Ecole Normale Supérieure de
l'Enseignement Technique Mohammedia
Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

**DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE**

# Compte rendu de l'activité pratique 3

**JEE : ORM / JPA / Hibernate / Spring Data**

## Filière :

**« Génie du Logiciel et des Systèmes Informatiques Distribués »**

## GLSID2

**Module : Architecture Distribuée et Middlewares**

Élaboré par :

ELMAJNI Khaoula

Encadré par :

M. YOUSSFI Mohammed

**Année Universitaire : 2021-2022**

# Introduction

Dans ce travail on va initier avec le Mapping Objet Relationnel (ORM), étant donné comme faire une relation et correspondance entre les objets d'une application et les données stockées dans une table dans une BDDR, cette opération à travers le JDBC.

**JDBC :** Java DataBase Connectivity qui permet à une application Java de communiquer avec n'importe quel SGBDR.

Egalement nous allons voir l'utilité d'utilisation des frameworks lors du développement :

- Le gain du temps
- La portabilité d'application avec n'importe quel SGBD
- Les performances

**JPA :** Java Persistance API est une spécification crée par SUN pour standardiser le ORM, et on peut considérer que ce JPA est une interface globale et comme implémentation nous avons le choix de travailler avec différents outils : Hibernate / TopLink / EclipseLink.

Nous avons testé les 2 moyens de Mapping Objet Relationnel :

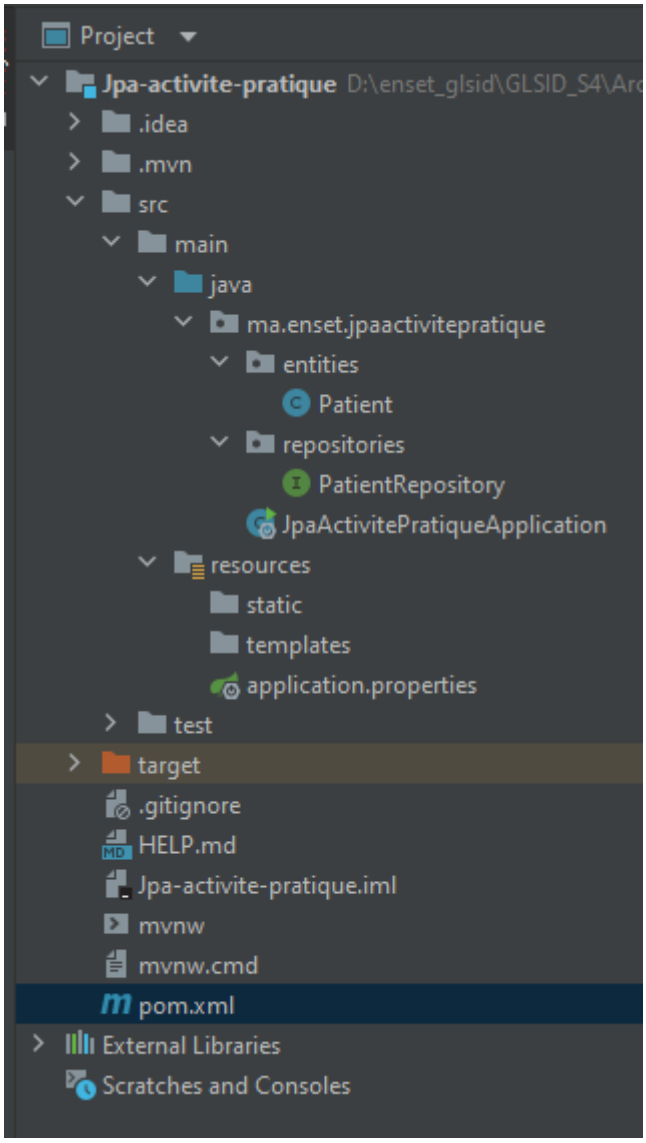
- Les fichirs XML de mapping
- Les annotations JPA

Java Bean : grain de café

Et pour ce faire nous nous avons basé sur le Spring Boot qui est une version de Spring framework, crée principalement pour les applications micro-services, et il se base sur l'auto-configuration.

**Spring Data :** un module de Spring qui facilite le Mapping Objet Relationnel basé sur JPA.

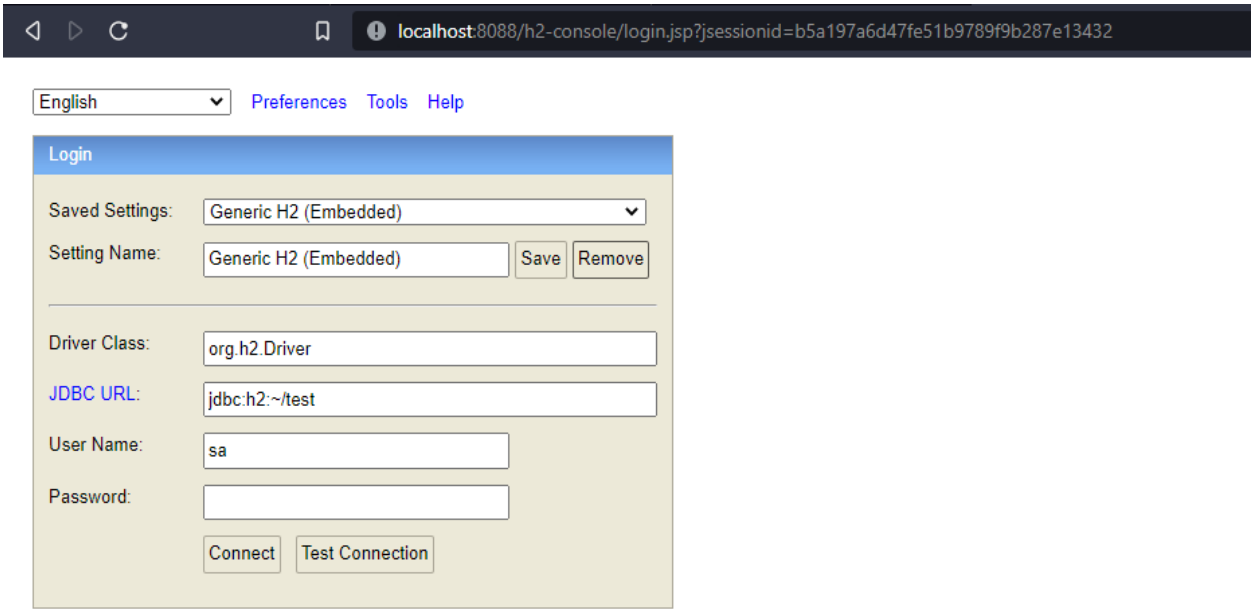
# L'architecture du projet :



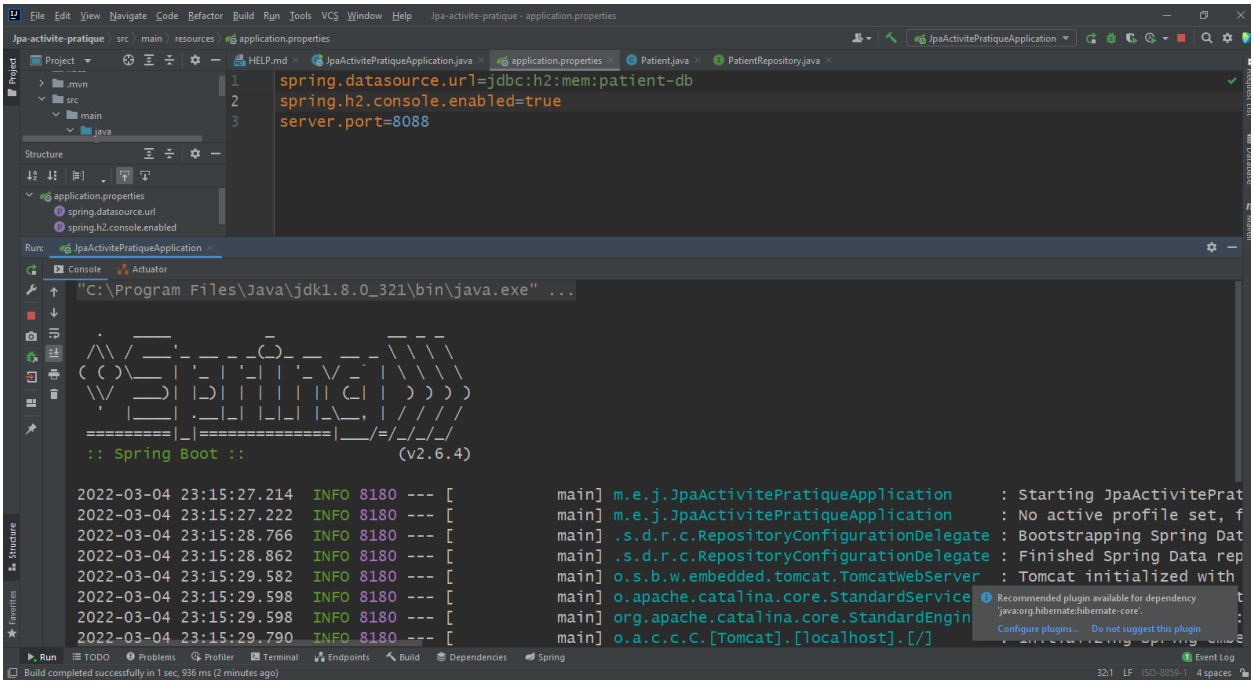

Après la création du projet le fichier de configuration 'application.properties' est comme suit :

H2 est une base de données mémoire pour le test.

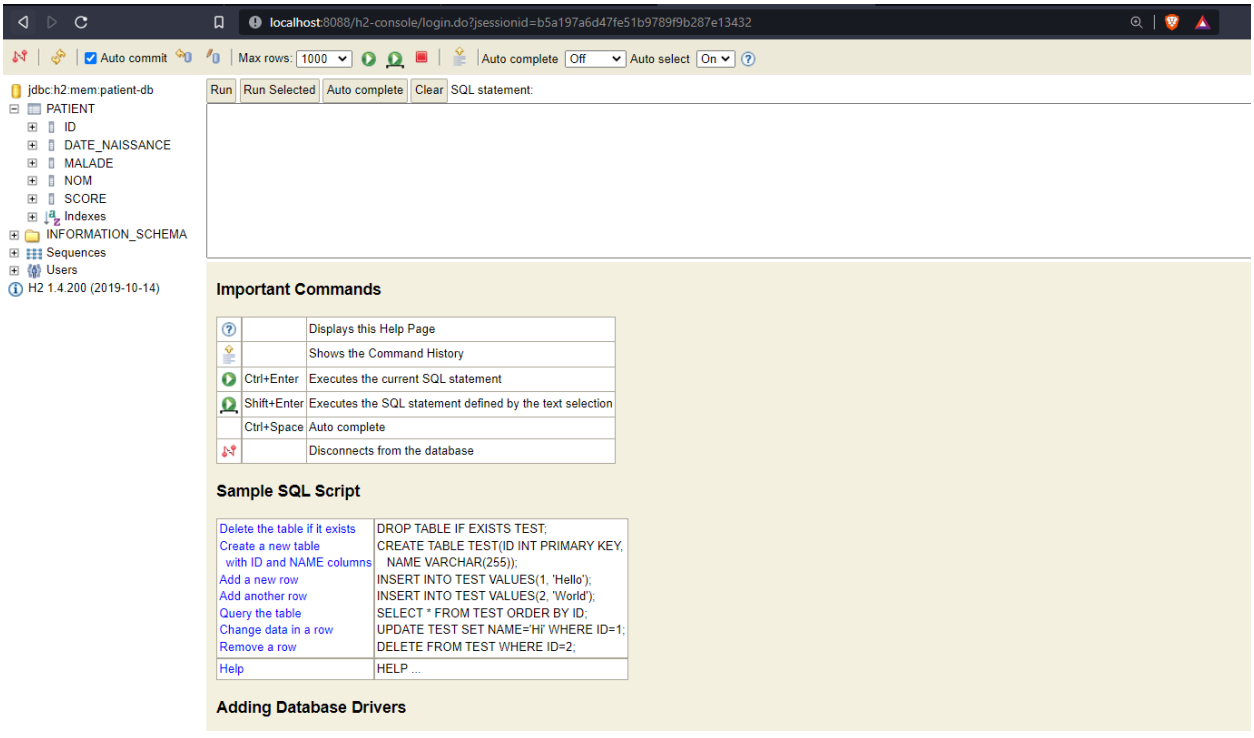
```
spring.datasource.url=jdbc:h2:mem:patient-db
spring.h2.console.enabled=true
spring.jpa.show-sql=true
server.port=8088
```

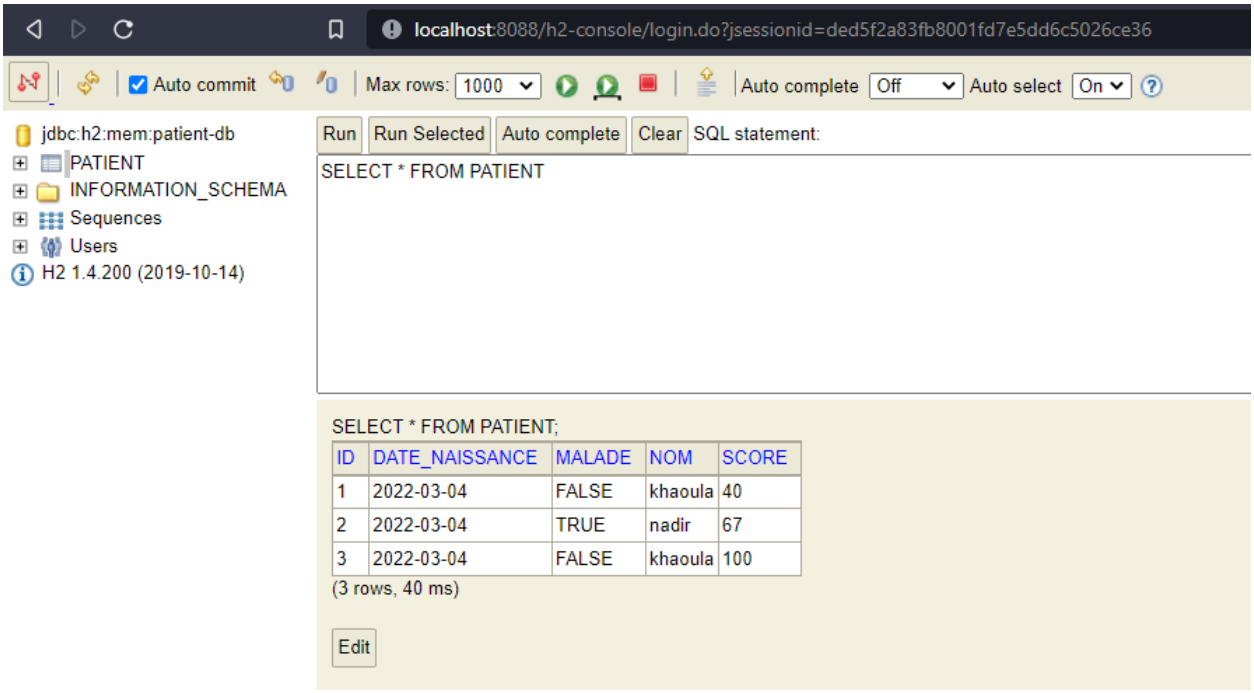

localhost:8088/h2-console/login.jsp?jsessionid=b5a197a6d47fe51b9789f9b287e13432
English
Preferences
Tools
Help
Login
Saved Settings:
Generic H2 (Embedded)
Setting Name:
Generic H2 (Embedded)
Save
Remove
Driver Class:
org.h2.Driver
JDBC URL:
jdbc:h2:~/test
User Name:
sa
Password:
Connect
Test Connection

# L'exécution :

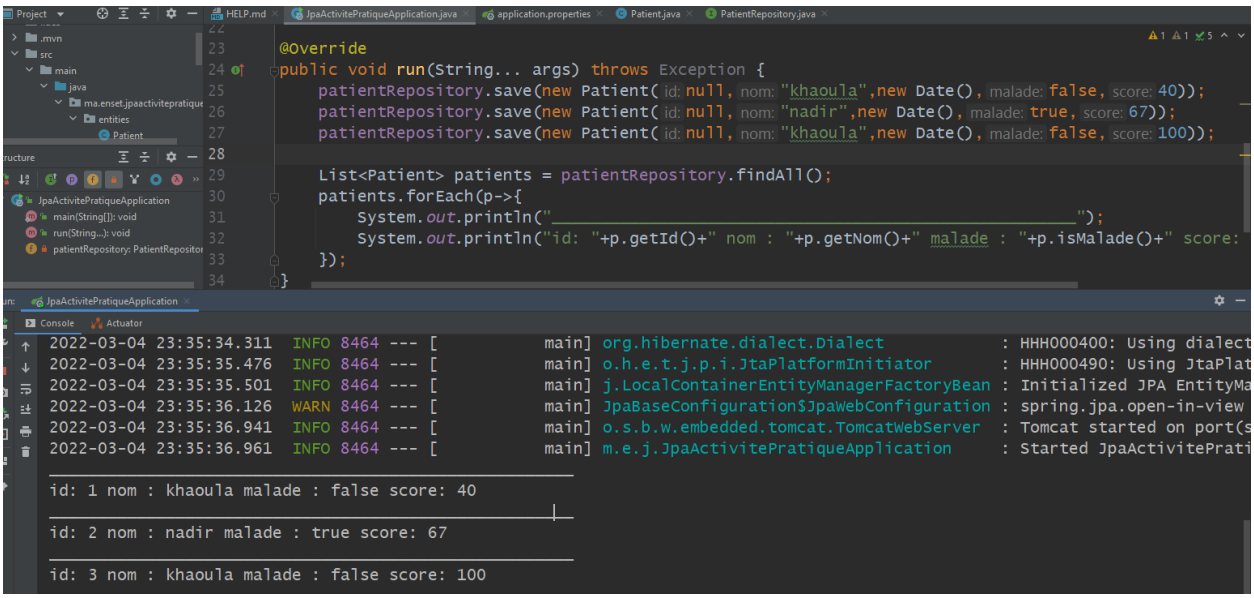



La base de données sous H2:






Récupération des données :




On a utilisé la pagination pour faciliter la récupération des données qui sont un peu costaud, et grace à cet outil on peut récupérer des pages de données avec un nombre de données bien précis :

```
Page<Patient> patients = patientRepository.findAll(PageRequest.of( page: 0, size: 5));
System.out.println("Total elements = "+patients.getTotalElements());
System.out.println("Total pages = "+patients.getTotalPages());
System.out.println("Numero de page = "+patients.getNumber());
List<Patient>content = patients.getContent();
```

La récupération des données grace à HQL(Hibernate Query Language) :

```
//une autre façon de récupération des données grace à HQL
@Query("select p from Patient p where p.dateNaissance between :x and :y or p.nom like :z")
List<Patient> chercherPatients(@Param("x") Date d1, @Param("y") Date d2,@Param("z") String keyword);

@Query("select p from Patient p where p.nom like :x and p.score < :y")
List<Patient> chercherPatients(@Param("x") String keyword, @Param("y") int scoreMin );
```

Nous avons ajouté d'autres méthodes en les déclarant à l'intérieur de l'interface JPARepository, sans avoir besoin de les implémenter et ça était attéignable grace à Spring Data :

```
package ma.enset.jpaactivitepratique.repositories;

import ma.enset.jpaactivitepratique.entities.Patient;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.Date;
import java.util.List;

//Spring data
public interface PatientRepository extends JpaRepository <Patient,Long>{
    //select * from patient where malade = m;
    List<Patient> findByMalade(boolean m);

    //retourne les personnes malades et la page (numero et size)
    Page<Patient> findByMalade(boolean m, Pageable pageable);

    List<Patient> findByMaladeAndScoreLessThan(boolean m ,int score);

    List<Patient> findByMaladeIsTrueAndScoreLessThan(int score);
```

```
    List<Patient> findByDateNaissanceBetween(Date d1, Date d2);

    List<Patient> findByDateNaissanceBetweenAndMaladeIsTrueOrNomLike(Date
d1, Date d2,String keyword);

    //une autre façon de récupération des données grace à HQL
    @Query("select p from Patient p where p.dateNaissance between :x and :y or
p.nom like :z")
    List<Patient> chercherPatients(@Param("x") Date d1, @Param("y") Date
d2,@Param("z") String keyword);

    @Query("select p from Patient p where p.nom like :x and p.score < :y")
    List<Patient> chercherPatients(@Param("x") String keyword, @Param("y") int
scoreMin );
}
```

Basculement vers MySQL

```
spring.datasource.url=jdbc:mysql://localhost:3306/patients?createDatabaseIfNotExist= true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
#spring.h2.console.enabled=true
spring.jpa.show-sql=true
server.port=8088
```

La base de données dans MySQL PHPMyAdmin :

| | id | date_naissance | malade | nom | score |
|---|---|---|---|---|---|
| Edit Copy Delete | 1 | 2022-03-05 | 1 | khaoula | 1928 |
| Edit Copy Delete | 2 | 2022-03-05 | 1 | khaoula | 11 |
| Edit Copy Delete | 3 | 2022-03-05 | 1 | khaoula | 59 |
| Edit Copy Delete | 4 | 2022-03-05 | 1 | khaoula | 44 |
| Edit Copy Delete | 5 | 2022-03-05 | 1 | khaoula | 33 |
| Edit Copy Delete | 6 | 2022-03-05 | 1 | khaoula | 22 |
| Edit Copy Delete | 7 | 2022-03-05 | 1 | khaoula | 27 |
| Edit Copy Delete | 8 | 2022-03-05 | 0 | khaoula | 69 |
| Edit Copy Delete | 9 | 2022-03-05 | 1 | khaoula | 81 |
| Edit Copy Delete | 10 | 2022-03-05 | 0 | khaoula | 89 |
| Edit Copy Delete | 11 | 2022-03-05 | 0 | khaoula | 38 |
| Edit Copy Delete | 12 | 2022-03-05 | 1 | khaoula | 21 |
| Edit Copy Delete | 13 | 2022-03-05 | 1 | khaoula | 67 |
| Edit Copy Delete | 14 | 2022-03-05 | 0 | khaoula | 89 |
| Edit Copy Delete | 15 | 2022-03-05 | 0 | khaoula | 33 |
| Edit Copy Delete | 16 | 2022-03-05 | 1 | khaoula | 63 |
| Edit Copy Delete | 17 | 2022-03-05 | 1 | khaoula | 75 |
| Copy Delete | 18 | 2022-03-05 | 0 | khaoula | 10 |

Console

## Conclusion

Ce travail nous a permis de bien connaitre comment le Mapping Objet Relationnel se fait au niveau du Java grace à l'API JPA, et on a initié avec Hibernate, et bien évidement la manière de configuration d'un projet Spring pour faciliter la tache au développeur.