



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**

Membre de

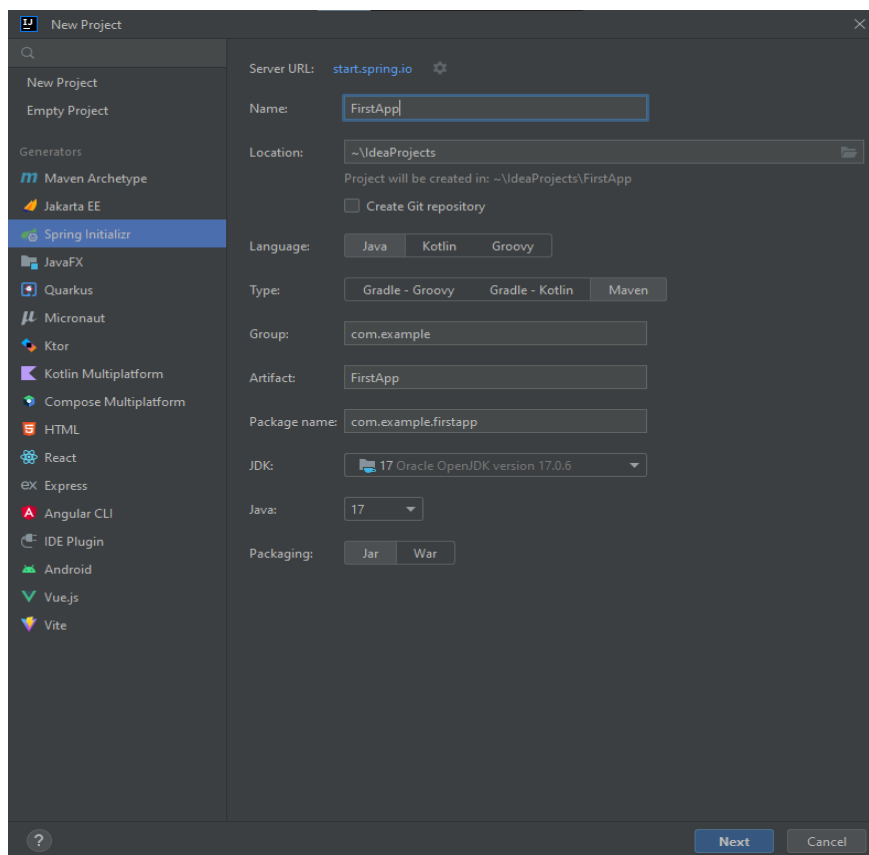
HONORIS UNITED UNIVERSITIES

Compte Rendu TP Jpa, Hibernate, Spring data

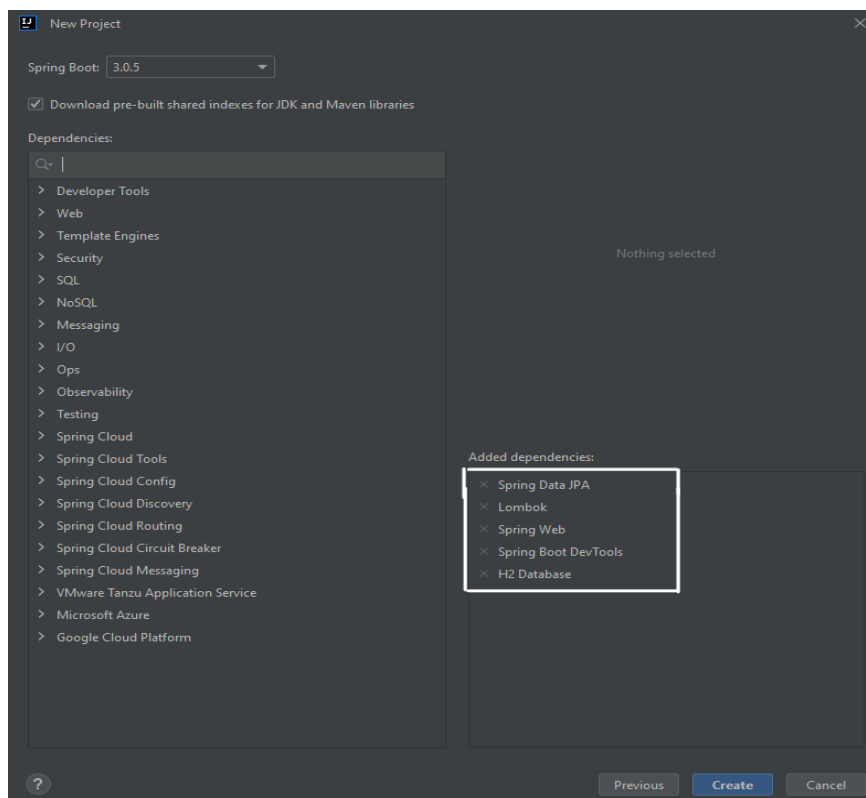
4IIR-EMSI Centre -G5

Réalisé par : Faidi Zahira

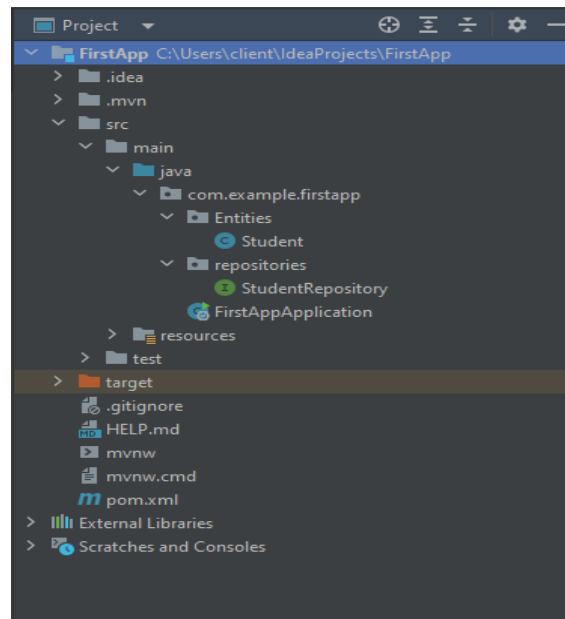
Création de projet Spring



L'ajout des dépendances nécessaire : spring data jpa, lombok, spring web, spring web dev tools, et h2 BD



Structure Globale du projet



Code de l'entité JPA Student

```
package com.example.firstapp.Entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.CreationTimestamp;

import java.util.Date;

@Entity @Table(name="EMSI_STUDENTS")
@Data @AllArgsConstructor @NoArgsConstructor
public class Student {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column(name = "REGISTRATION N", unique = true)
    private String registrationNumber;
    @Column(name = "Name", length = 30, nullable = false)
    private String fullName;
    @Temporal(TemporalType.DATE)
    private Date birthday;
    private Boolean stillActive;
    @Temporal(TemporalType.TIMESTAMP) @CreationTimestamp
    private Date lastConnection;
}
```

Configuration de la data source (application.properties)

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

Création de l'interface Etudiant Repository basé sur spring data

```
package com.example.firstapp.repositories;

import com.example.firstapp.Entities.Student;
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Integer> {

}
```

Implementation des Opérations CRUD

```
package com.example.firstapp;

import com.example.firstapp.Entities.Student;
import com.example.firstapp.repositories.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import java.util.Date;
import java.util.List;

@SpringBootApplication
public class FirstAppApplication implements CommandLineRunner {
    @Autowired
    private StudentRepository studentRepository;

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

    @Override
    public void run(String... args) throws Exception {
        System.out.println("***** Insertion *****");
        studentRepository.save(new Student(null, "A1", "Amine", new Date(), true, null));
        studentRepository.save(new Student(null, "A2", "Ilyas", new Date(), true, null));
        studentRepository.save(new Student(null, "A3", "Saad", new Date(), true, null));
        studentRepository.save(new Student(null, "A4", "Arij", new Date(), true, null));
        studentRepository.save(new Student(null, "A5", "Lina", new Date(), true, null));
        System.out.println("***** Inserted rows *****");
        System.out.println("Count : "+studentRepository.count());
        System.out.println("***** Display rows *****");
        List<Student> students = studentRepository.findAll();
        students.forEach(student -> {
            System.out.println(student.toString());
        });
        System.out.println("***** Get Element By ID *****");
        Student student = studentRepository.findById(3).orElse(null);
        System.out.println(student.toString());
        System.out.println("***** Update an Element *****");
        student.setRegistrationNumber("S3");
        studentRepository.save(student);
        System.out.println("***** Delete an Element *****");
        studentRepository.delete(student);
        System.out.println("Count : "+studentRepository.count());

        studentRepository.deleteById(5);
        System.out.println("Count : "+studentRepository.count());
    }
}
```

Résultat de l'exécution

```
***** Insertion *****
***** Inserted rows *****
Count : 5
***** Display rows *****
Student(id=1, registrationNumber=A1, fullName=Amine, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.88)
Student(id=2, registrationNumber=A2, fullName=Ilyas, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.947)
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.949)
Student(id=4, registrationNumber=A4, fullName=Arij, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.95)
Student(id=5, registrationNumber=A5, fullName=Lina, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.952)
***** Get Element By ID *****
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.949)
***** Update an Element *****
***** Delete an Element *****
Count : 4
Count : 3
```

Visualiser les données à partir de la base de données H2

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

Auto commit Auto complete Off Auto select On

Run Run Selected Auto complete Clear SQL statement:

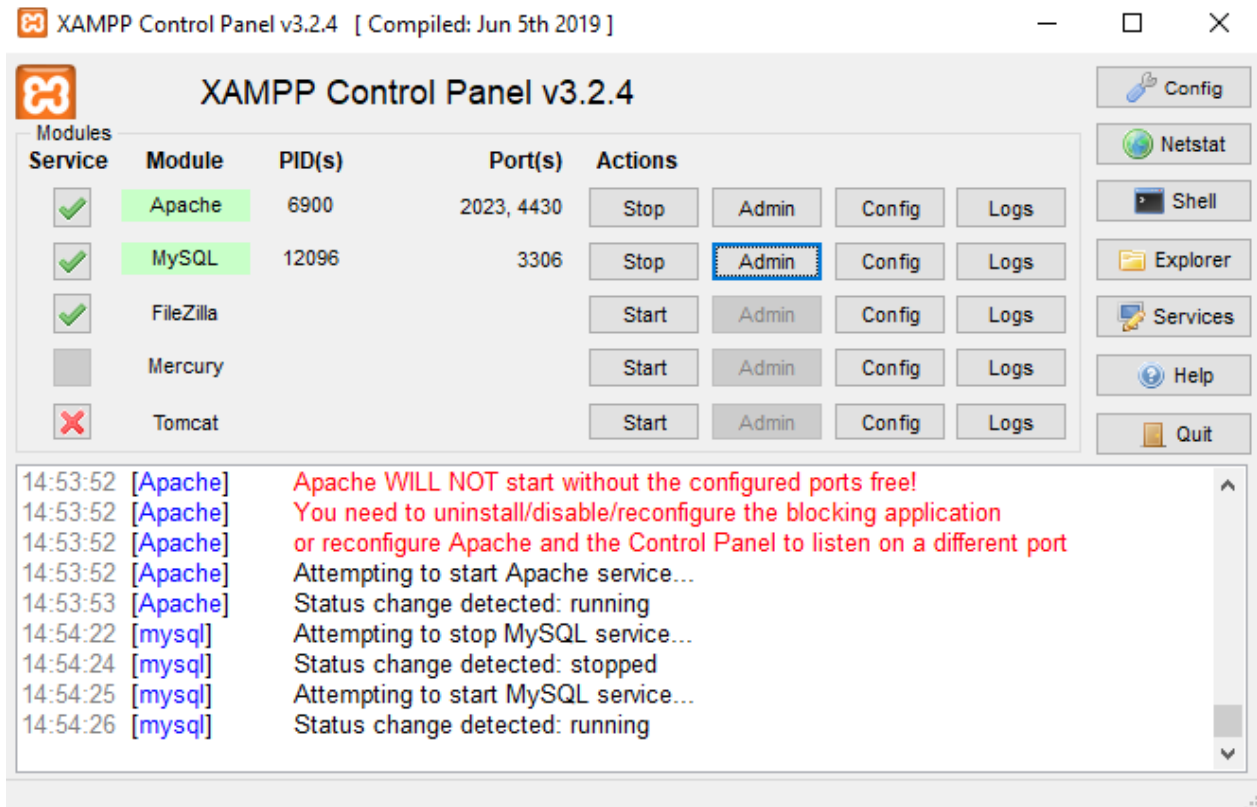
select * from EMSI_Students

ID	BIRTHDAY	NAME	LAST_CONNECTION	REGISTRATION_N	STILL_ACTIVE
1	2023-04-09	Amine	2023-04-09 13:52:13.976	A1	TRUE
2	2023-04-09	Ilyas	2023-04-09 13:52:14.06	A2	TRUE
4	2023-04-09	Arij	2023-04-09 13:52:14.063	A4	TRUE

(3 rows, 3 ms)

Edit

Changer la base de données H2 en MySql



Lancement de service MySQL avec XAMPP ensuite en modifier la configuration du fichier application.properties avec la configuration suivante :

```
spring.datasource.url=jdbc:mysql://localhost:3306/Etudiant?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
server.port=2020
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=true
```

L'ajout de la dépendance JDBC MySQL dans le fichier pom.xml

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.32</version>
</dependency>
```

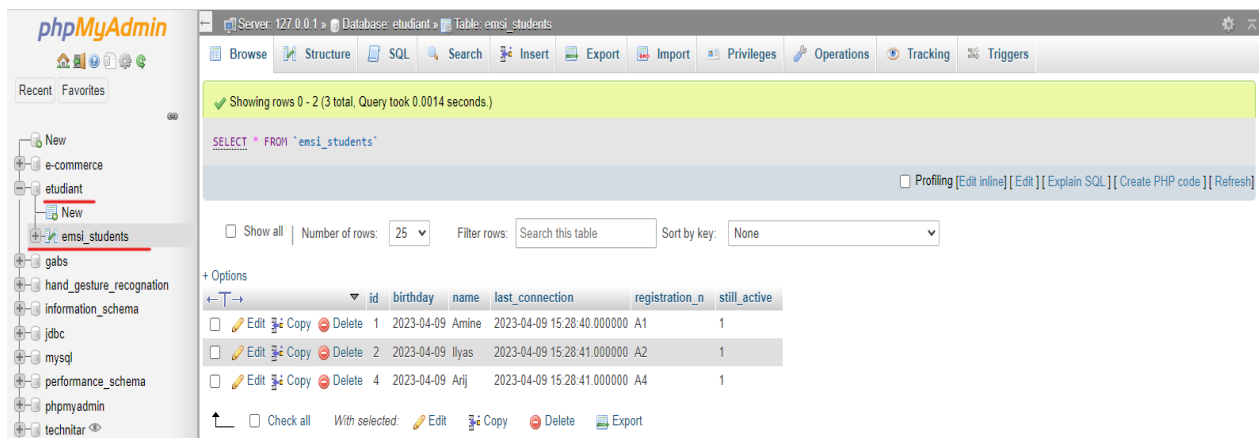
La base de données Etudiant est créée s'il n'existe pas ainsi que la table « emsi_students »

Résultat de compilation des opérations CRUD

```
***** Insertion *****
Hibernate: insert into emsi_students (birthday, name, last_connection, registration_n, still_active) values (?, ?, ?, ?, ?)
Hibernate: insert into emsi_students (birthday, name, last_connection, registration_n, still_active) values (?, ?, ?, ?, ?)
Hibernate: insert into emsi_students (birthday, name, last_connection, registration_n, still_active) values (?, ?, ?, ?, ?)
Hibernate: insert into emsi_students (birthday, name, last_connection, registration_n, still_active) values (?, ?, ?, ?, ?)
***** Inserted rows *****
Hibernate: select count(*) from emsi_students s1_0
Count : 5
***** Display rows *****
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0
Student(id=1, registrationNumber=A1, fullName=Amine, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:40.0)
Student(id=2, registrationNumber=A2, fullName=Ilyas, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
Student(id=4, registrationNumber=A4, fullName=Arif, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
Student(id=5, registrationNumber=A5, fullName=Lina, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
***** Get Element By ID *****
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0 where s1_0.id=?
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
***** Update an Element *****
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0 where s1_0.id=?
Hibernate: update emsi_students set birthday=?, name=?, last_connection=?, registration_n=?, still_active=? where id=?

***** Delete an Element *****
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0 where s1_0.id=?
Hibernate: delete from emsi_students where id=?
Hibernate: select count(*) from emsi_students s1_0
Count : 4
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0 where s1_0.id=?
Hibernate: delete from emsi_students where id=?
Hibernate: select count(*) from emsi_students s1_0
Count : 3
```

Résultat d'exécution dans phpMyAdmin



The screenshot shows the phpMyAdmin interface for the 'etudiant' database. The 'emsi_students' table is selected, and the SQL query 'SELECT * FROM `emsi_students`' is executed. The table contains 3 rows of data. The interface includes a sidebar with a tree view of databases and tables, and a main area with tabs for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, Tracking, and Triggers. The 'Browse' tab is active, showing the table structure and data.

	id	birthday	name	last_connection	registration_n	still_active
<input type="checkbox"/>	1	2023-04-09	Amine	2023-04-09 15:28:40.000000	A1	1
<input type="checkbox"/>	2	2023-04-09	Ilyas	2023-04-09 15:28:41.000000	A2	1
<input type="checkbox"/>	4	2023-04-09	Arif	2023-04-09 15:28:41.000000	A4	1