

**DEPARTEMENT MATHÉMATIQUES 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. YOUSSEFI 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 Persistence API est une spécification créée 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 :

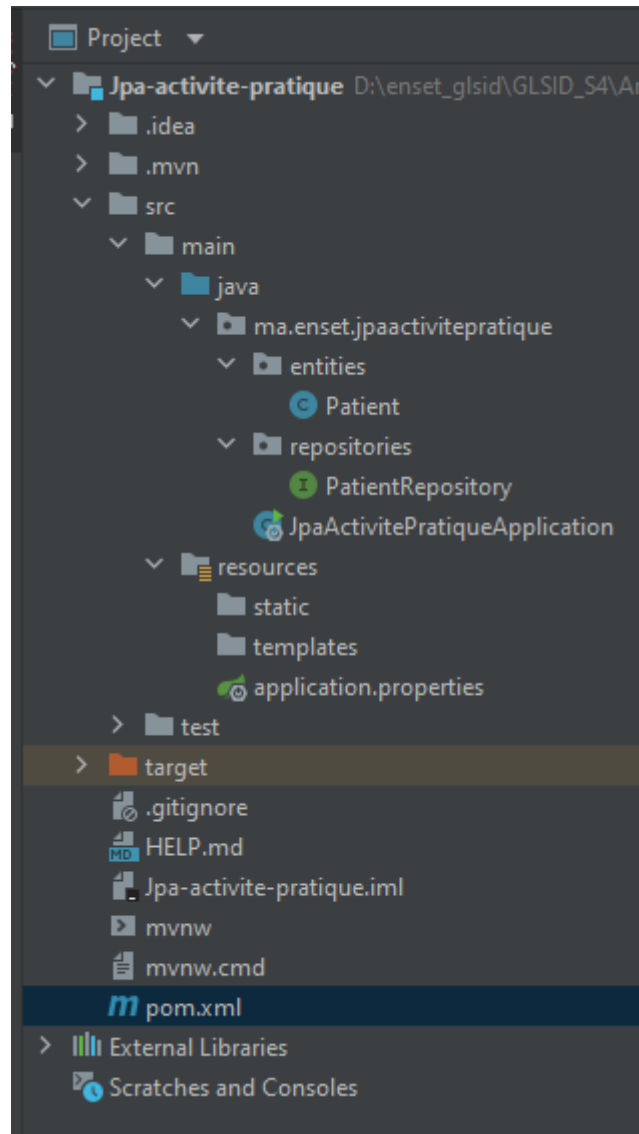
- Les fichiers XML de mapping
- Les annotations JPA

Java Bean : grain de café

Et pour ce faire nous nous avons basé sur le Spring Boot qui est une version de Spring framework, créée 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?sessionId=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 :**

```

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

```

```

:: Spring Boot ::
2022-03-04 23:15:27.214 INFO 8180 --- [main] m.e.j.JpaActivePratiqueApplication : Starting JpaActivePrat
2022-03-04 23:15:27.222 INFO 8180 --- [main] m.e.j.JpaActivePratiqueApplication : No active profile set, f
2022-03-04 23:15:28.766 INFO 8180 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Dat
2022-03-04 23:15:28.862 INFO 8180 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data rep
2022-03-04 23:15:29.582 INFO 8180 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with
2022-03-04 23:15:29.598 INFO 8180 --- [main] o.apache.catalina.core.StandardService : Recommended plugin available for dependency
2022-03-04 23:15:29.598 INFO 8180 --- [main] org.apache.catalina.core.StandardEngine : Do not suggest this plugin
2022-03-04 23:15:29.790 INFO 8180 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] :

```

La base de données sous H2:

jdbc:h2:mem:patient-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:

**Important Commands**

?	Displays this Help Page
↑	Shows the Command History
Ctrl+Enter	Executes the current SQL statement
Shift+Enter	Executes the SQL statement defined by the text selection
Ctrl+Space	Auto complete
Ctrl+D	Disconnects from the database

**Sample SQL Script**

Delete the table if it exists	DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row	INSERT INTO TEST VALUES(1, 'Hello');
Add another row	INSERT INTO TEST VALUES(2, 'World');
Query the table	SELECT * FROM TEST ORDER BY ID;
Change data in a row	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row	DELETE FROM TEST WHERE ID=2;
Help	HELP ...

**Adding Database Drivers**

The screenshot shows a web-based database console interface. The URL bar indicates the connection to a local database at localhost:8088/h2-console/login.do. The interface includes a sidebar with a database tree showing 'jdbc:h2:mem:patient-db' and its schema 'INFORMATION\_SCHEMA'. The main area displays a SQL query 'SELECT \* FROM PATIENT' and its results in a table format. The results show three rows of patient data.

ID	DATE_NAISSANCE	MALADE	NOM	SCORE
1	2022-03-04	FALSE	khaoula	40
2	2022-03-04	TRUE	nadir	67
3	2022-03-04	FALSE	khaoula	100

(3 rows, 40 ms)

## Récupération des données :

The screenshot shows an IDE with a Java project. The code defines a `PatientRepository` interface and a `JpaActivitePratiqueApplication` class that implements it. The application uses JPA to interact with a database. The console output shows the application running successfully and displaying the patient data.

```

@Override
public void run(String... args) throws Exception {
    patientRepository.save(new Patient(id: null, nom: "khaoula", new Date(), malade: false, score: 40));
    patientRepository.save(new Patient(id: null, nom: "nadir", new Date(), malade: true, score: 67));
    patientRepository.save(new Patient(id: null, nom: "khaoula", new Date(), malade: false, score: 100));

    List<Patient> patients = patientRepository.findAll();
    patients.forEach(p -> {
        System.out.println("id: " + p.getId() + " nom: " + p.getNom() + " malade: " + p.isMalade() + " score: " + p.getScore());
    });
}

```

```

2022-03-04 23:35:34.311 INFO 8464 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect
2022-03-04 23:35:35.476 INFO 8464 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform
2022-03-04 23:35:35.501 INFO 8464 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory
2022-03-04 23:35:36.126 WARN 8464 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. This can lead to stale data being visible while your application is running. Read https://hibernate.org/blog/2019/05/28/hibernate-jpa-2-1-0-configuration.html for more details.
2022-03-04 23:35:36.941 INFO 8464 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s) 8080
2022-03-04 23:35:36.961 INFO 8464 --- [main] m.e.j.JpaActivitePratiqueApplication : Started JpaActivitePratiqueApplication

```

id: 1 nom : khaoula malade : false score: 40  
id: 2 nom : nadir malade : true score: 67  
id: 3 nom : khaoula malade : false score: 100

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 atteignable grace à Spring Data :

```
package ma.enset.jpaaactivitepratique.repositories;  
  
import ma.enset.jpaaactivitepratique.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 PHPMysqlAdmin :



←T→		id	date_naissance	malade	nom	score
<input type="checkbox"/>	Edit  Copy  Delete	1	2022-03-05	1	khaoula	1928
<input type="checkbox"/>	Edit  Copy  Delete	2	2022-03-05	1	khaoula	11
<input type="checkbox"/>	Edit  Copy  Delete	3	2022-03-05	1	khaoula	59
<input type="checkbox"/>	Edit  Copy  Delete	4	2022-03-05	1	khaoula	44
<input type="checkbox"/>	Edit  Copy  Delete	5	2022-03-05	1	khaoula	33
<input type="checkbox"/>	Edit  Copy  Delete	6	2022-03-05	1	khaoula	22
<input type="checkbox"/>	Edit  Copy  Delete	7	2022-03-05	1	khaoula	27
<input type="checkbox"/>	Edit  Copy  Delete	8	2022-03-05	0	khaoula	69
<input type="checkbox"/>	Edit  Copy  Delete	9	2022-03-05	1	khaoula	81
<input type="checkbox"/>	Edit  Copy  Delete	10	2022-03-05	0	khaoula	89
<input type="checkbox"/>	Edit  Copy  Delete	11	2022-03-05	0	khaoula	38
<input type="checkbox"/>	Edit  Copy  Delete	12	2022-03-05	1	khaoula	21
<input type="checkbox"/>	Edit  Copy  Delete	13	2022-03-05	1	khaoula	67
<input type="checkbox"/>	Edit  Copy  Delete	14	2022-03-05	0	khaoula	89
<input type="checkbox"/>	Edit  Copy  Delete	15	2022-03-05	0	khaoula	33
<input type="checkbox"/>	Edit  Copy  Delete	16	2022-03-05	1	khaoula	63
<input type="checkbox"/>	Edit  Copy  Delete	17	2022-03-05	1	khaoula	75
<input type="checkbox"/>	Edit  Copy  Delete	18	2022-03-05	0	khaoula	10

## Conclusion

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