



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

Membre de **HONORIS UNITED UNIVERSITIES**

Compte Rendu Du TP JPA

Fait par KNAR Imane

On va commencer par créer un projet Spring Boot :

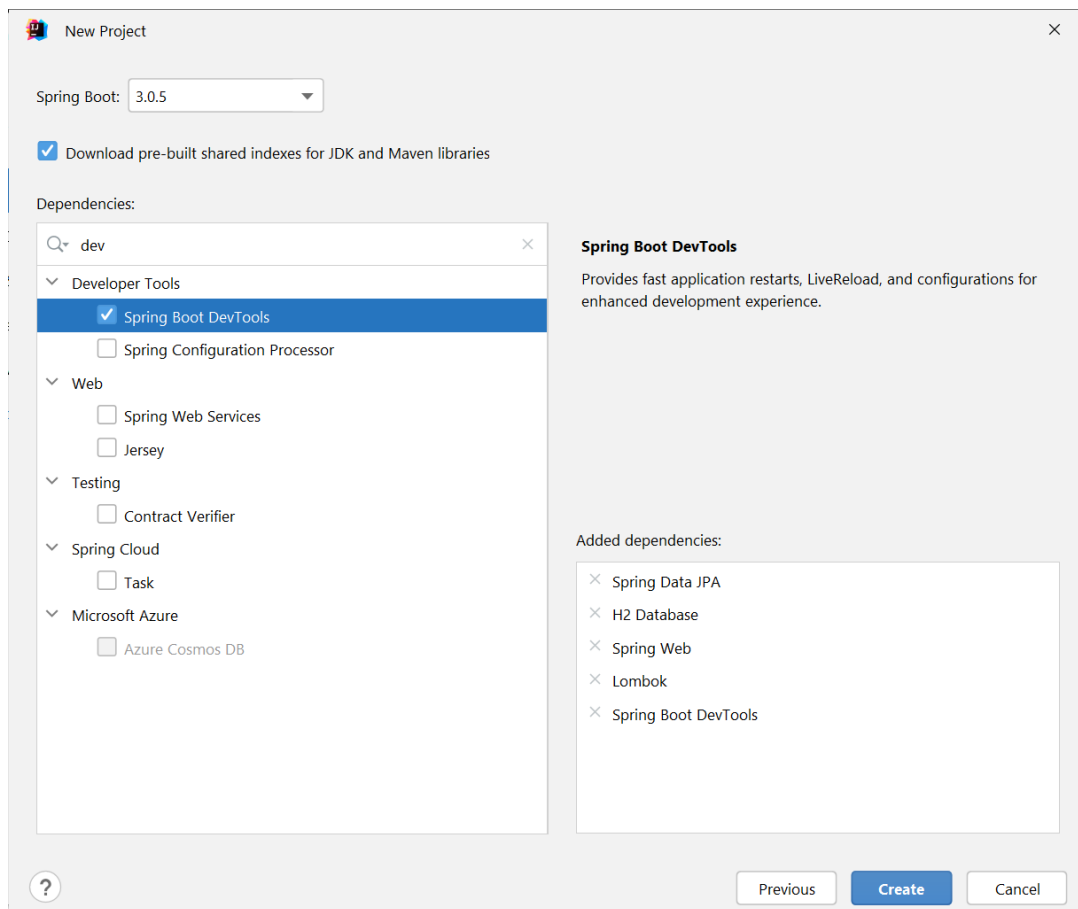
Spring Boot est une version de Spring qui se base sur l'autoconfiguration, càd on aura pas besoin de créer beaucoup de fichiers de configuration xml.. etc.

The screenshot shows the 'New Project' dialog in an IDE. The 'Spring Initializr' generator is selected in the left sidebar. The main form contains the following fields and options:

- Server URL: start.spring.io
- Name:
- Location: (Project will be created in: ~\Desktop\TP-JPA)
- ☒ Create Git repository
- Language: ☐ Java ☐ Kotlin ☐ Groovy
- Type: ☐ Gradle - Groovy ☐ Gradle - Kotlin ☐ Maven
- Group:
- Artifact:
- Package name:
- JDK:
- Java:
- Packaging: ☐ Jar ☐ War

Buttons at the bottom: **Next** and **Cancel**.

On va ajouter des dépendances :



On va créer l'entité JPA Etudiant :

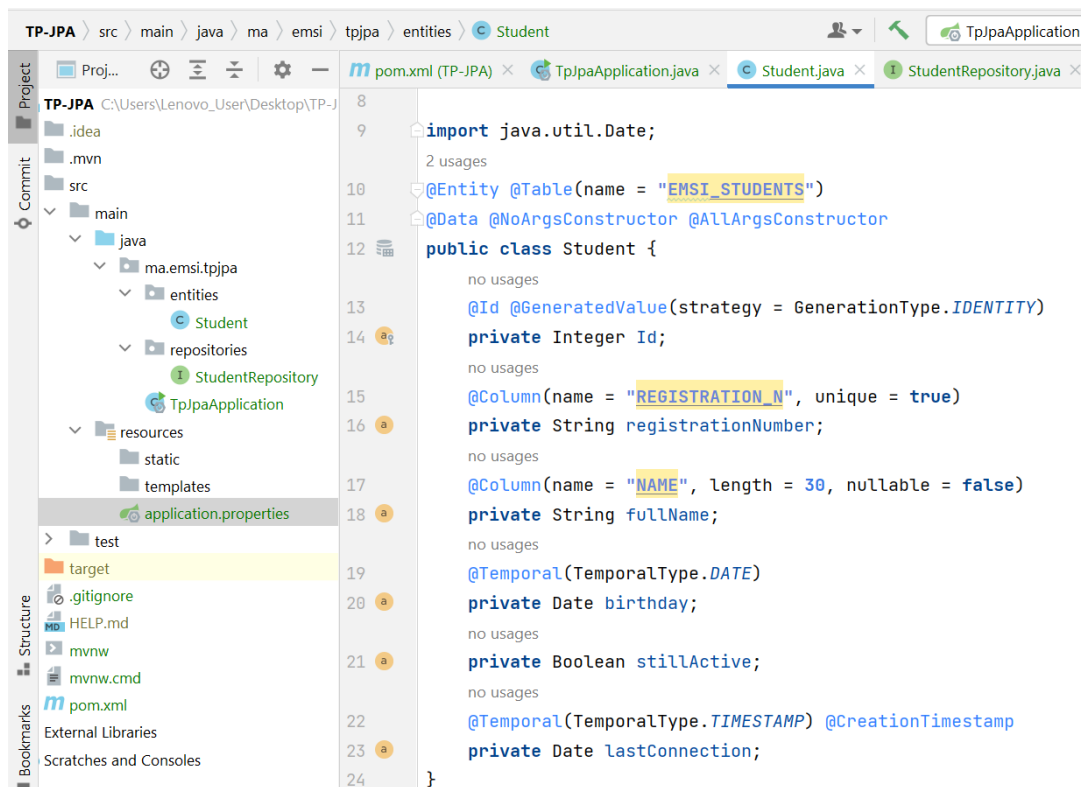
->L'annotation `@Data` va ajouter automatiquement les getters et setters.

->`@NoArgsConstructor` va ajouter un constructeur sans paramètres.

->`@AllArgsConstructor` va ajouter un constructeur avec paramètres.

Pour que cette classe soit une entité JPA on doit ajouter `@Entity`.

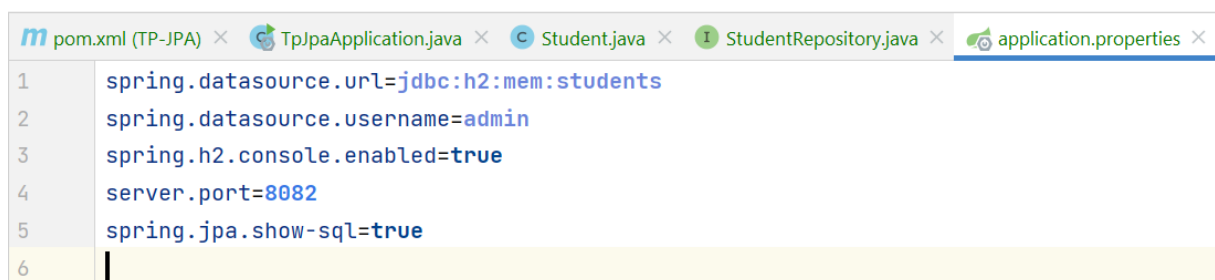
Une entité doit avoir un ID, alors on va ajouter `@Id`



On va configurer le Data source :

On va d'abord travailler avec une base de données h2.

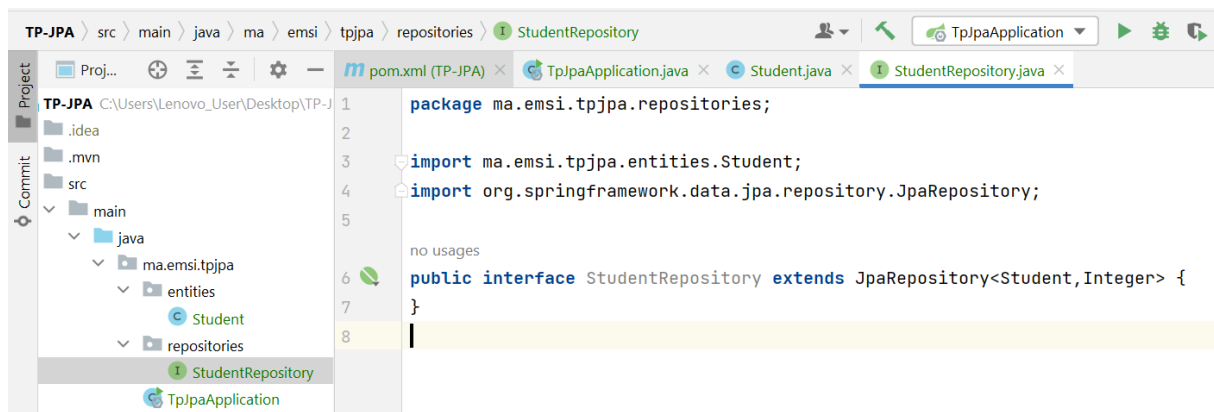
La propriété show-sql permet d'afficher dans la console les requêtes SQL qui sont exécutées.



On va créer l'interface Etudiant Repository basé sur Spring Data :

Cette interface va hériter d'une interface générique dont on va lui spécifier 2 types génériques :

- 1) L'entité qu'on veut gérer
- 2) Le type de l'Id de cette entité

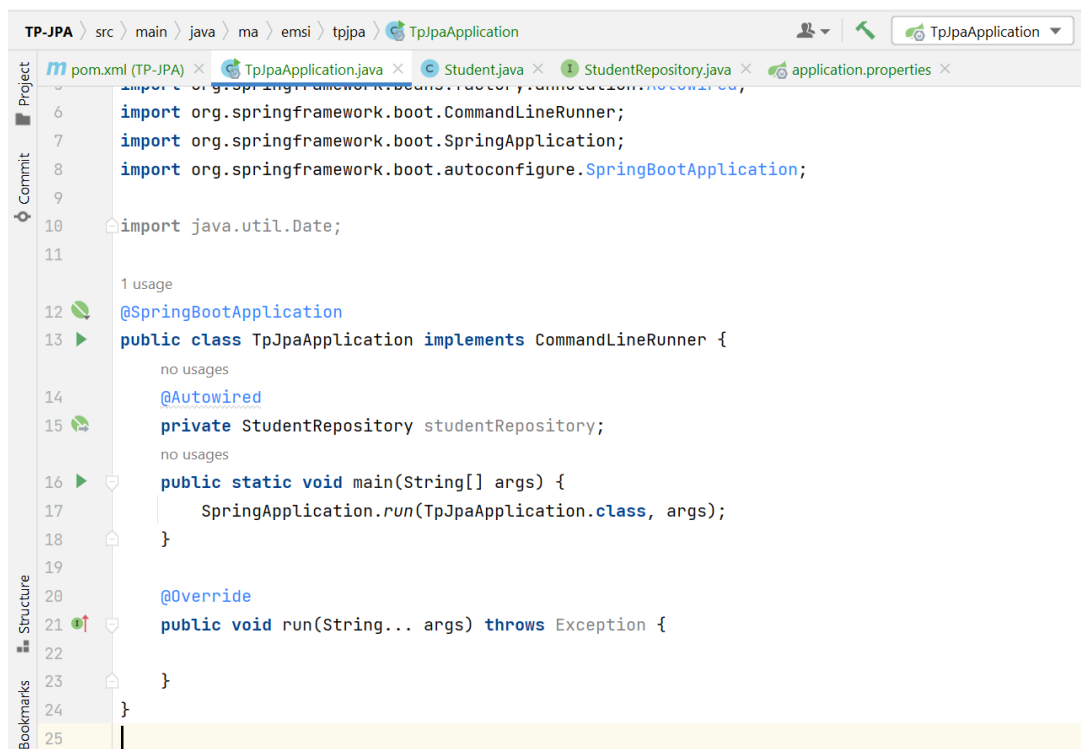


On va tester l'application avec des opérations CRUD :

La classe Application va implémenter une interface CommandLineRunner où on sera obligé de redéfinir la méthode run.

Si on veut faire des tests au démarrage de l'application, on va les mettre à l'intérieur de la méthode run.

On a besoin de gérer les étudiants, c'est pour cela qu'on doit déclarer un attribut de type StudentRepository. Et pour faire l'injection des dépendances, on va utiliser l'annotation @Autowired.



On va ajouter des étudiants dans la base de données :

```
20 @Override
21 public void run(String... args) throws Exception {
22     studentRepository.save(
23         new Student( Id: null, registrationNumber: "S1", fullName: "Imane", new Date(), stillActive: true, lastConnection: null));
24     studentRepository.save(
25         new Student( Id: null, registrationNumber: "S2", fullName: "Aymen", new Date(), stillActive: true, lastConnection: null));
26     studentRepository.save(
27         new Student( Id: null, registrationNumber: "S3", fullName: "Youssef", new Date(), stillActive: false, lastConnection: null));
28     studentRepository.save(
29         new Student( Id: null, registrationNumber: "S4", fullName: "Hasnaa", new Date(), stillActive: false, lastConnection: null));
30     studentRepository.save(
31         new Student( Id: null, registrationNumber: "S5", fullName: "Hakim", new Date(), stillActive: true, lastConnection: null));
32     }
33 }
```

Dans la base de données :

The screenshot shows a database console interface. On the left, a tree view displays the database structure, including the 'EMSI_STUDENTS' table. The main area shows the SQL statement 'SELECT * FROM EMSI_STUDENTS;' and its results. The results are displayed in a table with 6 columns: ID, BIRTHDAY, NAME, LAST_CONNECTION, REGISTRATION_N, and STILL_ACTIVE. There are 5 rows of data.

ID	BIRTHDAY	NAME	LAST_CONNECTION	REGISTRATION_N	STILL_ACTIVE
1	2023-04-09	Imane	2023-04-09 15:14:31.045	S1	TRUE
2	2023-04-09	Aymen	2023-04-09 15:14:31.108	S2	TRUE
3	2023-04-09	Youssef	2023-04-09 15:14:31.11	S3	FALSE
4	2023-04-09	Hasnaa	2023-04-09 15:14:31.111	S4	FALSE
5	2023-04-09	Hakim	2023-04-09 15:14:31.112	S5	TRUE

(5 rows, 5 ms)

On va afficher dans la console :

The screenshot shows an IDE with a Java project. The main window displays the code for 'TpJpaApplication.java'. The code includes a 'run' method that saves five students and then prints them out. The console window at the bottom shows the output of the application, including the startup logs and the list of students.

```
2023-04-09T15:23:52.529Z INFO 13524 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-04-09T15:23:52.605Z INFO 13524 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8082 (http) with
2023-04-09T15:23:52.630Z INFO 13524 --- [ restartedMain] ma.emsi.tpjpa.TpJpaApplication : Started TpJpaApplication in 7.666 seconds (
***** Affichage des lignes *****
Student(Id=1, registrationNumber=S1, fullName=Imane, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:23:52.676)
Student(Id=2, registrationNumber=S2, fullName=Aymen, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:23:52.727)
Student(Id=3, registrationNumber=S3, fullName=Youssef, birthday=2023-04-09, stillActive=false, lastConnection=2023-04-09 15:23:52.729)
Student(Id=4, registrationNumber=S4, fullName=Hasnaa, birthday=2023-04-09, stillActive=false, lastConnection=2023-04-09 15:23:52.73)
Student(Id=5, registrationNumber=S5, fullName=Hakim, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:23:52.732)
```

On va récupérer un étudiant par son ID :

```
System.out.println("***** Get Element By Id *****");
Student student = studentRepository.findById(2).orElse(null);
if (student!=null) System.out.println(student);
}
|
```

```
***** Get Element By Id *****
Student(Id=2, registrationNumber=S2, fullName=Aymen, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:32:02.28)
```

On va mettre à jour l'élément qu'on vient de récupérer :

```
System.out.println("***** Update an Element *****");
student.setRegistrationNumber("A2");
studentRepository.save(student);
}
|
```

```
SELECT * FROM EMSI_STUDENTS|
```

SELECT * FROM EMSI_STUDENTS;					
ID	BIRTHDAY	NAME	LAST_CONNECTION	REGISTRATION_N	STILL_ACTIVE
1	2023-04-09	Imane	2023-04-09 16:37:55.943	S1	TRUE
2	2023-04-09	Aymen	2023-04-09 16:37:56.025	A2	TRUE
3	2023-04-09	Youssef	2023-04-09 16:37:56.027	S3	FALSE
4	2023-04-09	Hasnaa	2023-04-09 16:37:56.028	S4	FALSE
5	2023-04-09	Hakim	2023-04-09 16:37:56.03	S5	TRUE

(5 rows, 6 ms)

Edit

On va supprimer un étudiant par son ID :

```
System.out.println("***** Delete an Element *****");
studentRepository.deleteById(4);
}
|
```

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT * FROM EMSI_STUDENTS |

SELECT * FROM EMSI_STUDENTS;

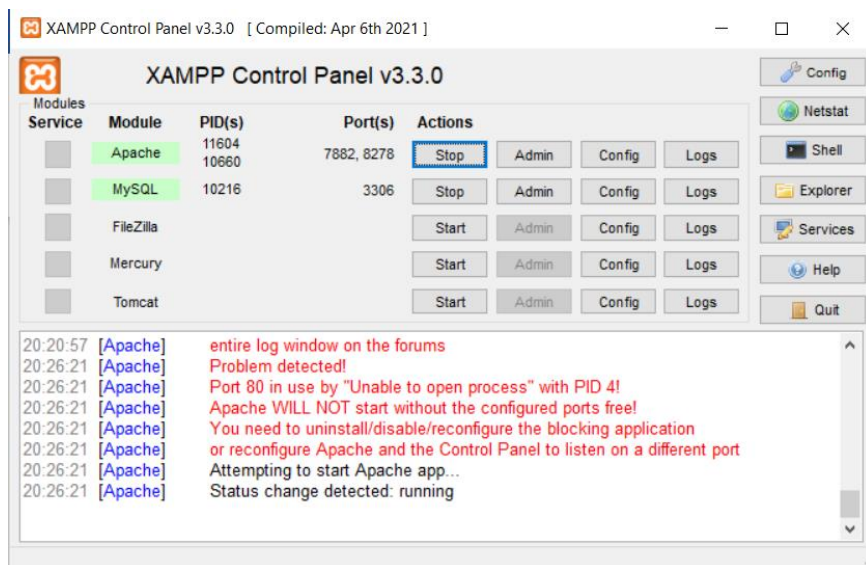
ID	BIRTHDAY	NAME	LAST_CONNECTION	REGISTRATION_N	STILL_ACTIVE
1	2023-04-09	Imane	2023-04-09 16:45:53.659	S1	TRUE
2	2023-04-09	Aymen	2023-04-09 16:45:53.761	A2	TRUE
3	2023-04-09	Youssef	2023-04-09 16:45:53.763	S3	FALSE
5	2023-04-09	Hakim	2023-04-09 16:45:53.769	S5	TRUE

(4 rows, 4 ms)

Edit

On va utiliser une BD MySQL au lieu de h2 :

On va démarrer XAMPP.



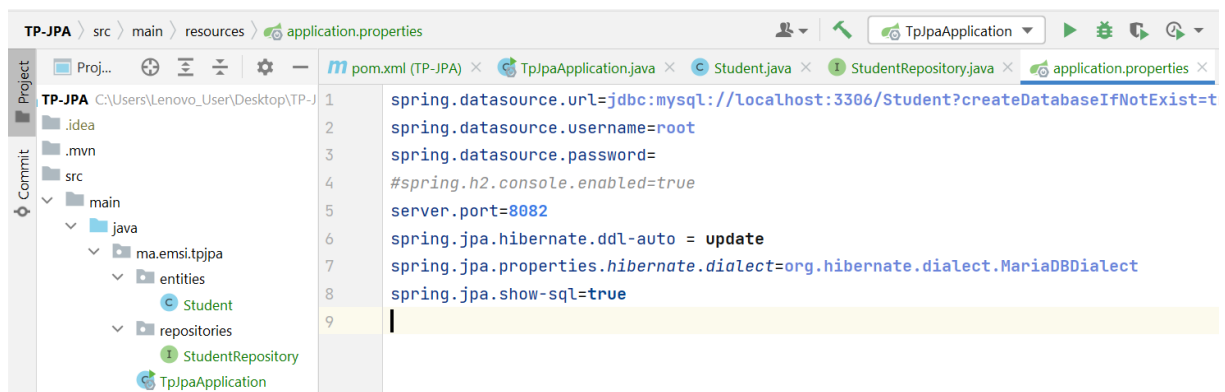
On va modifier le fichier pom.xml

```

34 </dependency>
35 <!--<dependency>
36     <groupId>com.h2database</groupId>
37     <artifactId>h2</artifactId>
38     <scope>runtime</scope>
39 </dependency>-->
40 <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
41 <dependency>
42     <groupId>mysql</groupId>
43     <artifactId>mysql-connector-java</artifactId>
44     <version>8.0.32</version>
45 </dependency>
46

```

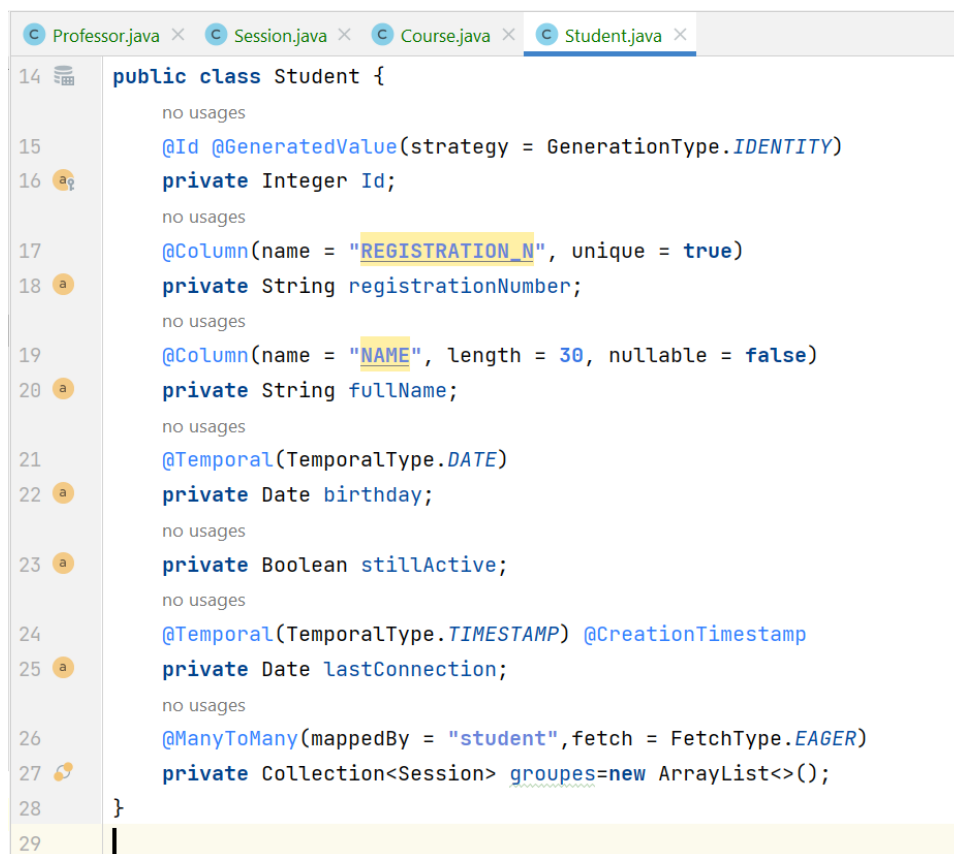
On va modifier le fichier application.properties :



```
1 spring.datasource.url=jdbc:mysql://localhost:3306/Student?createDatabaseIfNotExist=t
2 spring.datasource.username=root
3 spring.datasource.password=
4 #spring.h2.console.enabled=true
5 server.port=8082
6 spring.jpa.hibernate.ddl-auto = update
7 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
8 spring.jpa.show-sql=true
9
```

On va faire le mapping des associations :

Entity Student :



```
14 public class Student {
15     no usages
16     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Integer Id;
18     no usages
19     @Column(name = "REGISTRATION_N", unique = true)
20     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     no usages
30     @Temporal(TemporalType.TIMESTAMP) @CreationTimestamp
31     private Date lastConnection;
32     no usages
33     @ManyToMany(mappedBy = "student", fetch = FetchType.EAGER)
34     private Collection<Session> groupes=new ArrayList<>();
35 }
36
```


Entity Session :

```
Professor.java x Session.java x Course.java x Student.java x
4 usages
13 @Entity
14 @Data @NoArgsConstructor @AllArgsConstructor
15 public class Session {
    no usages
16 @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
17 private int id;
    no usages
18 @Temporal(TemporalType.DATE)
19 private Date date;
    no usages
20 @Temporal(TemporalType.TIME)
21 private Date start_time;
    no usages
22 @Temporal(TemporalType.TIME)
23 private Date end_time;
    no usages
24 @ManyToMany(fetch = FetchType.EAGER)
25 private Collection<Student> students=new ArrayList<>();
    no usages
26 @ManyToOne
27 private Course course;
28 }
29 |
```

Entity Course :

```
Professor.java x Session.java x Course.java x Student.java x
4 usages
10 @Entity
11 @Data @NoArgsConstructor @AllArgsConstructor
12 public class Course {
    no usages
13 @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
14 private int id;
    no usages
15 @Column(length = 30, nullable = false, unique = false)
16 private String fullName;
    no usages
17 private String title;
    no usages
18 private String description;
    no usages
19 private int timing;
    no usages
20 @OneToMany(mappedBy = "Course", fetch = FetchType.LAZY)
21 private Collection<Session> session;
    no usages
22 @OneToOne
23 private Professor consultation;
24 }
25 |
```

Entity Professor :

```
Professor.java x Session.java x Course.java x Student.java x
6 import lombok.NoArgsConstructor;
7 import org.hibernate.annotations.CreationTimestamp;
8
9 import java.util.Date;
10
11 3 usages
12 @Entity
13 @Data @NoArgsConstructor @AllArgsConstructor
14 public class Professor {
15     no usages
16     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private int id;
18     no usages
19     @Column(length = 30, nullable = false)
20     private String fullName;
21     no usages
22     @Temporal(TemporalType.DATE) @CreationTimestamp
23     private Date assignmentDate;
24     no usages
25     @OneToOne(mappedBy = "professor")
26     private Course rendezVous;
27 }
```