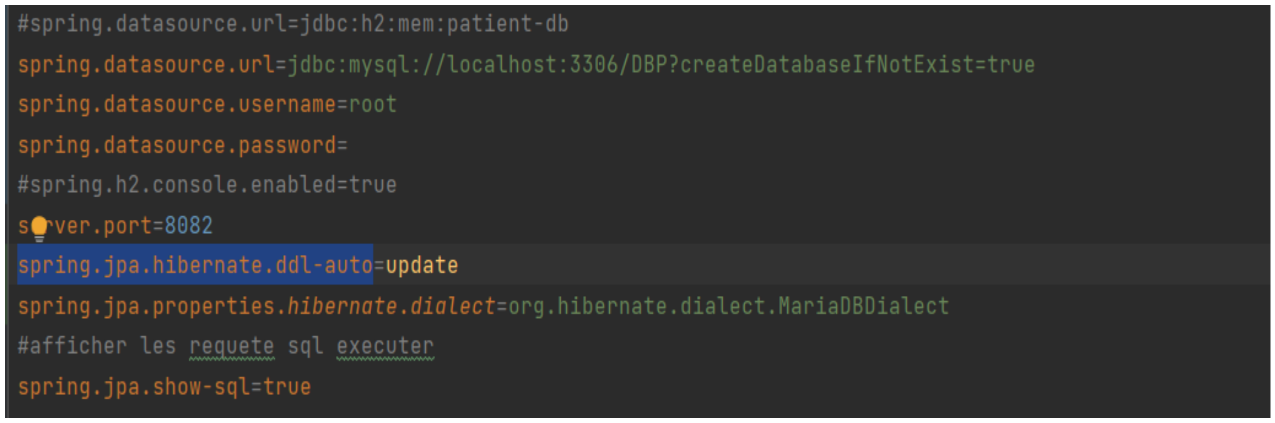
- **Question :How Switch from H2 database to MYsql**

  - The first step is to download mysql-connector-java dependency



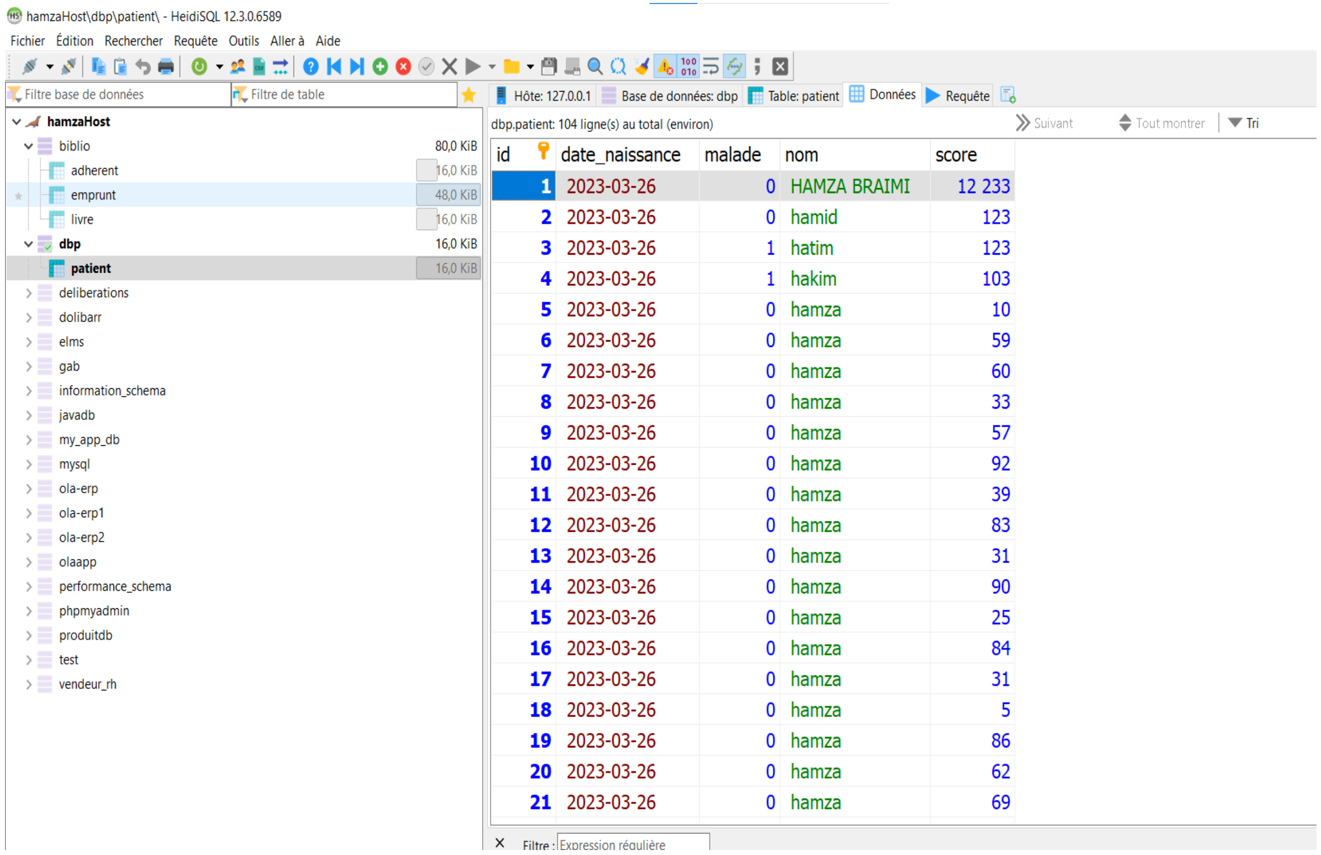
```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.32</version>
</dependency>
<dependency>
```

● The second step is to add mysql configuration in application.properties



```
#spring.datasource.url=jdbc:h2:mem:patient-db
spring.datasource.url=jdbc:mysql://localhost:3306/DBP?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
#spring.h2.console.enabled=true
server.port=8082
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
#afficher les requete sql executer
spring.jpa.show-sql=true
```

So Now we can get this result:



hamzaHost\dbp\patient\ - HeidiSQL 12.3.0.6589
Fichier Édition Rechercher Requête Outils Aller à Aide
Filtre base de données
Filtre de table
Hôte: 127.0.0.1
Base de données: dbp
Table: patient
Données
Requête
Suivant
Tout montrer
Tri
Filtre : Expression régulière

hamzaHost
- biblio 80,0 KiB
  - adherent 16,0 KiB
  - emprunt 48,0 KiB
  - livre 16,0 KiB
- dbp 16,0 KiB
  - patient 16,0 KiB
- deliberations
- dolibarr
- elms
- gab
- information_schema
- javadb
- my_app_db
- mysql
- ola-erp
- ola-erp1
- ola-erp2
- olaapp
- performance_schema
- phpmyadmin
- produitdb
- test
- vendeur_rh

dbp.patient: 104 ligne(s) au total (environ)

id	date_naissance	malade	nom	score
1	2023-03-26	0	HAMZA BRAIMI	12 233
2	2023-03-26	0	hamid	123
3	2023-03-26	1	hatim	123
4	2023-03-26	1	hakim	103
5	2023-03-26	0	hamza	10
6	2023-03-26	0	hamza	59
7	2023-03-26	0	hamza	60
8	2023-03-26	0	hamza	33
9	2023-03-26	0	hamza	57
10	2023-03-26	0	hamza	92
11	2023-03-26	0	hamza	39
12	2023-03-26	0	hamza	83
13	2023-03-26	0	hamza	31
14	2023-03-26	0	hamza	90
15	2023-03-26	0	hamza	25
16	2023-03-26	0	hamza	84
17	2023-03-26	0	hamza	31
18	2023-03-26	0	hamza	5
19	2023-03-26	0	hamza	86
20	2023-03-26	0	hamza	62
21	2023-03-26	0	hamza	69