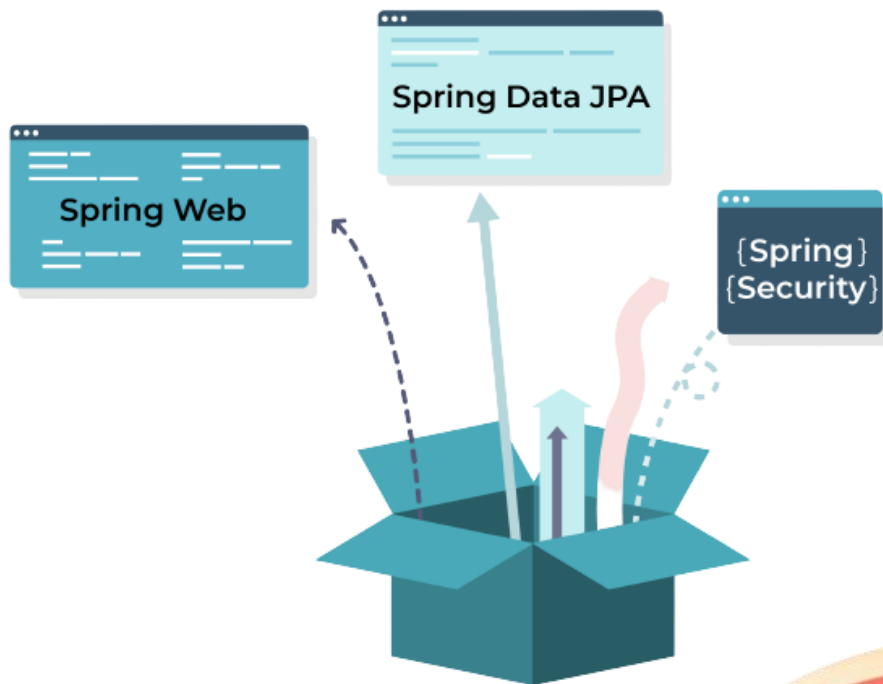


Compte rendu de :

SEANCE 03

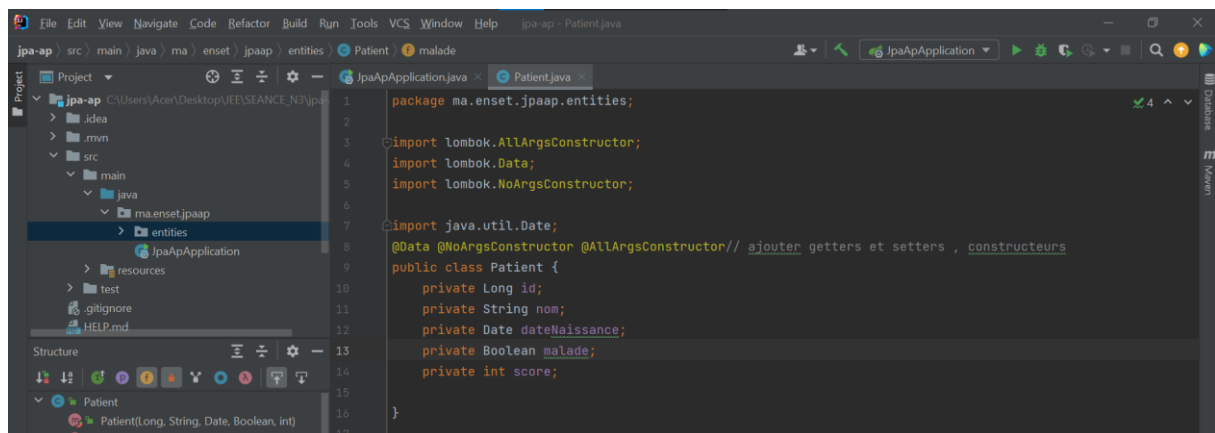


Réalisé Par : AYOUIJIL Soukayna
Département : Mathématiques et Informatique
Filière : II-BDCC
Professeur : M. Mohamed YOUSSEFI

Notions plus importantes :

- Pourquoi il faut utiliser des Framework ORM
- Le but d'Hibernate est de libérer le développeur de 95% des tâches de programmation liées à la persistance des données communes.
- Hibernate assure la portabilité de votre application si vous changez de SGBD.
- Hibernate propose au développeur des méthodes d'accès aux bases de données plus efficaces que ce qui devrait rassurer les développeurs.
- JPA est une spécification créée par Sun pour standardiser le mapping Objet relationnel.
- JPA définit un ensemble d'interfaces, de classes abstraites et d'annotations qui permettent la description du mapping objet relationnel
- Il existe plusieurs implémentations de JPA : Hibernate, Toplink, IBatis et EclipseLink
- Spring est Framework qui assure l'inversion de contrôle
- Spring peut s'occuper du code technique comme la configuration de JPA et la gestion des transactions.
- Avec Spring on peut simplifier le code de notre application en lui déléguant des tâches techniques.
- Spring Boot est une version de Spring qui permet de simplifier à l'extrême, entre autres : La gestion des dépendances Maven et Auto Configuration

Etape 1 : Gestion des entités – Patient

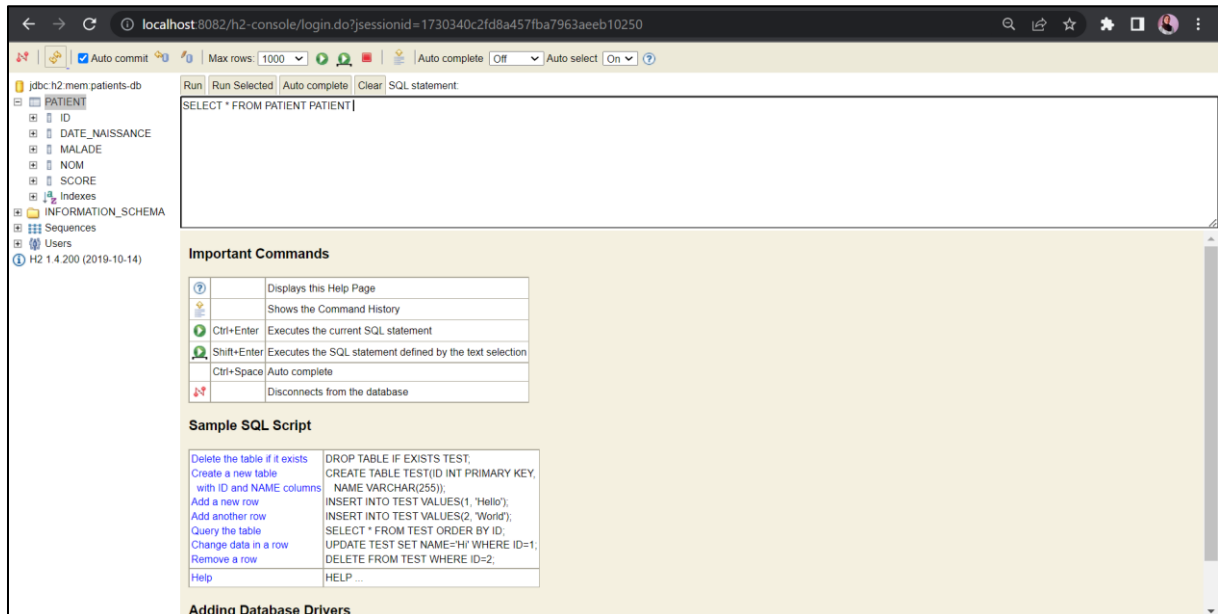


Etape 2 : Démarrer l'application Spring

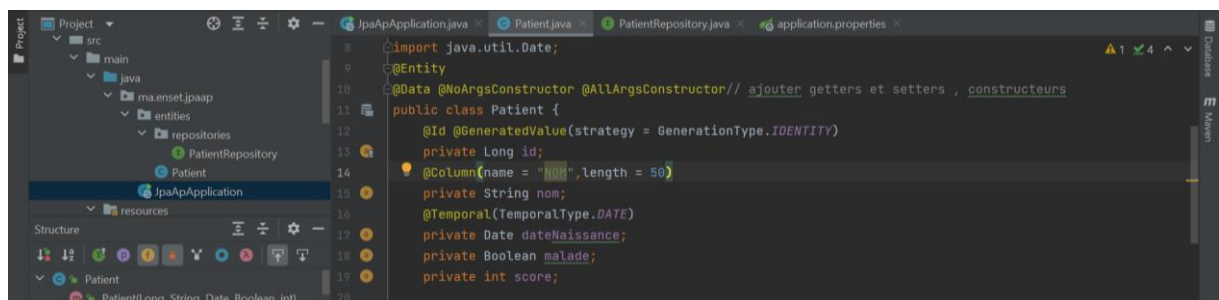
- Se connecter à la base de données



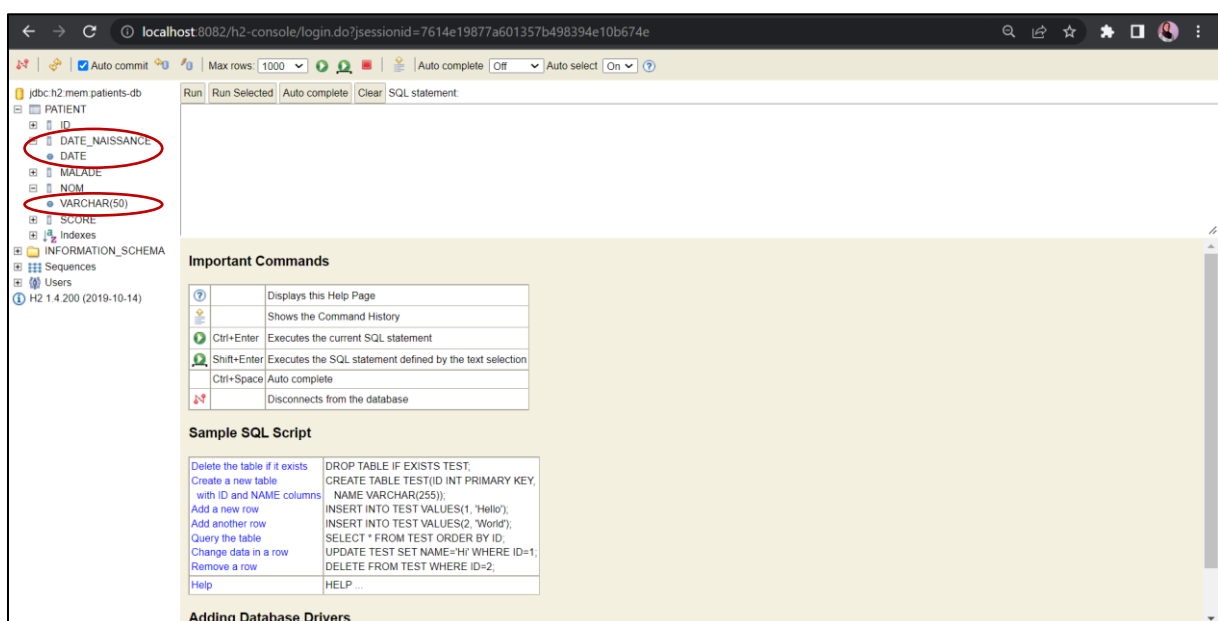
- Consulter tous les patients



Etape 3 : Associer un champ de la colonne à la propriété. + L'annotation `@Temporal` a la valeur de paramètre unique de type `TemporalType`. Il peut s'agir de `DATE`, `TIME` ou `TIMESTAMP`, selon le type SQL sous-jacent que nous voulons utiliser pour le mappage.



Etape 4 : Démarrer l'application Spring



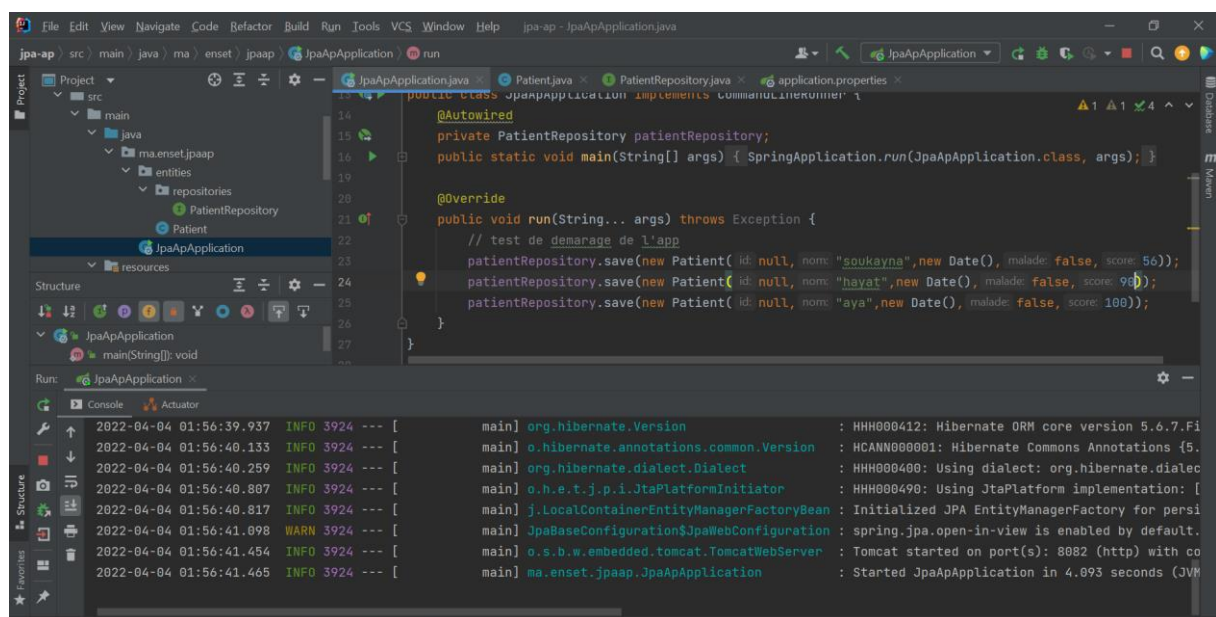
Etape 5 : Gestion des repositories – PatientRepository

```
package ma.enset.jpaap.entities.repositories;

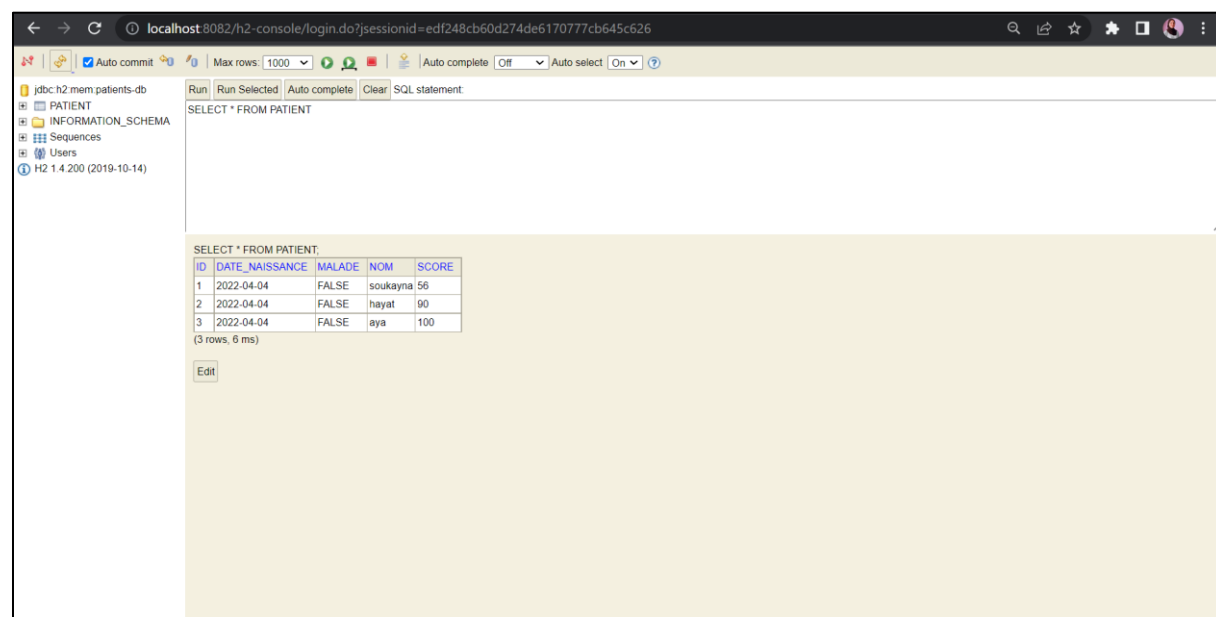
import ...

public interface PatientRepository extends JpaRepository<Patient,Long> {
    //CLASSE, type de id
    public List<Patient> findByMalade(boolean m);
    Page<Patient> findByMalade(boolean m, Pageable pageable);
    // liste des patients dont le score est inferieure a score
    List<Patient> findAllByMaladeAndScoreLessThan(boolean m,int score);
    List<Patient> findAllByMaladeIsTrueAndScoreLessThan(int score);
    List<Patient> findAllByDateNaissanceBetweenAndMaladeIsTrueOrNomLike(Date d1, Date d2,String mc);
    @Query("select p from Patient p where p.nom like :x and p.score<:y")
    List<Patient> chercherPatients(@Param("x") String nom, @Param("y") int scoreMin) ;
}
```

Etape 6 : Création des patients



Etape 7 : Resultats d'execution – consulter tous les patients



Etape 7 : Rechercher sur un patient par son id, afficher ses informations et modifier son score

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help jpa-ap - JpaApApplication.java
jpa-ap src main java ma enset jpaap JpaApApplication run
Project
31 Patient p = patients.get(p.getId());
32 System.out.println("*****");
33 System.out.println(p.getId());
34 System.out.println(p.getNom());
35 System.out.println(p.getScore());
36 System.out.println(p.getDateNaissance());
37 System.out.println(p.getMalade());
38 }
39 System.out.println("*****");
40 Patient patient = patientRepository.findById(1L).orElse( other: null);
41 //Patient patient = patientRepository.findById(1L).orElseThrow(() -> new RuntimeException("Patient not found"));
42 if(patient!=null){
43     System.out.println(patient.getNom());
44     System.out.println(patient.getMalade());
45 }
46 patient.setScore(870);
47 patientRepository.save(patient);
48 }
```

Etape 8 : Résultats des modifications apres demarage de l'application

H2 Console

localhost:8082/h2-console/login.do?jsessionid=a55e4b9cf200a82aeb6640935c2d45c0

jdbc:h2:mem:patients-db

PATIENT

- ID
- DATE_NAISSANCE
- MALADE
- NOM
- SCORE

Indexes

INFORMATION_SCHEMA

Sequences

Users

H2 1.4.200 (2019-10-14)

Run Run Selected Auto complete Clear SQL statement

SELECT * FROM PATIENT

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM	SCORE
1	2022-04-09	FALSE	soukayna	870
2	2022-04-09	FALSE	hayat	90
3	2022-04-09	FALSE	aya	100

(3 rows, 12 ms)

Edit

Etape 9 : Supprimer un patient

```
47 patientRepository.deleteById(1L);
48 }
49 }
50 }
```

Etape 10 : Résultats coté base de données

H2 Console

localhost:8082/h2-console/login.do?jsessionid=7febd5f0aaee6147cbb6c81d92e1ecfb

jdbc:h2:mem:patients-db

PATIENT

- INFORMATION_SCHEMA
- Sequences
- Users

H2 1.4.200 (2019-10-14)

Run Run Selected Auto complete Clear SQL statement

SELECT * FROM PATIENT

SELECT * FROM PATIENT;

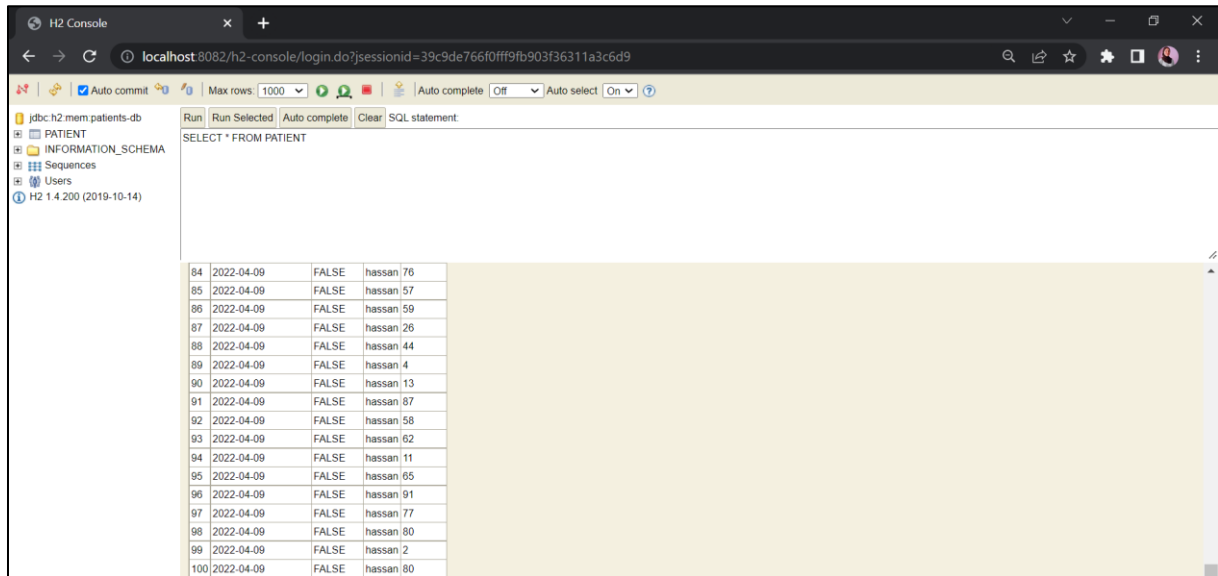
ID	DATE_NAISSANCE	MALADE	NOM	SCORE
2	2022-04-09	FALSE	hayat	90
3	2022-04-09	FALSE	aya	100

(2 rows, 9 ms)

Etape 11 : Insérer 100 patients

```
27
28 // insertion de 100 patients
29 for (int i=0;i<100;i++){
30     patientRepository.save(new Patient( id: null, nom: "hassan", new Date(), malade: false, (int) (Math.random()*100)));
31 }
```

Etape 12 : Résultats



The screenshot shows the H2 Console interface. The left sidebar displays the database structure: jdbc:h2:mem:patients-db, PATIENT, INFORMATION_SCHEMA, Sequences, and Users. The main area shows a SQL statement: `SELECT * FROM PATIENT`. Below the statement, a table of results is displayed with 100 rows. The columns are: Row ID, Date (2022-04-09), Malade (FALSE), and Name (hassan).

Row ID	Date	Malade	Name
84	2022-04-09	FALSE	hassan 76
85	2022-04-09	FALSE	hassan 57
86	2022-04-09	FALSE	hassan 59
87	2022-04-09	FALSE	hassan 26
88	2022-04-09	FALSE	hassan 44
89	2022-04-09	FALSE	hassan 4
90	2022-04-09	FALSE	hassan 13
91	2022-04-09	FALSE	hassan 87
92	2022-04-09	FALSE	hassan 58
93	2022-04-09	FALSE	hassan 62
94	2022-04-09	FALSE	hassan 11
95	2022-04-09	FALSE	hassan 65
96	2022-04-09	FALSE	hassan 91
97	2022-04-09	FALSE	hassan 77
98	2022-04-09	FALSE	hassan 80
99	2022-04-09	FALSE	hassan 2
100	2022-04-09	FALSE	hassan 80

Etape 13 : Pagination

```
37 //pagination
38 Page<Patient> patients = patientRepository.findAll(PageRequest.of( page: 0, size: 5));
39
```

```
=====
Id : 1, Nom : hassan, Score : 74
Date Naissance : 2022-04-09, Malade : false
=====
Id : 2, Nom : hassan, Score : 1
Date Naissance : 2022-04-09, Malade : false
=====
Id : 3, Nom : hassan, Score : 79
Date Naissance : 2022-04-09, Malade : false
=====
Id : 4, Nom : hassan, Score : 30
Date Naissance : 2022-04-09, Malade : false
=====
Id : 5, Nom : hassan, Score : 47
Date Naissance : 2022-04-09, Malade : false
*****
```

```
//pagination
Page<Patient> patients = patientRepository.findAll(PageRequest.of( page: 0, size: 5));
System.out.println("Total pages : "+patients.getTotalPages());
System.out.println("Total elements : "+patients.getTotalElements());
System.out.println("Num page : "+patients.getNumber());
List<Patient> content = patients.getContent();
for (Patient p : content){
    System.out.println("=====");
    System.out.print("Id : "+p.getId());
    System.out.print(", Nom : "+p.getNom());
    System.out.println(", Score : "+p.getScore());
    System.out.print("Date Naissance : "+p.getDateNaissance());
    System.out.println(", Malade : "+p.getMalade());
}
```

```
Run: JpaApplication
Console
Actuator
Total pages : 20
Total elements : 100
Num page : 0
=====
Id : 1, Nom : hassan, Score : 75
Date Naissance : 2022-04-09, Malade : false
=====
Id : 2, Nom : hassan, Score : 8
Date Naissance : 2022-04-09, Malade : false
=====
Id : 3, Nom : hassan, Score : 23
Date Naissance : 2022-04-09, Malade : false
=====
Id : 4, Nom : hassan, Score : 77
Date Naissance : 2022-04-09, Malade : false
=====
Id : 5, Nom : hassan, Score : 39
Date Naissance : 2022-04-09, Malade : false
*****
```

Etape 14 : Rechercher listes des patients qui ne sont pas malade

```
42 List<Patient> content = patients.getContent();
43
44 List<Patient> byMalade = patientRepository.findByMalade(m: false);
45
```

```
Run: JpaApplication
Console
Actuator
Num page : 0
=====
Id : 1, Nom : hassan, Score : 11
Date Naissance : 2022-04-09, Malade : false
=====
Id : 4, Nom : hassan, Score : 2
Date Naissance : 2022-04-09, Malade : false
=====
Id : 5, Nom : hassan, Score : 12
Date Naissance : 2022-04-09, Malade : false
=====
Id : 7, Nom : hassan, Score : 84
Date Naissance : 2022-04-09, Malade : false
=====
Id : 9, Nom : hassan, Score : 12
Date Naissance : 2022-04-09, Malade : false
=====
Id : 10, Nom : hassan, Score : 45
Date Naissance : 2022-04-09, Malade : false
=====
Id : 13, Nom : hassan, Score : 18
Date Naissance : 2022-04-09, Malade : false
```

Etape 15 : Resultats

H2 Console

localhost:8082/h2-console/login.do?sessionId=42a1cdb0ae4aafb55a96944850a0bf3e

Max rows: 1000 | Auto complete: Off | Auto select: On

jdbc:h2:mem:patients-db

PATIENT

INFORMATION_SCHEMA

Sequences

Users

H2 1.4.200 (2019-10-14)

Run | Run Selected | Auto complete | Clear | SQL statement:

SELECT * FROM PATIENT

7	2022-04-09	FALSE	hassan	84
8	2022-04-09	TRUE	hassan	20
9	2022-04-09	FALSE	hassan	12
10	2022-04-09	FALSE	hassan	45
11	2022-04-09	TRUE	hassan	52
12	2022-04-09	TRUE	hassan	75
13	2022-04-09	FALSE	hassan	18
14	2022-04-09	FALSE	hassan	95
15	2022-04-09	TRUE	hassan	77
16	2022-04-09	TRUE	hassan	96
17	2022-04-09	FALSE	hassan	68
18	2022-04-09	TRUE	hassan	89
19	2022-04-09	TRUE	hassan	88
20	2022-04-09	TRUE	hassan	43
21	2022-04-09	FALSE	hassan	47
22	2022-04-09	FALSE	hassan	21
23	2022-04-09	FALSE	hassan	87
24	2022-04-09	TRUE	hassan	9
25	2022-04-09	FALSE	hassan	54
26	2022-04-09	TRUE	hassan	10
27	2022-04-09	FALSE	hassan	93

Etape 16 : Recherche + Pagination

```
public List<Patient> findByMalade(boolean m),
Page<Patient> findByMalade(boolean m, Pageable pageable);

Page<Patient> byMalade = patientRepository.findByMalade(m: true, PageRequest.of(page: 0, size: 4));

=====
Id : 1, Nom : hassan, Score : 75
Date Naissance : 2022-04-09, Malade : true
=====
Id : 2, Nom : hassan, Score : 35
Date Naissance : 2022-04-09, Malade : true
=====
Id : 3, Nom : hassan, Score : 55
Date Naissance : 2022-04-09, Malade : true
=====
Id : 9, Nom : hassan, Score : 56
Date Naissance : 2022-04-09, Malade : true
*****
```

Etape 17 : Annotation @Query

```
List<Patient> findAllByMaladeAndScoreLessThan(boolean m, int score);
List<Patient> findAllByMaladeIsTrueAndScoreLessThan(int score);
List<Patient> findAllByDateNaissanceBetweenAndMaladeIsTrueOrNomLike(Date d1, Date d2, String mc);
@Query("select p from Patient p where p.nom like :x and p.score<:y")
List<Patient> chercherPatients(@Param("x") String nom, @Param("y") int scoreMin) ;

}

List<Patient> patientsList = patientRepository.chercherPatients(nom: "%h%", scoreMin: 40);

for (Patient p : patientsList){
    System.out.println("=====");
    System.out.print("Id : "+p.getId());
    System.out.print(", Nom : "+p.getNom());
    System.out.print(", Score : "+p.getScore());
    System.out.print("Date Naissance : "+p.getDateNaissance());
    System.out.print(", Malade : "+p.getMalade());
}
```

Run: JpaApApplication

Console

```
↑ Id : 41, Nom : hassan, Score : 39
Date Naissance : 2022-04-10, Malade : true
=====
↓ Id : 45, Nom : hassan, Score : 24
Date Naissance : 2022-04-10, Malade : true
=====
Id : 48, Nom : hassan, Score : 35
Date Naissance : 2022-04-10, Malade : true
=====
Id : 49, Nom : hassan, Score : 34
Date Naissance : 2022-04-10, Malade : true
=====
Id : 55, Nom : hassan, Score : 4
Date Naissance : 2022-04-10, Malade : false
=====
Id : 58, Nom : hassan, Score : 31
Date Naissance : 2022-04-10, Malade : true
```

Etape 18 : Basculement vers la base de données MySQL

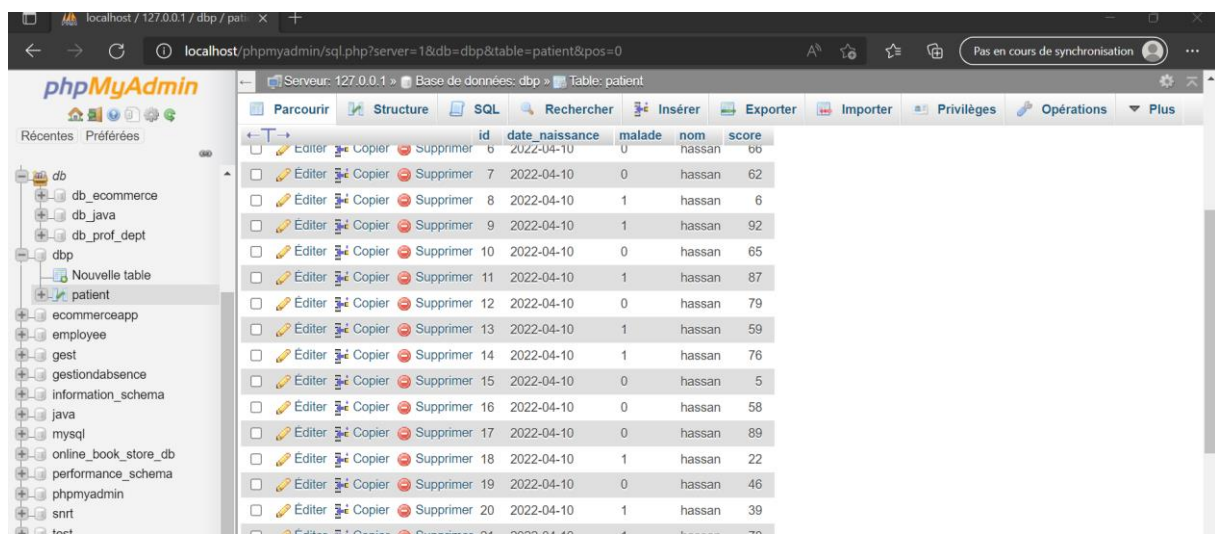
- Ajouter les dépendances

```
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
</dependency>
```


- application.properties

```
1 #spring.datasource.url=jdbc:h2:mem:patients-db
2 spring.datasource.url=jdbc:mysql://localhost:3306/dbp?createDatabaseIfNotExist=true
3 spring.datasource.username=root
4 spring.datasource.password=
5 #spring.h2.console.enabled=true
6 server.port=8082
7 spring.jpa.hibernate.ddl-auto=update
8 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
9 spring.jpa.show-sql=true
```

- Résultats du démarrage de l'application spring



localhost / 127.0.0.1 / dbp / patient

localhost/phpmyadmin/sql.php?server=1&db=dbp&table=patient&pos=0

Pas en cours de synchronisation

Seigneur: 127.0.0.1 » Base de données: dbp » Table: patient

	id	date_naissance	malade	nom	score
<input type="checkbox"/>	6	2022-04-10	0	hassan	66
<input type="checkbox"/>	7	2022-04-10	0	hassan	62
<input type="checkbox"/>	8	2022-04-10	1	hassan	6
<input type="checkbox"/>	9	2022-04-10	1	hassan	92
<input type="checkbox"/>	10	2022-04-10	0	hassan	65
<input type="checkbox"/>	11	2022-04-10	1	hassan	87
<input type="checkbox"/>	12	2022-04-10	0	hassan	79
<input type="checkbox"/>	13	2022-04-10	1	hassan	59
<input type="checkbox"/>	14	2022-04-10	1	hassan	76
<input type="checkbox"/>	15	2022-04-10	0	hassan	5
<input type="checkbox"/>	16	2022-04-10	0	hassan	58
<input type="checkbox"/>	17	2022-04-10	0	hassan	89
<input type="checkbox"/>	18	2022-04-10	1	hassan	22
<input type="checkbox"/>	19	2022-04-10	0	hassan	46
<input type="checkbox"/>	20	2022-04-10	1	hassan	39
<input type="checkbox"/>	21	2022-04-10	1	hassan	70