



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

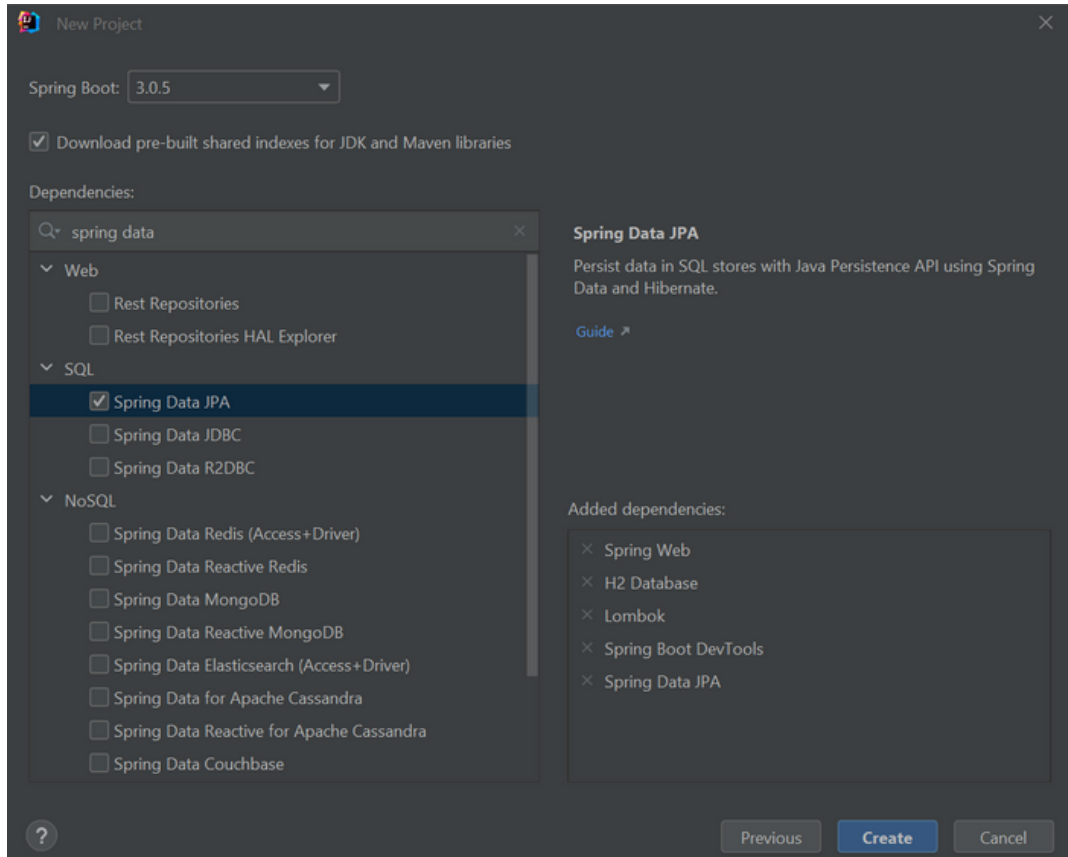
Compte Rendu TP2: Spring Data JPA

RÉALISÉ PAR:
EL YOUSFI SAFAA

2023

Objectif :

- L'objectif de ce tp est de créer un projet Spring Boot avec les dépendances suivantes:



Pour la base de données, on a utilisé deux méthodes:

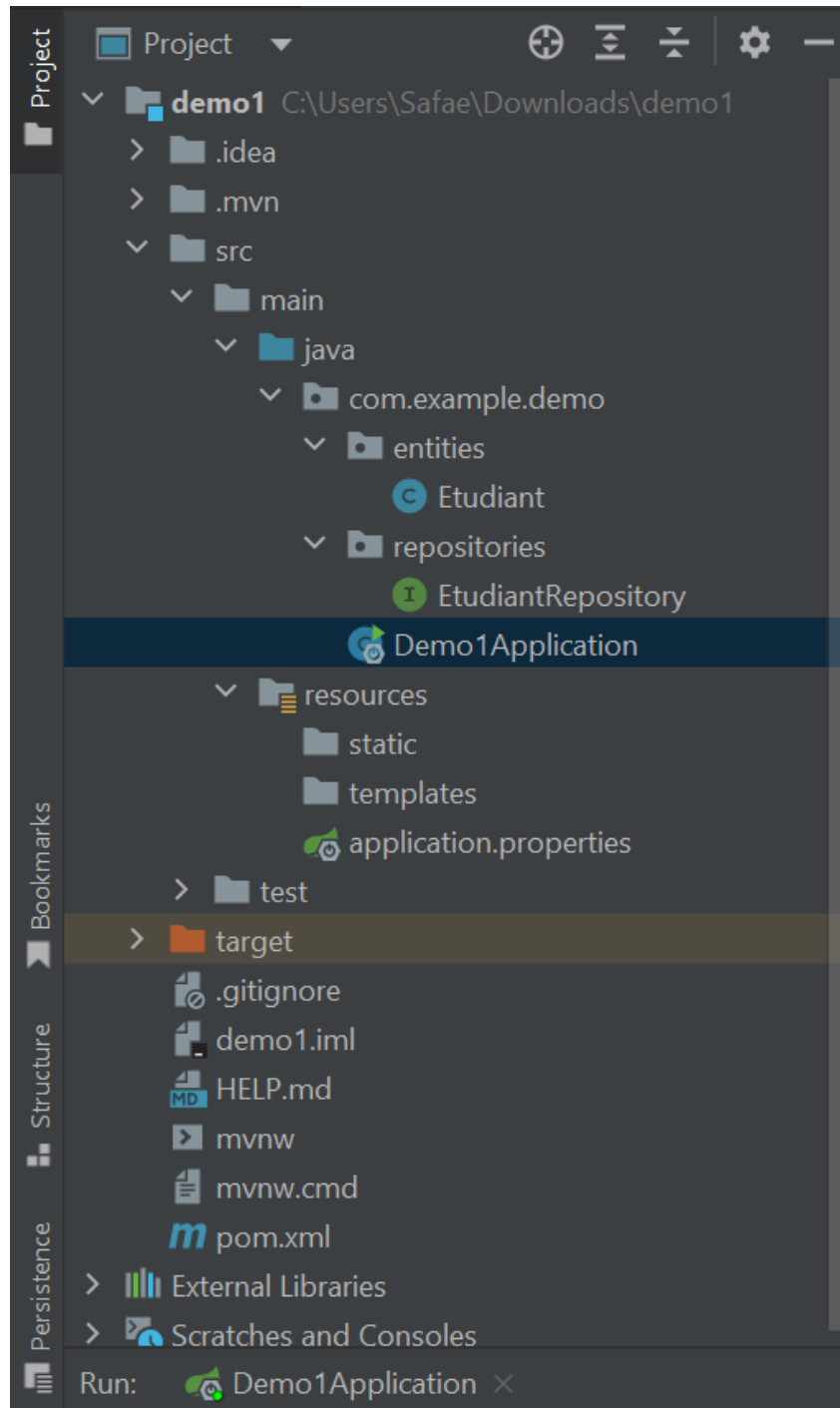
1- *h2 Database*

2- *mysql Database*

Frameworks ORM : Hibernate

- Hibernate est un framework open source gérant la persistance des objets en base de données relationnelle.

Les composants du projet



entities:

- Dossier contenant la classe Java **Etudiant**.

repositories:

- Dossier contenant l'interface **EtudiantRepository**.

DemoApplication:

- Fichier contenant la classe **Main**.

resources:

- Dossier contenant le fichier **application.properties** => fichier de configuration d'une application **Spring Boot**.

Partie 1 :h2

Classe Etudiant

```
1 package com.example.demo.entities;
2 import jakarta.persistence.*;
3 import lombok.*; /* Lombok:C'est un pluggin: à chaque enregistrement du fichier, le compilateur
4 Java compile avec lombok */
5 import java.util.Date;
6
7 @Entity //entité jpa
8 @Table(name="Etudiant") @Data @NoArgsConstructor @AllArgsConstructor
9 /* @Data:Pour ajouter les getters et les setters
10 @NoArgsConstructor: Constructeur sans paramètre
11 @AllArgsConstructor: Constructeur avec paramètre
12 @Id: Champ clé primaire
13 */
14 @Setter @Getter
15 public class Etudiant {
16     no usages
17     @Id
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private Integer id;
20
21     no usages
22     @Column(name="Registration_number", unique = true)
23     private String RegistrationNumber;
```

```
21     no usages
22     @Column(name = "Name", length = 30, nullable = false)
23     private String fullName;
24     no usages
25     @Temporal(TemporalType.DATE)
26     private Date birthday;
27     no usages
28     private Boolean stillActive;
29 }
30
```

Interface EtudiantRepository

```
1 package com.example.demo.repositories;
2
3 import com.example.demo.entities.Etudiant;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 2 usages
7 public interface EtudiantRepository extends JpaRepository<Etudiant,Integer> {
8     // spring data : hériter d'une interface générique JpaRepository<Entité, Type d'id>
9 }
10
```

Classe Demo1Application

```
1 package com.example.demo;
2 import com.example.demo.entities.Etudiant;
3 import com.example.demo.repositories.EtudiantRepository;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.CommandLineRunner;
6 import org.springframework.boot.SpringApplication;
7 import org.springframework.boot.autoconfigure.SpringBootApplication;
8 import java.util.Date;
9 import java.util.List;
10 @SpringBootApplication
11 /* configurer automatiquement Spring, et automatiquement scanner
12 (Scan) le projet intégral afin de découvrir des composants de Spring */
13 public class Demo1Application implements CommandLineRunner {
14     @Autowired
15     //activer l'injection automatique de dépendance
16     private EtudiantRepository etudiantRepository;
17     public static void main(String[] args) { SpringApplication.run(Demo1Application.class, args); }
```

```
20 @Override
21 public void run(String... args) throws Exception {
22     //Create (Insertion)
23     /*L'id est null car elle s'incrémente automatiquement */
24     System.out.println("***** Insert *****");
25     etudiantRepository.save(new Etudiant( id: null, RegistrationNumber: "A1", fullName: "EL YOUSFI Safaa", new Date(),
26     etudiantRepository.save(new Etudiant( id: null, RegistrationNumber: "A2", fullName: "EL YOUSFI Ali", new Date(), st
27     System.out.println("*****");
28     System.out.println("Count:" + etudiantRepository.count());
29     List<Etudiant> etudiantList = etudiantRepository.findAll();
30     etudiantList.forEach(e -> {
31         System.out.println("=====");
32         System.out.println(e.getId());
33         System.out.println(e.getRegistrationNumber());
34         System.out.println(e.getFullName());
35     });
```

```
36     System.out.println("***** Read *****");
37     Etudiant etudiant = etudiantRepository.findById(1).orElse( other: null);
38     System.out.println(etudiant.toString());
39     System.out.println("***** Update *****");
40     etudiant.setRegistrationNumber("S2");
41     etudiantRepository.save(etudiant);
42     System.out.println(etudiant.toString());
43     System.out.println("***** Delete *****");
44     etudiantRepository.deleteById(1);
45     System.out.println("Count: " + etudiantRepository.count());
46
47 }
48 }
49 }
```

Résultat d'exécution

Auto commit

Max rows: 1000

Auto complete

Auto select

jdbc:h2:mem:students

ETUDIANT

ID

BIRTHDAY

NAME

REGISTRATION_NUMBER

STILL_ACTIVE

Indexes

INFORMATION_SCHEMA

Users

H2 2.1.214 (2022-06-13)

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT * FROM ETUDIANT

ID	BIRTHDAY	NAME	REGISTRATION_NUMBER	STILL_ACTIVE
1	2023-04-09	EL YOUSFI Safaa	A1	TRUE
2	2023-04-09	EL YOUSFI Ali	A2	TRUE

(2 rows, 2 ms)

Edit

Auto commit

Max rows: 1000

Auto complete

Auto select

jdbc:h2:mem:students

ETUDIANT

ID

BIRTHDAY

NAME

REGISTRATION_NUMBER

STILL_ACTIVE

Indexes

INFORMATION_SCHEMA

Users

H2 2.1.214 (2022-06-13)

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT * FROM ETUDIANT WHERE ID=1

ID	REGISTRATION_NUMBER	BIRTHDAY	NAME	STILL_ACTIVE
1	A1	2023-04-09	EL YOUSFI Safaa	TRUE

(1 row, 21 ms)

Edit

```
*****
Count:2
=====
1
A1
EL YOUSFI Safaa
=====
2
A2
EL YOUSFI Ali
***** Read *****
Etudiant(id=1, RegistrationNumber=A1, fullName=EL YOUSFI Safaa, birthday=2023-04-09, stillActive=true)
***** Update *****
Etudiant(id=1, RegistrationNumber=S2, fullName=EL YOUSFI Safaa, birthday=2023-04-09, stillActive=true)
***** Delete *****
Count:1
```


Partie 2 :mysql

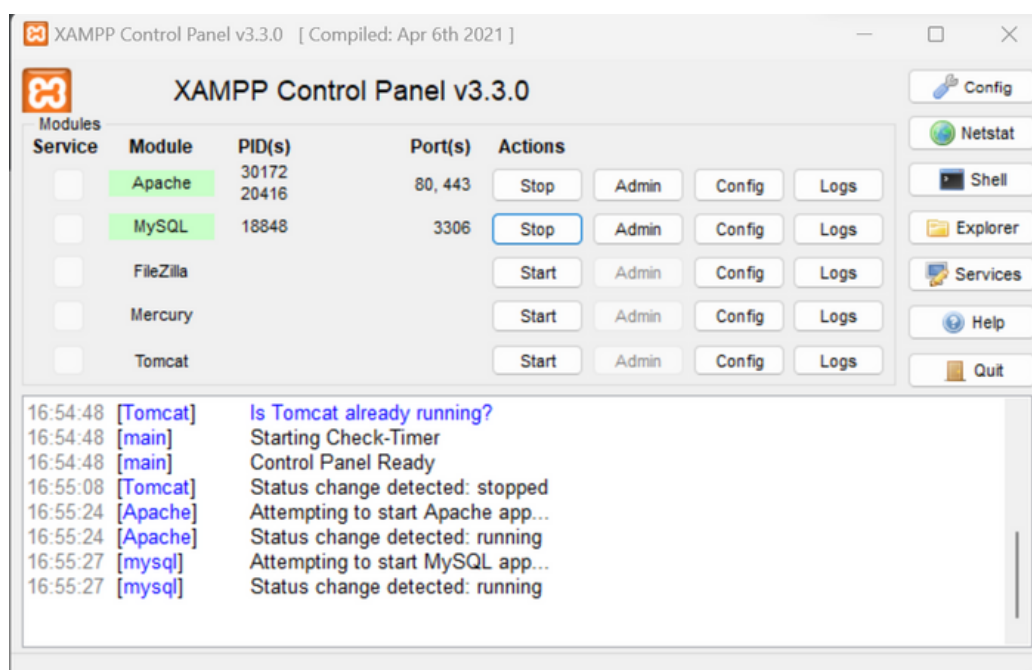
On met en block de commentaire la dépendance de h2 et on ajoute la dépendance de mysql

```
35 <!--<dependency>
36 <groupId>com.h2database</groupId>
37 <artifactId>h2</artifactId>
38 <scope>runtime</scope>
39 </dependency>-->
40 <dependency>
41 <groupId>mysql</groupId>
42 <artifactId>mysql-connector-java</artifactId>
43 <version>8.0.25</version>
44 </dependency>
```

On change les paramètres du fichier de configuration application.properties

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/students?createDatabaseIfNotExist=true
2 spring.datasource.username=root
3 spring.datasource.password=
4 #spring.h2.console.enabled=true
5 server.port=8080
6 spring.jpa.hibernate.ddl-auto=update
7 spring.jpa.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
8 spring.jpa.show-sql=true
9
```

On active Apache et MySQL



Résultat d'exécution

Base de donnée →
Table →

phpMyAdmin interface showing the 'etudiant' table in the 'students' database. The table structure is displayed with columns: id, registration_number, birthday, name, and still_active. The table contains 2 rows of data.

	id	registration_number	birthday	name	still_active
<input type="checkbox"/>	2	A2	2023-04-09	EL YOUSFI Ali	1
<input type="checkbox"/>	3	A1	2023-04-09	EL YOUSFI Safaa	1

phpMyAdmin interface showing the execution of an UPDATE query on the 'etudiant' table. The query updates the 'registration_number' and 'still_active' for the row with id=2. The result shows 1 line affected.

```
UPDATE `etudiant` SET `registration_number` = 'S2', `still_active` = b'1' WHERE `etudiant`.`id` = 2;
```

Affichage des lignes 0 - 1 (total de 2, traitement en 0,0006 seconde(s).)

```
SELECT * FROM `etudiant`
```

	id	registration_number	birthday	name	still_active
<input type="checkbox"/>	2	S2	2023-04-09	EL YOUSFI Ali	1
<input type="checkbox"/>	3	A1	2023-04-09	EL YOUSFI Safaa	1