

Matière : JEE

Rapport de Travaux Pratique

Lien Git du Code : <https://github.com/Salah2210/JeeTps>

Plan

Introduction

Réalisation

Test

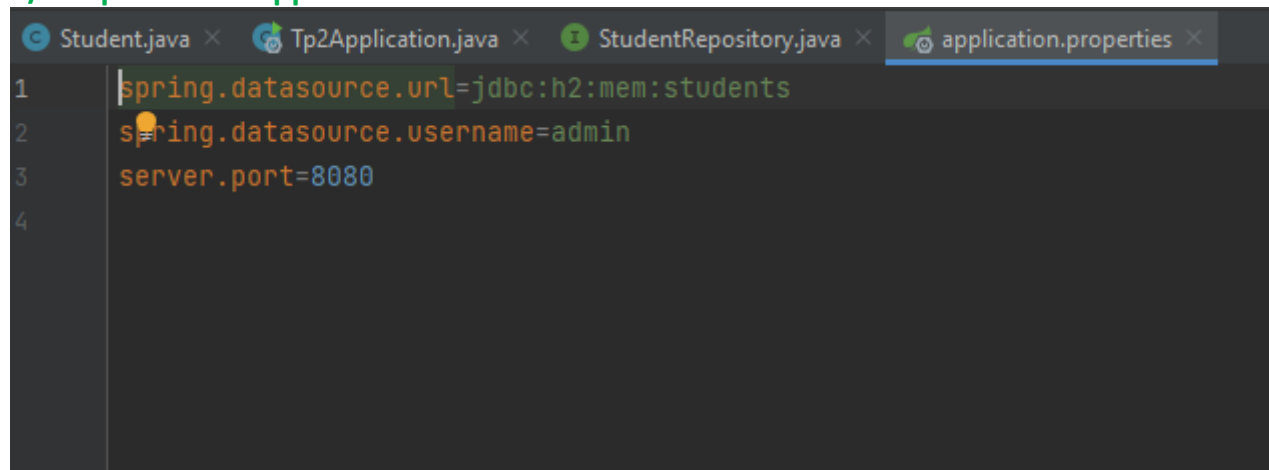
Conclusion

Introduction

La JPA est une API Java qui permettant aux développeurs de manipuler des objets Java sans avoir à écrire de code SQL pour effectuer des opérations de base de données tel que la création d'une table de base de données à partir d'une class modèle ou l'application des différentes transactions INSERT, UPDATE, DELETE grâce à des méthodes prédéfinis fournis par l'interface JpaRepository ou par des méthode qu'on va créer dans une interface qui héritera de l'interface JPA.

Réalisation

1/ Propriété de l'application



```
1 spring.datasource.url=jdbc:h2:mem:students
2 spring.datasource.username=admin
3 server.port=8080
4
```

2/ Class Student

```
Student.java x Tp2Application.java x StudentRepository.java x application.properties x
5 import lombok.Data;
6 import lombok.NoArgsConstructor;
7 import org.hibernate.annotations.CreationTimestamp;
8 import java.util.Date;
9 @Entity @Table(name = "EMSI_STUDENTS")
10 @Data @AllArgsConstructor @NoArgsConstructor
11
12 public class Student {
13     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Integer id;
15     @Column(name = "REGISTRATION_N", unique = true)
16     private String registrationNumber;
17     @Column(name = "NAME", length = 30, nullable = false)
18     private String fullName;
19     @Temporal(TemporalType.DATE)
20     private Date birthday;
21     private Boolean stillActive;
22     @Temporal(TemporalType.TIMESTAMP)@CreationTimestamp
23     private Date lastConnection;
24 }
```

Cette classe permet de créer le modèle de notre table de base de données en spécifiant les colonnes ainsi que la clé primaire.

3/ Interface studentRepository

```
Student.java × Tp2Application.java × StudentRepository.java × application.properties ×
1 package com.example.tp2.main.repositories;
2
3 import com.example.tp2.main.entities.Student;
4 import jakarta.transaction.Transactional;
5 import org.springframework.data.jpa.repository.JpaRepository;
6
7 import java.util.Date;
8 import java.util.List;
9
10 @Transactional
11 public interface StudentRepository extends JpaRepository<Student, Integer> {
12     List<Student> findByRegistrationNumber(String registrationNumber);
13     List<Student> findByStillActive(Boolean stillActive);
14     List<Student> findByBirthday(Date birthday);
15     List<Student> findByFullName(String fullName);
16     List<Student> findByLastConnection(Date lastConnection);
17
18     List<Student> deleteByFullName(String ls);
19     long deleteByStillActive(Boolean stillActive);
20
21     List<Student> findByFullNameAndStillActiveIsTrue(String fullName);
22     List<Student> findByRegistrationNumberLike(String registrationNumber);
23     List<Student> findByRegistrationNumberLikeAndFullNameLike(String registrationNumber, String fullName);
24     List<Student> findByIdIsLessThan(Integer id);
25     List<Student> findByIdIsLessThanEqual(Integer id);
26     List<Student> deleteByFullNameAndStillActiveIsTrue(String fullName);
27 }
28
```

L'interface studentRepository hérite des méthodes de l'interface JpaRepository qui prend en considération la classe et sa clé primaire tout en ayant la possibilité d'ajouter d'autres méthodes.

4/ Class Main

```

1  package com.example.tp2;
2
3  import ...
12
13  @SpringBootApplication
14  public class Tp2Application implements CommandLineRunner {
15
16      @Autowired
17      private StudentRepository studentRepository;
18
19      public static void main(String[] args) { SpringApplication.run(Tp2Application.class, args); }
20
21      @Override
22      public void run(String... args) throws Exception {
23          System.out.println("*****Insertion *****");
24
25          studentRepository.save(
26              new Student( id: null, registrationNumber: "A1", fullName: "Amine", new Date(), stillActive: true, lastConnection: null));
27          studentRepository.save(
28              new Student( id: null, registrationNumber: "A2", fullName: "Ilyas", new Date(), stillActive: true, lastConnection: null));
29          studentRepository.save(
30              new Student( id: null, registrationNumber: "A3", fullName: "Saad", new Date(), stillActive: false, lastConnection: null));
31          studentRepository.save(
32              new Student( id: null, registrationNumber: "A4", fullName: "Arij", new Date(), stillActive: true, lastConnection: null));
33
34          studentRepository.save(
35              new Student( id: null, registrationNumber: "A5", fullName: "Lina", new Date(), stillActive: false, lastConnection: null));
36
37          System.out.println("***Inserted rows ***");
38          System.out.println("Count: "+studentRepository.count());
39
40          System.out.println("***Display rows**");
41          List<Student> students = studentRepository.findAll();
42          students.forEach(student -> {System.out.println(student.toString());});
43
44          System.out.println("***Get Element By ID ***");
45          Student student = studentRepository.findById(3).orElse( other: null);
46          System.out.println(student.toString());
47
48          System.out.println("*** Update an Element *****");
49          student.setRegistrationNumber("S8");
50          studentRepository.save(student);
51
52          System.out.println("***Delete ana Element *****");
53          studentRepository.delete(student);
54          System.out.println("Count: " +studentRepository.count());
55
56          studentRepository.deleteById(5);
57          System.out.println("Count: "+studentRepository.count());

```

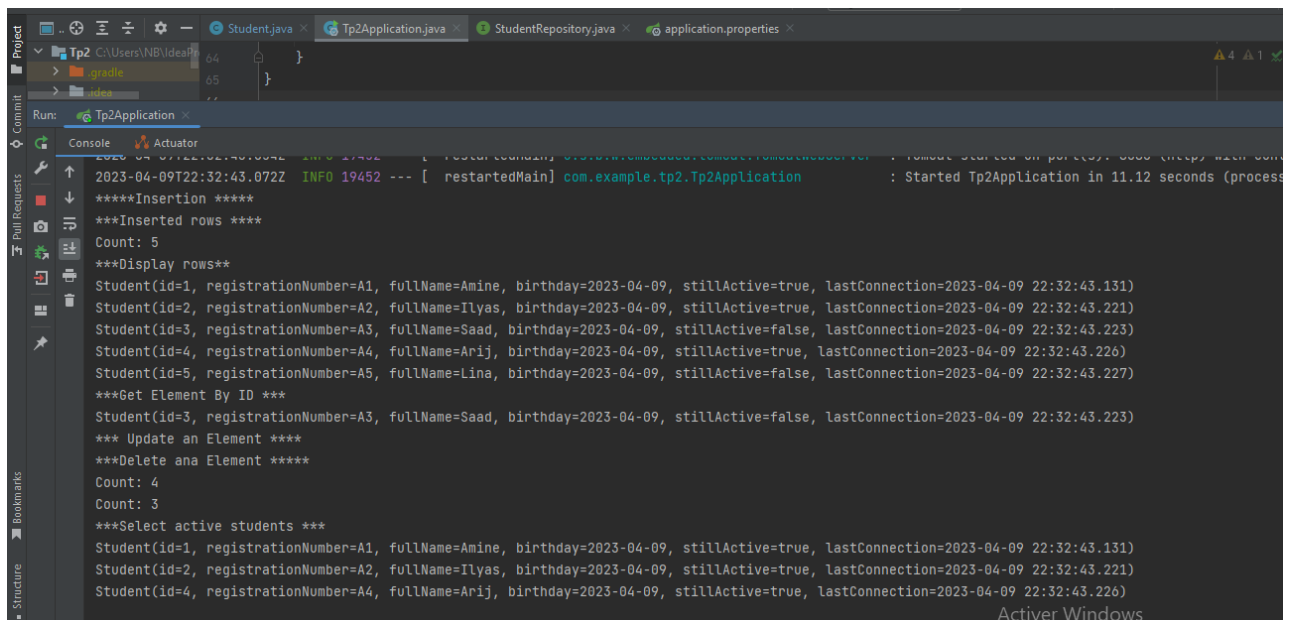
```

50
51     System.out.println("***Delete ana Element *****");
52     studentRepository.delete(student);
53     System.out.println("Count: " +studentRepository.count());
54
55     studentRepository.deleteById(5);
56     System.out.println("Count: "+studentRepository.count());
57
58     System.out.println("***Select active students ***");
59     List<Student> activeStudents = studentRepository.findByStillActive(true);
60     activeStudents.forEach(s -> {
61         System.out.println(s.toString());
62     });
63
64 }
65
66

```

La méthode run utilisé dans la class main s'exécute automatiquement après le lancement du programme, dans ce cas on utilisera la méthode save pour insérer une ligne dans notre table. Voici d'autre méthodes qu'on peut utiliser citons parmi eux findById qui permet de récupérer un enregistrement à partir de son id indiqué comme paramètre, setNomColumn permet d'affecter une valeur a un enregistrement précis puis on utilise la méthode save pour appliquer les modifications, delete pour supprimer un élément en question ou deleteById pour specifier son identifiant

Test



```

Run: Tp2Application
Console
2023-04-09T22:32:43.072Z INFO 19452 --- [ restartedMain] com.example.tp2.Tp2Application : Started Tp2Application in 11.12 seconds (process
****Insertion ****
***Inserted rows ***
Count: 5
***Display rows**
Student(id=1, registrationNumber=A1, fullName=Amine, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 22:32:43.131)
Student(id=2, registrationNumber=A2, fullName=Ilyas, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 22:32:43.221)
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=false, lastConnection=2023-04-09 22:32:43.223)
Student(id=4, registrationNumber=A4, fullName=Arij, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 22:32:43.226)
Student(id=5, registrationNumber=A5, fullName=Lina, birthday=2023-04-09, stillActive=false, lastConnection=2023-04-09 22:32:43.227)
***Get Element By ID ***
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=false, lastConnection=2023-04-09 22:32:43.223)
*** Update an Element ****
***Delete ana Element *****
Count: 4
Count: 3
***Select active students ***
Student(id=1, registrationNumber=A1, fullName=Amine, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 22:32:43.131)
Student(id=2, registrationNumber=A2, fullName=Ilyas, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 22:32:43.221)
Student(id=4, registrationNumber=A4, fullName=Arij, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 22:32:43.226)
Active Windows

```

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:mem:students

User Name: admin

Password:

Connect Test Connection

Connexion à la base de donnée

Auto commit ☒ Max rows: 1000 Auto complete Off Auto select On ?

jdbc:h2:mem:students

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM EMSI_STUDENTS

EMSI_STUDENTS

- ID
- BIRTHDAY
- NAME
- LAST_CONNECTION
- REGISTRATION_N
- STILL_ACTIVE
- Indexes
- INFORMATION_SCHEMA
- Users
- H2 2.1.214 (2022-06-13)

SELECT * FROM EMSI_STUDENTS;

ID	BIRTHDAY	NAME	LAST_CONNECTION	REGISTRATION_N	STILL_ACTIVE
1	2023-04-09	Amine	2023-04-09 22:32:43.131	A1	TRUE
2	2023-04-09	Ilyas	2023-04-09 22:32:43.221	A2	TRUE
4	2023-04-09	Arij	2023-04-09 22:32:43.226	A4	TRUE

(3 rows, 4 ms)

Edit

Affichage de la table EMSI_STUDENTS



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**
Membre de
HONORIS UNITED UNIVERSITIES

Conclusion

En conclusion, la JPA est une spécification Java EE qui permet la gestion des objets persistants dans une base de données relationnelle. Elle fournit un environnement de programmation qui permet d'interagir avec la base de données sans avoir à écrire de code SQL.