

Rapport TP3 :

Activité Pratique N°3 : JPA Hibernate Spring Data



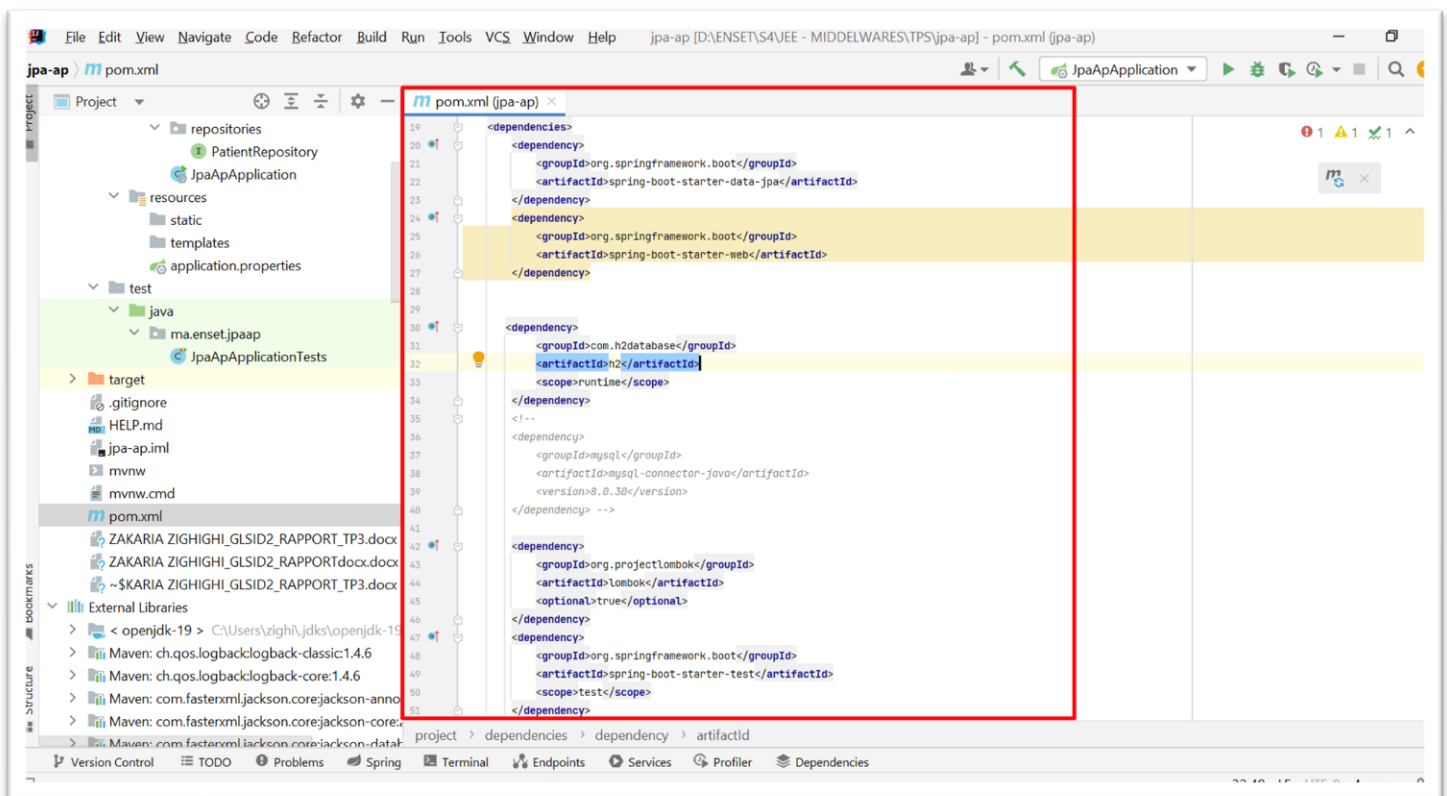
Réalisé par :
ZAKARIA ZIGHIGHI
Encadré par :
Pr. Mohammed Youssfi

Rapport TP3 :

LIEN REPO GITHUB :

// 1- INSATLLER INTELLIJ

2. Créer un projet Spring Initializer avec les dépendances JPA, H2, Spring Web et Lombok



3. Créer l'entité JPA Patient ayant les attributs :

package ma.enset.jpap.entities;

```
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
import java.util.Date;
```

```
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Patient {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(length = 50)
    private String nom;
    @Temporal(TemporalType.DATE)
```



```
private Date dateNaissance;  
private boolean malade;  
private int score;  
}
```

4. Configurer l'unité de persistance dans le fichier application.properties

```
spring.datasource.url=jdbc:h2:mem:jpcap  
spring.h2.console.enabled=true  
server.port=8089
```

5. Créer l'interface JPA Repository basée sur Spring data

```
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface PatientRepository extends JpaRepository<Patient, Long> {  
  
}
```

6. Tester quelques opérations de gestion de patients :

- Ajouter des patients :

```
package ma.enset.jpap;  
  
import ma.enset.jpap.entities.Patient;  
import ma.enset.jpap.repositories.PatientRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.PageRequest;  
  
import java.util.Date;  
import java.util.List;  
  
@SpringBootApplication  
public class JpaApApplication implements CommandLineRunner {  
    @Autowired  
    private PatientRepository patientRepository;  
  
    public static void main(String[] args) {  
        SpringApplication.run(JpaApApplication.class, args);  
    }  
  
    @Override  
    public void run(String... args) throws Exception {  
        patientRepository.save(new Patient(null, "Amine", new Date(), true, 100));  
        patientRepository.save(new Patient(null, "SAID", new Date(), true, 168));  
        patientRepository.save(new Patient(null, "MOHAMMED", new Date(), true, 70));  
    }  
}
```

The screenshot shows the H2 Console web interface in a browser. The address bar shows the URL: `localhost:8089/h2-console/login.do?jsessionid=cedb89492a5c7ca00b179b7879d...`. The interface includes a sidebar with a tree view showing the database structure: `jdbc:h2:mem:jpcap`, `PATIENT`, `INFORMATION_SCHEMA`, `Users`, and `H2 2.1.214 (2022-06-13)`. The main area displays the SQL statement `SELECT * FROM PATIENT` and its results. The results are shown in a table with 5 columns: `ID`, `DATE_NAISSANCE`, `MALADE`, `NOM`, and `SCORE`. The table contains 3 rows of data. Below the table, it indicates `(3 rows, 3 ms)` and an `Edit` button.

| ID | DATE_NAISSANCE | MALADE | NOM | SCORE |
|----|----------------|--------|----------|-------|
| 1 | 2023-05-20 | TRUE | Amine | 100 |
| 2 | 2023-05-20 | TRUE | SAID | 168 |
| 3 | 2023-05-20 | TRUE | MOHAMMED | 70 |

- Consulter tous les patients

```
List<Patient> patients = patientRepository.findAll();
patients.forEach(p->{

    System.out.println("-----");
    System.out.println(p.getId());
    System.out.println(p.getNom());
    System.out.println(p.getDateNaissance());
    System.out.println(p.getScore());
    System.out.println(p.isMalade());

});
```

The screenshot shows the Spring Boot Actuator console output. The logs include various startup messages such as `HikariPool-1 - Start completed.`, `H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:jpcap'.`, `Processing PersistenceUnitInfo [name: default]`, `org.hibernate.Version`, `Using dialect: org.hibernate.dialect.H2Dialect`, `Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]`, `Initialized JPA EntityManagerFactory for persistence unit 'default'`, `spring.jpa.open-in-view is enabled by default. Therefore, database query may be slower than expected. To disable this warning, explicitly configure spring.jpa.open-in-view to false.`, `Tomcat started on port(s): 8089 (http) with context path ''`, and `Started JpaApplication in 2.542 seconds (process running for 3.005)`. A red box highlights the output of the `patients.forEach` loop, showing the patient data in a structured format:

```
1
Amine
2023-05-20
100
true
-----
2
SAID
2023-05-20
168
true
-----
3
MOHAMMED
2023-05-20
70
true
```



- Consulter un patient by id

```
System.out.println("*****");
Patient patient=patientRepository.findById(1L).orElse(null);
System.out.println(patient);
```

```
true
*****
Patient(id=1, nom=Amine, dateNaissance=2023-05-20, malade=true, score=100)
```

- - Mettre à jour un patient

```
System.out.println("apres mise a jour");
patient.setScore(199);
patientRepository.save(patient);
System.out.println(patient);
```

```
3
MOHAMMED
2023-05-20
70
true
*****
Patient(id=1, nom=Amine, dateNaissance=2023-05-20, malade=true, score=100)
apres mise a jour
Patient(id=1, nom=Amine, dateNaissance=2023-05-20, malade=true, score=199)
```

- supprimer un patient

```
patientRepository.deleteById(1L);
```

```
Patient(id=1, nom=Amine, dateNaissance=2023-05-20, malade=true, score=199)
supprimer patient avec id 1
-----
2
SAID
2023-05-20
168
true
-----
3
MOHAMMED
2023-05-20
70
true
```

7. Migrer de H2 Database vers MySQL

```
spring.datasource.url=jdbc:mysql://localhost:3306/TP3DB?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto= create
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MariaDBDialect
server.port=8089
spring.jpa.show-sql=true
```

Et dans pom.xml on ajoute la dépendance

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.30</version>
</dependency>
```

localhost/phpmyadmin/index.php?route=/sql&db=tp3db&table=patient&pos=0

réception (...), Untitled | Visual Par..., Free MongoDB Offi..., WEB - admin.123.m..., bdd, Sage - Poe, Document sans titr..., Oracle Quiz Questi...

Serveur : 127.0.0.1 » Base de données : tp3db » Table : patient

Parcourir, Structure, SQL, Rechercher, Insérer, Exporter, Importer, Privilèges, Opérations, Suivre

✓ Affichage des lignes 0 - 1 (total de 2, traitement en 0,0005 seconde(s).)

SELECT * FROM `patient`

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

Options supplémentaires

| | | | id | date_naissance | malade | nom | score |
|--------------------------|--------|--------|----|----------------|--------|----------|-------|
| <input type="checkbox"/> | Éditer | Copier | 2 | 2023-05-20 | 1 | SAID | 168 |
| <input type="checkbox"/> | Éditer | Copier | 3 | 2023-05-20 | 1 | MOHAMMED | 70 |

↑ ☐ Tout cocher Avec la sélection : Éditer, Copier, Supprimer, Exporter

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

8. Reprendre les exemples du Patient, Médecin, rendez-vous, consultation, users et roles de la vidéo

vidéo 1 : lien repo github :

Vidéo 2 : lien repo github :