

Architecture JEE et Middlewares

Compte-rendu

Activité Pratique N3

JPA Hibernate Spring Data

Nom et Prénom : BOUSSAIRI Ikram

E-mail : ikramboussairi12@gmail.com

ENSET GLSID – S4

Professeur : Mohamed YOUSSEFI



Java EE™

Sommaire

Table des matières

Avant-propos	3
Mise en contexte	4
Application	5
1. Créer un projet Spring Initializer avec les dépendances JPA, H2, Spring Web et Lombok.....	5
2. Créer l'entité JPA Patient ayant les attributs	5
3. Configurer l'unité de persistance dans le fichier application.properties	6
4. Créer l'interface JPA Repository basée sur Spring data	6
5. Tester quelques opérations de gestion de patients :	6

Avant-propos

Il est difficile de développer un système logiciel qui respecte l'ensemble des exigences fonctionnelles, satisfaire les besoins métiers de projet, ainsi que les exigences techniques à savoir : la performance ; le temps de réponse de notre système, l'équilibrage de charges et tolérance aux pannes, le Problème de montée en charge. La maintenance, c'est-à-dire notre application doit être facile à maintenir et pour se faire l'application doit évoluer dans le temps ainsi qu'il doit être fermée à la modification et ouverte à l'extension. Sans oublier la sécurité et la persistance des données dans des SGBD appropriés, etc.

En effet, afin de pouvoir relever se défi il va falloir utiliser l'expérience des autres ; autrement dit bâtir notre application sur une architecture d'entreprise. L'inversion de contrôle alors prend en charge du flot d'exécution de notre logiciel, c'est-à-dire le flot d'exécution d'un logiciel n'est plus sous le contrôle direct de l'application elle-même mais du Framework. D'autre part, le développeur s'occupe uniquement du code métier (exigences fonctionnelles).

Dans ce contexte, J'ai eu l'occasion d'appliquer l'ensemble de concepts de bases de l'inversion de contrôle et l'injection de dépendance à travers ces deux activités pratique.

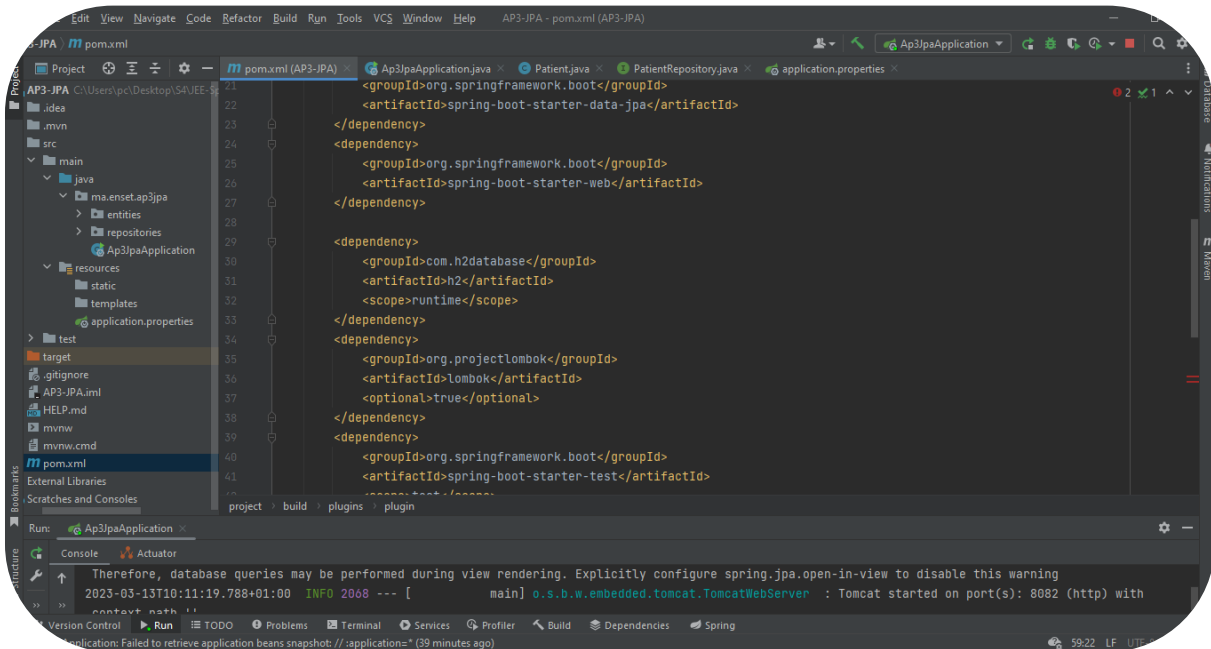
Ce compte rendu se compose de deux parties, la première partie consiste sur la création des différentes classes et interfaces en utilisant le couplage faible et en utilisant l'injection des dépendances de différentes manières ; instanciation statique, instanciation dynamique et en utilisant le Framework Spring (Version XML et version annotations). La deuxième partie consiste sur la réalisation d'un Mini Projet ; un Framework de l'injection des dépendances.

Mise en contexte

1. Créer un projet Spring Initializer avec les dépendances JPA, H2, Spring Web et Lombok
2. Créer l'entité JPA Patient ayant les attributs :
 - id de type Long
 - nom de type String
 - dateNaissance de type Date
 - malade de type boolean
 - score de type int
3. Configurer l'unité de persistance dans le fichier application.properties
4. Créer l'interface JPA Repository basée sur Spring data
5. . 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

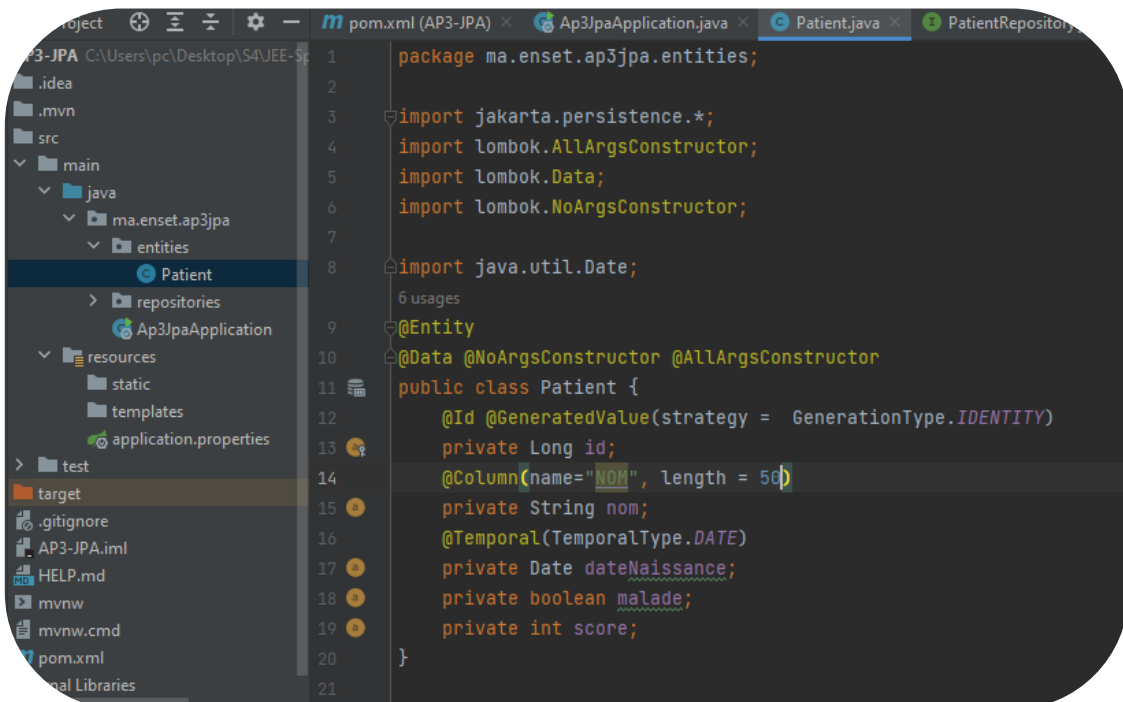
Application

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

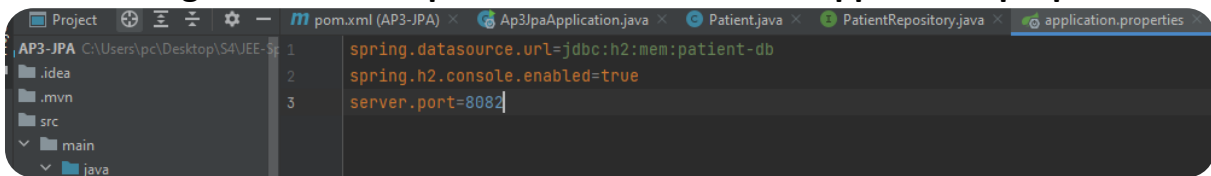


2. Créer l'entité JPA Patient ayant les attributs

- id de type Long
- nom de type String
- dateNaissance de type Date
- malade de type boolean
- score de type int



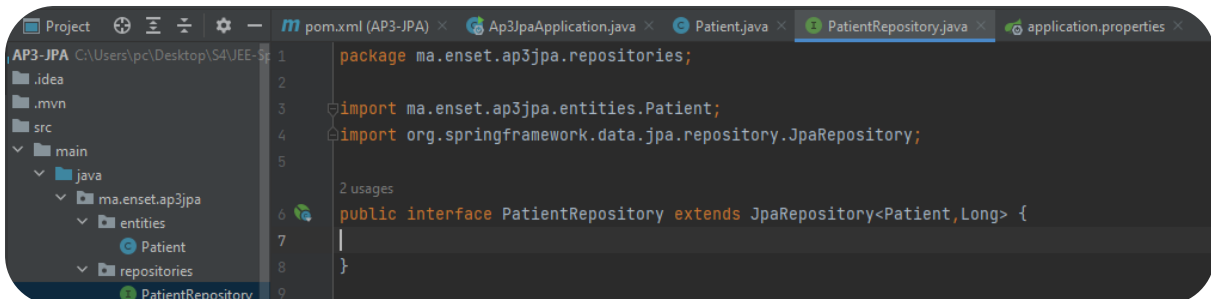
3. Configurer l'unité de persistance dans le fichier application.properties



The screenshot shows the 'application.properties' file in an IDE. The configuration includes the Spring Data JPA datasource URL, console logging, and the server port.

```
spring.datasource.url=jdbc:h2:mem:patient-db
spring.h2.console.enabled=true
server.port=8082
```

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



The screenshot shows the 'PatientRepository.java' file in an IDE. It defines a package, imports the necessary classes, and declares the 'PatientRepository' interface extending 'JpaRepository'.

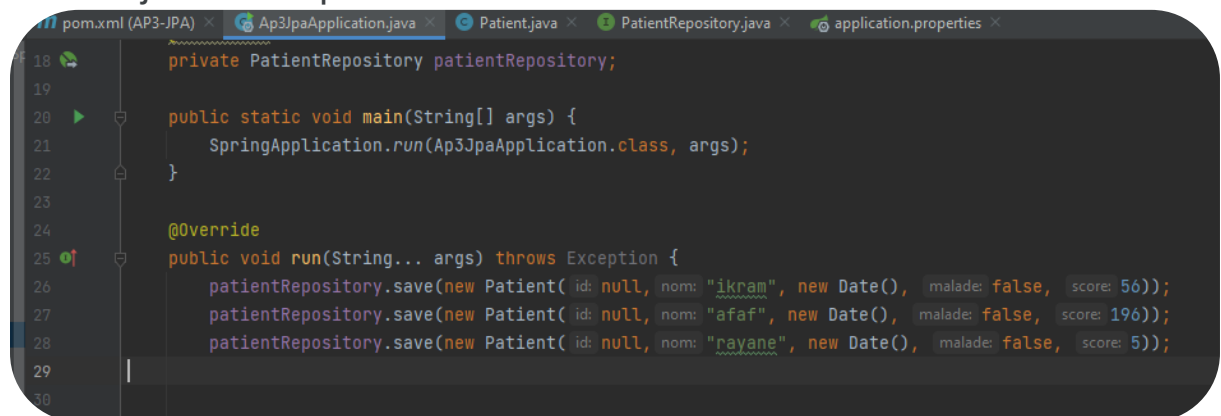
```
package ma.enset.ap3jpa.repositories;

import ma.enset.ap3jpa.entities.Patient;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PatientRepository extends JpaRepository<Patient, Long> {
}
```

5. Tester quelques opérations de gestion de patients :

- Ajouter des patients



The screenshot shows the 'Ap3JpaApplication.java' file in an IDE. The 'main' method calls 'SpringApplication.run'. The 'run' method is annotated with '@Override' and contains three 'save' calls to the 'PatientRepository'.

```
private PatientRepository patientRepository;

public static void main(String[] args) {
    SpringApplication.run(Ap3JpaApplication.class, args);
}

@Override
public void run(String... args) throws Exception {
    patientRepository.save(new Patient(id: null, nom: "ikram", new Date(), malade: false, score: 56));
    patientRepository.save(new Patient(id: null, nom: "afaf", new Date(), malade: false, score: 196));
    patientRepository.save(new Patient(id: null, nom: "rayane", new Date(), malade: false, score: 5));
}
```

jdbc:h2:mem:patient-db
PATIENT
INFORMATION_SCHEMA
Users
H2 2.1.214 (2022-06-13)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM PATIENT

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM	SCORE
1	2023-03-13	FALSE	ikram	56
2	2023-03-13	FALSE	afaf	196
3	2023-03-13	FALSE	rayane	5

(3 rows, 13 ms)

Edit

- Consulter tous les patients

```
pom.xml (AP3-JPA) x Ap3JpaApplication.java x Patient.java x PatientRepository.java x application.properties x
@Override
25 public void run(String... args) throws Exception {
26     patientRepository.save(new Patient(id: null, nom: "ikram", new Date(), malade: false, score: 56));
27     patientRepository.save(new Patient(id: null, nom: "afaf", new Date(), malade: false, score: 196));
28     patientRepository.save(new Patient(id: null, nom: "rayane", new Date(), malade: false, score: 5));
29
30
31     List<Patient> patients= patientRepository.findAll();
32     patients.forEach( patient -> {
33         System.out.println("-----");
34         System.out.println(patient.getId());
35         System.out.println(patient.getNom());
36         System.out.println(patient.getDateNaissance());
37         System.out.println(patient.getScore());
38         System.out.println(patient.isMalade());
39     });
40 }
```

```
-----
1
ikram
2023-03-13
56
false
-----
2
afaf
2023-03-13
196
false
-----
3
rayane
2023-03-13
5
false
```

- Consulter un patient

```
pom.xml (AP3-JPA) x Ap3JpaApplication.java x Patient.java x PatientRepository.java x application.properties x
37     System.out.println(patient.getScore());
38     System.out.println(patient.isMalade());
39 }
40
41 Patient patient = patientRepository.findById(1L).get();
42 if(patient!=null){
43     System.out.println("-----");
44     System.out.println(patient.getNom());
45     System.out.println(patient.getDateNaissance());
46     System.out.println(patient.getScore());
47     System.out.println(patient.isMalade());
48 }
49
50 }
```

- Chercher des patients

```
-----  
ikram  
2023-03-13  
56  
false
```

- Mettre à jour un patient

```
Patient patient = patientRepository.findById(1L).get();  
patient.setScore(1000);  
patientRepository.save(patient);
```

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure: jdbc:h2:mem:patient-db, PATIENT, INFORMATION_SCHEMA, Users, and H2 2.1.214 (2022-06-13). The main area shows a SQL statement: `SELECT * FROM PATIENT`. Below the statement, the results are displayed as 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 there is an "Edit" button.

ID	DATE_NAISSANCE	MALADE	NOM	SCORE
1	2023-03-13	FALSE	ikram	1000
2	2023-03-13	FALSE	afaf	196
3	2023-03-13	FALSE	rayane	5

(3 rows, 3 ms)

Edit

- supprimer un patient

```
patientRepository.deleteById(2L);
```


Auto

Max

1000

Auto complet

jdbc:h2:mem:patient-db

PATIENT

INFORMATION_SCHEMA

Users

H2 2.1.214 (2022-06-13)

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT * FROM PATIENT|

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM	SCORE
1	2023-03-13	FALSE	ikram	56
3	2023-03-13	FALSE	rayane	5

(2 rows, 20 ms)

Edit