



Compte Rendu : Spring Data JPA

— **DE** —

2 ÈME ANNÉE CYCLE D'INGÉNIEUR

GENIE INFORMATIQUE

AKRAM LKIHAL

Encadrée par :

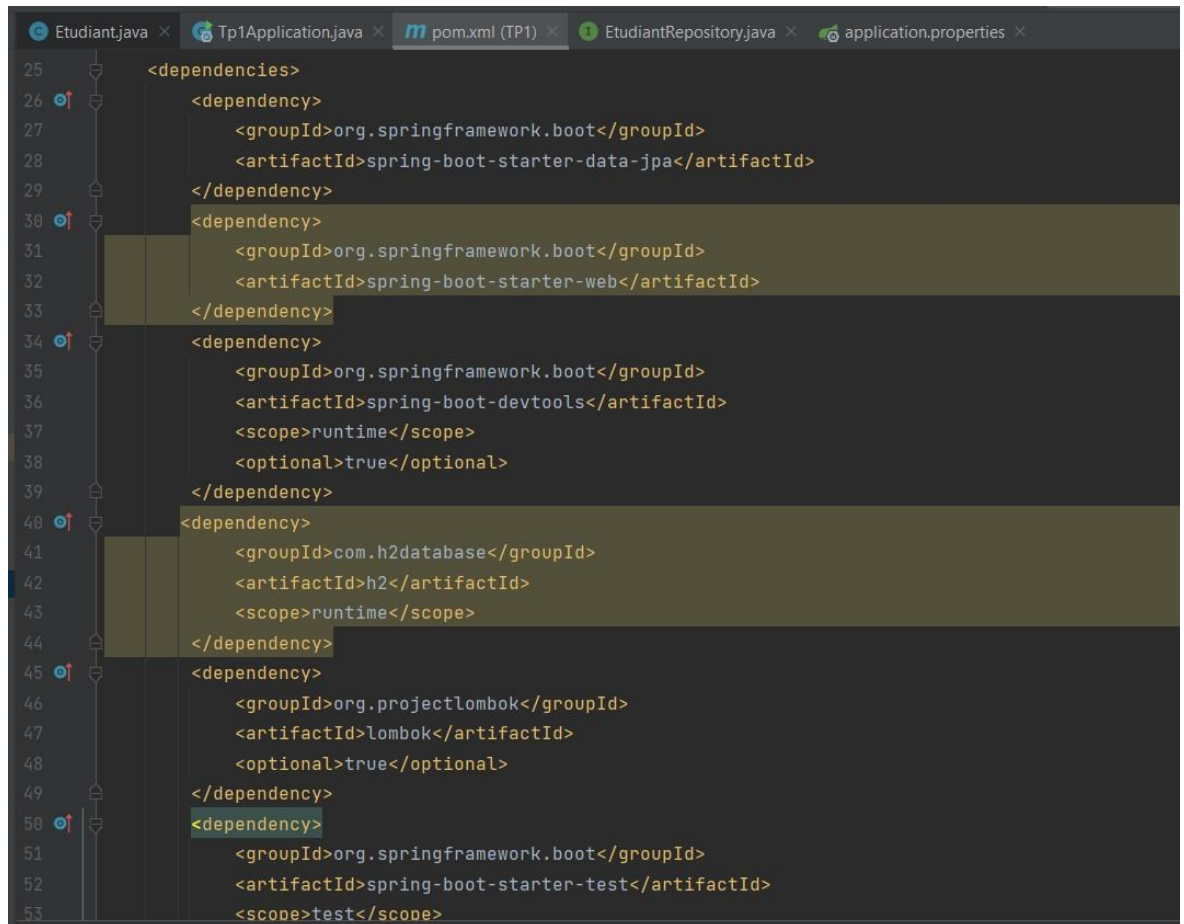
Mme. BADRI TIJANE

Établissement :

EMSI

(École Marocaine des Sciences de L'Ingénieur)

Année universitaire 2023-2024



```
25 <dependencies>
26 <dependency>
27   <groupId>org.springframework.boot</groupId>
28   <artifactId>spring-boot-starter-data-jpa</artifactId>
29 </dependency>
30 <dependency>
31   <groupId>org.springframework.boot</groupId>
32   <artifactId>spring-boot-starter-web</artifactId>
33 </dependency>
34 <dependency>
35   <groupId>org.springframework.boot</groupId>
36   <artifactId>spring-boot-devtools</artifactId>
37   <scope>runtime</scope>
38   <optional>true</optional>
39 </dependency>
40 <dependency>
41   <groupId>com.h2database</groupId>
42   <artifactId>h2</artifactId>
43   <scope>runtime</scope>
44 </dependency>
45 <dependency>
46   <groupId>org.projectlombok</groupId>
47   <artifactId>lombok</artifactId>
48   <optional>true</optional>
49 </dependency>
50 <dependency>
51   <groupId>org.springframework.boot</groupId>
52   <artifactId>spring-boot-starter-test</artifactId>
53   <scope>test</scope>
```

1) Créer l'entité Jpa Etudiant.

Création d'une classe Java représentant une entité appelée Etudiant en utilisant les annotations JPA (Java Persistence API). Passons en revue les annotations utilisées dans cette classe :

@Entity : Cette annotation indique que la classe Etudiant est une entité JPA, ce qui signifie qu'elle sera mappée sur une table de base de données.

@Data : Il s'agit d'une annotation Lombok qui génère automatiquement les méthodes getter, setter, equals, hashCode et toString pour la classe.

@NoArgsConstructor : Il s'agit d'une annotation Lombok qui génère un constructeur sans argument pour la classe.

@AllArgsConstructor : Il s'agit d'une annotation Lombok qui génère un constructeur avec tous les arguments pour la classe.

@ToString : Il s'agit d'une annotation Lombok qui génère une méthode toString() pour la classe.

@Id : Cette annotation indique que le champ id est la clé primaire de l'entité.

@GeneratedValue : Cette annotation spécifie la stratégie de génération des valeurs pour la clé primaire. Dans ce cas, elle utilise la stratégie GenerationType.IDENTITY, ce qui signifie que les valeurs de la clé primaire seront générées automatiquement par la base de données.

@Column : Cette annotation est utilisée pour spécifier la correspondance entre le champ de l'entité et la colonne de base de données correspondante. Dans ce cas, elle spécifie le nom, la longueur et la possibilité de nullité des champs name et CNE.

@Temporal : Cette annotation est utilisée pour spécifier le type de données temporelles (par exemple, date, heure, horodatage) pour le champ datenaissance.

private Integer id; : Ce champ représente la clé primaire de l'entité Etudiant.

private String name; : Ce champ représente le nom de l'entité Etudiant, qui sera mappé sur une colonne de base de données appelée "NAME".

private Date datenaissance; : Ce champ représente la date de naissance de l'entité Etudiant, qui sera mappée sur une colonne de base de données.

private Integer Cne; : Ce champ représente le "CNE" (Identifiant national de l'étudiant) de l'entité Etudiant, qui sera mappé sur une colonne de base de données appelée "CNE" et devra avoir des valeurs uniques.

```
Etudiant.java x Tp1Application.java x EtudiantRepository.java x application.properties x
1 package ma.emsi.springjpa.entities;
2
3 import jakarta.persistence.*;
4 import lombok.*;
5
6
7 import java.util.Date;
8
9 @Entity
10 @Data @NoArgsConstructor @AllArgsConstructor @ToString
11 public class Etudiant {
12     @Id @GeneratedValue(strategy= GenerationType.IDENTITY)
13     private Integer id;
14     @Column(name="NAME", length = 30, nullable = false)
15     private String name;
16     @Temporal(TemporalType.DATE)
17     private Date datenaissance;
18     @Column(name="CNE", unique = true)
19     private Integer Cne;
20 }
```

2) Configurer le data source (application.properties)

Les deux premières lignes de configuration sont destinées à une application utilisant le framework Spring et la base de données H2.

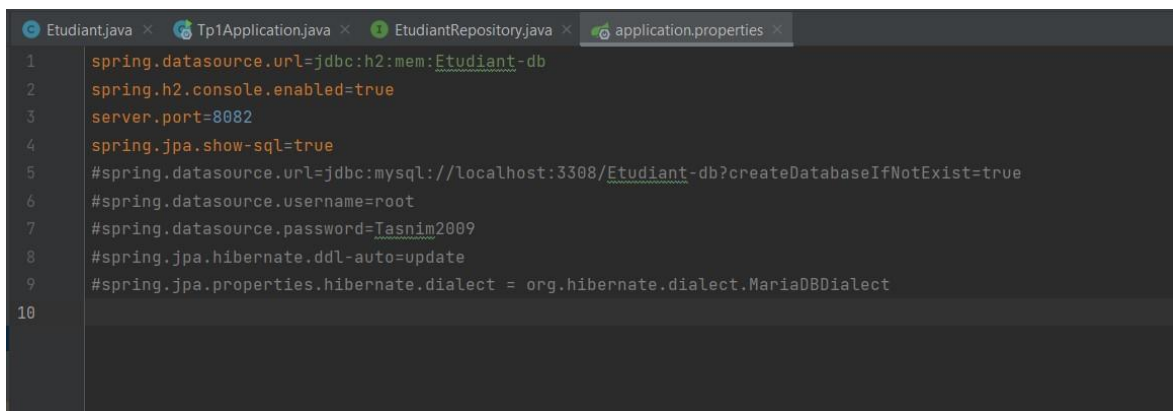
spring.datasource.url=jdbc:h2:mem:Etudiant-db: Cette ligne de configuration définit l'URL de la base de données H2 pour l'application Spring. Elle utilise une base de données H2 en mémoire avec le nom de base de données "Etudiant-db".

spring.h2.console.enabled=true: Cette ligne de configuration active la console H2 dans l'application Spring.

server.port=8082: Cette ligne de configuration définit le numéro de port sur lequel l'application Spring sera déployée et écoutera les requêtes

HTTP. Dans cet exemple, le port est configuré à 8082, ce qui signifie que l'application sera accessible à l'adresse `http://localhost:8082/`.

`spring.jpa.show-sql=true`: Cette ligne de configuration active l'affichage des requêtes SQL générées par JPA dans les logs de l'application Spring.

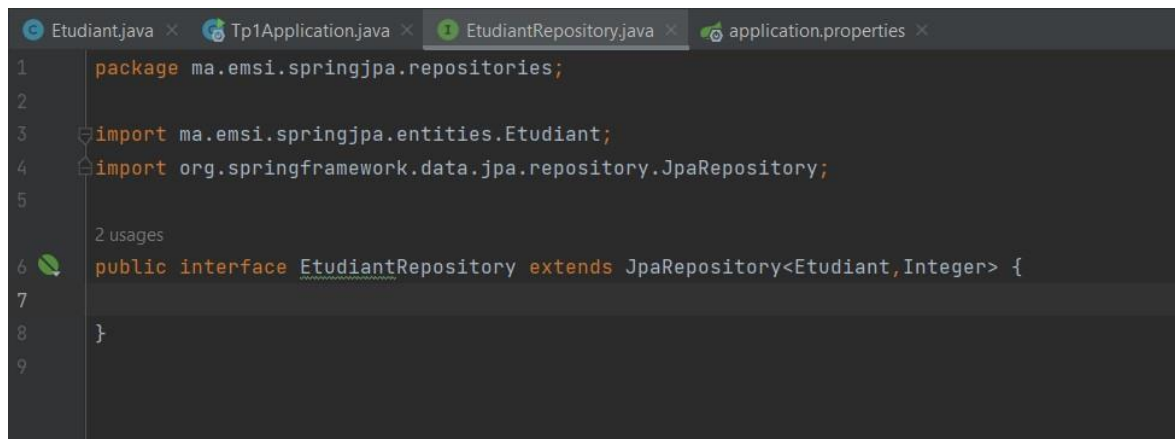
A screenshot of an IDE window showing the 'application.properties' file. The tabs at the top are 'Etudiant.java', 'Tp1Application.java', 'EtudiantRepository.java', and 'application.properties'. The code in the editor is as follows:

```
1 spring.datasource.url=jdbc:h2:mem:Etudiant-db
2 spring.h2.console.enabled=true
3 server.port=8082
4 spring.jpa.show-sql=true
5 #spring.datasource.url=jdbc:mysql://localhost:3308/Etudiant-db?createDatabaseIfNotExist=true
6 #spring.datasource.username=root
7 #spring.datasource.password=Iasnim2009
8 #spring.jpa.hibernate.ddl-auto=update
9 #spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDBDialect
10
```

3) Créer l'interface Etudiant Repository basé sur spring data.

EtudiantRepository est une interface qui hérite JpaRepository du framework Spring Data JPA. Elle est utilisée pour définir les opérations de persistance (CRUD) (Create, Read, Update, Delete) spécifiques à l'entité Etudiant dans la base de données.

Dans le cas de EtudiantRepository, en héritant JpaRepository<Etudiant,Integer>, On spécifie deux champs : Etudiant c'est le nom de la classe et Integer c'est le type de la clé primaire, les méthodes prédéfinies pour la persistance des données sont automatiquement générées pour l'entité Etudiant, avec les opérations courantes telles que l'ajout, la recherche, la mise à jour et la suppression d'étudiants dans la base de données.



```
1 package ma.emsi.springjpa.repositories;
2
3 import ma.emsi.springjpa.entities.Etudiant;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 2 usages
7 public interface EtudiantRepository extends JpaRepository<Etudiant,Integer> {
8
9 }
```

4) Tester l'application avec des opération d'ajout de consultation de mise a jour et de suppression des étudiants.

La classe Tp1Application implémente l'interface CommandLineRunner dans le but d'exécuter du code au démarrage de l'application Spring Boot.

Elle utilise l'annotation @Autowired pour injecter une instance de EtudiantRepository, qui est une interface étendue de JpaRepository pour la persistance des données de l'entité Etudiant dans la base de données.

La méthode main est la méthode principale de l'application qui est exécutée au démarrage. Elle utilise SpringApplication.run pour démarrer l'application Spring Boot.

La méthode run est la méthode qui sera exécutée au démarrage de l'application en implémentant l'interface CommandLineRunner. Dans cette méthode, des objets de type Etudiant sont créés et insérés dans la base de données en utilisant la méthode save de EtudiantRepository. Chaque objet représente un étudiant avec un nom, une date de naissance et un numéro CNE (Code National d'Etudiant) différent.

```

Etudiant.java x Tp1Application.java x EtudiantRepository.java x application.properties x
12 @SpringBootApplication
13 public class Tp1Application implements CommandLineRunner {
14     6 usages
15     @Autowired
16     private EtudiantRepository etudiantRepository;
17
18     no usages
19     public static void main(String[] args) {
20         SpringApplication.run(Tp1Application.class, args);
21     }
22
23     @Override
24     public void run(String... args) throws Exception {
25         System.out.println("***** Insertion *****");
26         etudiantRepository.save(new Etudiant( id: null, name: "akram", new Date(), Cne: 24566));
27         etudiantRepository.save(new Etudiant( id: null, name: "ahmed", new Date(), Cne: 24567));
28         etudiantRepository.save(new Etudiant( id: null, name: "yasser", new Date(), Cne: 24568));
29         etudiantRepository.save(new Etudiant( id: null, name: "ilyass", new Date(), Cne: 24569));
30         etudiantRepository.save(new Etudiant( id: null, name: "yassine", new Date(), Cne: 24565));
31         System.out.println("***** Inserted rows *****");
32         System.out.println("Count: " + etudiantRepository.count());
33     }
34 }

```

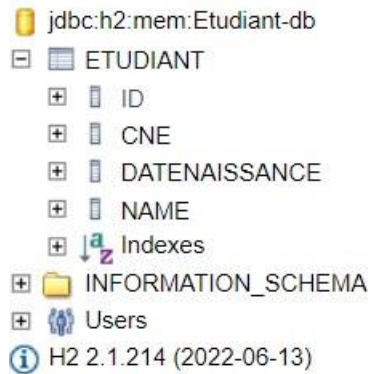
```

2023-04-08T14:31:12.985Z INFO 74640 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2023-04-08T14:31:12.994Z INFO 74640 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-04-08T14:31:13.253Z WARN 74640 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may run longer than you want. Read https://github.com/spring-projects/spring-data-jpa/issues/1501 for details.
2023-04-08T14:31:13.568Z INFO 74640 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-04-08T14:31:13.616Z INFO 74640 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8082 (http) with context path ''
2023-04-08T14:31:13.626Z INFO 74640 --- [ restartedMain] ma.ensi.springjpa.Tp1Application : Started Tp1Application in 4.245 seconds (process running for 5.208)

***** Insertion *****
Hibernate: insert into etudiant (id, cne, datenaissance, name) values (default, ?, ?, ?)
Hibernate: insert into etudiant (id, cne, datenaissance, name) values (default, ?, ?, ?)
Hibernate: insert into etudiant (id, cne, datenaissance, name) values (default, ?, ?, ?)
Hibernate: insert into etudiant (id, cne, datenaissance, name) values (default, ?, ?, ?)
Hibernate: insert into etudiant (id, cne, datenaissance, name) values (default, ?, ?, ?)
***** Inserted rows *****
Hibernate: select count(*) from etudiant e1_0
Count:5

```

Dans la console H2 on peut consulter la liste des étudiants



Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM ETUDIANT

ID	CNE	DATENAISSANCE	NAME
1	24566	2023-04-08	akram
2	24567	2023-04-08	ahmed
3	24568	2023-04-08	yasser
4	24569	2023-04-08	ilyass
5	24565	2023-04-08	yassine

(5 rows, 2 ms)

Edit

```
System.out.println("***** Display rows *****");
List<Etudiant> etudiants= etudiantRepository.findAll();
etudiants.forEach(etudiant -> {System.out.println(etudiant.toString());});
```

```
***** Display rows *****
Hibernate: select e1_0.id,e1_0.cne,e1_0.datenaissance,e1_0.name from etudiant e1_0
Etudiant(id=1, name=akram, datenaissance=2023-04-08, Cne=24566)
Etudiant(id=2, name=ahmed, datenaissance=2023-04-08, Cne=24567)
Etudiant(id=3, name=yasser, datenaissance=2023-04-08, Cne=24568)
Etudiant(id=4, name=ilyass, datenaissance=2023-04-08, Cne=24569)
Etudiant(id=5, name=yassine, datenaissance=2023-04-08, Cne=24565)
```


jdbc:h2:mem:Etudiant-db

- ETUDIANT
- INFORMATION_SCHEMA
- Users
- H2 2.1.214 (2022-06-13)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM ETUDIANT

SELECT * FROM ETUDIANT;

ID	CNE	DATENAISSANCE	NAME
1	24566	2023-04-08	akram
2	24567	2023-04-08	ahmed
3	24568	2023-04-08	yasser
4	24569	2023-04-08	ilyass
5	24565	2023-04-08	yassine

(5 rows, 0 ms)

Edit

Pour modifier l'étudiant ayant l'id 3 il faut suivre les «étapes suivantes :

On utilise la méthode `findById` de `EtudiantRepository` pour rechercher un étudiant dans la base de données avec l'identifiant 3.

Si un étudiant est trouvé, il est stocké dans un objet `Etudiant` appelé `etudiant`.

Ensuite, la méthode `setCne` est utilisée pour définir un nouveau numéro CNE (Code National d'Etudiant) sur l'objet `etudiant`, en lui attribuant la valeur 44444.

Enfin, la méthode `save` de `EtudiantRepository` est appelée pour mettre à jour les informations de l'étudiant dans la base de données avec le nouveau numéro CNE.

```
System.out.println("***** Update *****");
Etudiant etudiant=etudiantRepository.findById(3).orElse( other: null);
etudiant.setCne(44444);
etudiantRepository.save(etudiant);
```

CNE de l'étudiant 3 nommé yasser est modifié (24568 -> 44444)

jdbc:h2:mem:Etudiant-db

ETUDIANT

INFORMATION_SCHEMA

Users

H2 2.1.214 (2022-06-13)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM ETUDIANT

SELECT * FROM ETUDIANT;

ID	CNE	DATENAISSANCE	NAME
1	24566	2023-04-08	akram
2	24567	2023-04-08	ahmed
3	44444	2023-04-08	yasser
4	24569	2023-04-08	ilyass
5	24565	2023-04-08	yassine

(5 rows, 1 ms)

Edit

Pour supprimer l'étudiant ayant l'id 5 il faut suivre les étapes suivantes :

On utilise la méthode `findById` de `EtudiantRepository` pour rechercher un étudiant dans la base de données avec l'identifiant 5.

Si un étudiant est trouvé, il est stocké dans un objet `Etudiant` appelé `etudiant1`.

Ensuite, la méthode `count` de `EtudiantRepository` est utilisée pour obtenir le nombre d'étudiants dans la base de données avant la suppression de l'étudiant.

La méthode `delete` de `EtudiantRepository` est appelée pour supprimer l'objet `etudiant1` de la base de données.

Enfin, la méthode `count` de `EtudiantRepository` est à nouveau utilisée pour obtenir le nombre d'étudiants dans la base de données après la suppression de l'étudiant.

La valeur du nombre d'étudiants avant et après la suppression est affichée dans la console à l'aide de la méthode `System.out.println`.

```
System.out.println("***** Delete *****");
Etudiant etudiant1=etudiantRepository.findById(5).orElse( other: null);
System.out.println("AVANT Count:" +etudiantRepository.count());
etudiantRepository.delete(etudiant1);
System.out.println("APRES Count:" +etudiantRepository.count());
```

```

AVANT Count:5
Hibernate: select e1_0.id,e1_0.cne,e1_0.datenaissance,e1_0.name from etudiant e1_0 where e1_0.id=?
Hibernate: delete from etudiant where id=?
Hibernate: select count(*) from etudiant e1_0
APRES Count:4

```

L'étudiant 5 baptisé yassine est supprimé

The screenshot shows a database client interface. On the left, a tree view displays the database structure: 'jdbc:h2:mem:Etudiant-db' containing 'ETUDIANT', 'INFORMATION_SCHEMA', 'Users', and 'H2 2.1.214 (2022-06-13)'. The main area shows the 'SQL statement:' field with 'SELECT * FROM ETUDIANT'. Below this, the query results are displayed as a table with 4 rows and 4 columns: ID, CNE, DATENAISSANCE, and NAME. The rows contain data for students with IDs 1, 2, 3, and 4. Below the table, it indicates '(4 rows, 1 ms)' and there is an 'Edit' button.

ID	CNE	DATENAISSANCE	NAME
1	24566	2023-04-08	akram
2	24567	2023-04-08	ahmed
3	44444	2023-04-08	yasser
4	24569	2023-04-08	ilyass

5) Utiliser une base de donnée Mysql au lieu de h2.

The screenshot shows an IDE with the 'application.properties' file open. The file contains the following configuration for a MySQL database:

```

1 #spring.datasource.url=jdbc:h2:mem:Etudiant-db
2 #spring.h2.console.enabled=true
3 spring.datasource.url=jdbc:mysql://localhost:3308/Etudiant-db?createDatabaseIfNotExist=true
4 spring.datasource.username=root
5 spring.datasource.password=Tasnim2009
6 server.port=8082
7 spring.jpa.show-sql=true
8 spring.jpa.hibernate.ddl-auto=update
9 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDBDialect
10

```

The left sidebar shows the project structure with folders like 'main', 'resources', 'test', and 'target'.

```

<!--      <dependency>-->
<!--          <groupId>com.h2database</groupId>-->
<!--          <artifactId>h2</artifactId>-->
<!--          <scope>runtime</scope>-->
<!--      </dependency>-->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.29</version>
</dependency>

```

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

XAMPP Control Panel v3.3.0

Modules	Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	31800 77376	4433, 8081	Stop Admin Config Logs	
<input type="checkbox"/>	MySQL	50064	3308	Stop Admin Config Logs	
<input type="checkbox"/>	FileZilla			Start Admin Config Logs	
<input type="checkbox"/>	Mercury			Start Admin Config Logs	
<input type="checkbox"/>	Tomcat			Start Admin Config Logs	

15:14:28 [mysql] improper privileges, a crash, or a shutdown by another method.
 15:14:28 [mysql] Press the Logs button to view error logs and check
 15:14:28 [mysql] the Windows Event Viewer for more clues
 15:14:28 [mysql] If you need more help, copy and post this
 15:14:28 [mysql] entire log window on the forums
 15:14:33 [mysql] Attempting to start MySQL app...
 15:14:33 [mysql] Status change detected: running

Config

Netstat

Shell

Explorer

Services

Help

Quit



<

```
System.out.println("***** Mysql *****");
etudiantRepository.save(new Etudiant( id: null, name: "lkihal", new Date(), Cne: 983467));
```


				id	cne	datenaissance	name
<input type="checkbox"/>		Éditer		Copier		Supprimer	1 24566 2023-04-08 akram
<input type="checkbox"/>		Éditer		Copier		Supprimer	2 24567 2023-04-08 ahmed
<input type="checkbox"/>		Éditer		Copier		Supprimer	3 44444 2023-04-08 yasser
<input type="checkbox"/>		Éditer		Copier		Supprimer	4 24569 2023-04-08 ilyass
<input type="checkbox"/>		Éditer		Copier		Supprimer	8 983467 2023-04-08 lkihal

```

System.out.println("***** Mysql *****");
etudiantRepository.save(new Etudiant( id: null, name: "lkihal", new Date(), Cne: 983467));
System.out.println("***** Update *****");
Etudiant etudiant=etudiantRepository.findById(4).orElse( other: null);
etudiant.setName("amri");
etudiantRepository.save(etudiant);
System.out.println("***** Delete *****");
Etudiant etudiant1=etudiantRepository.findById(1).orElse( other: null);
System.out.println("AVANT Count: " +etudiantRepository.count());
etudiantRepository.delete(etudiant1);
System.out.println("APRES Count: " +etudiantRepository.count());

```

				id	cne	datenaissance	name
<input type="checkbox"/>		Éditer		Copier		Supprimer	2 24567 2023-04-08 ahmed
<input type="checkbox"/>		Éditer		Copier		Supprimer	3 44444 2023-04-08 yasser
<input type="checkbox"/>		Éditer		Copier		Supprimer	4 24569 2023-04-08 amri
<input type="checkbox"/>		Éditer		Copier		Supprimer	8 983467 2023-04-08 lkihal

```

Hibernate: select e1_0.id,e1_0.cne,e1_0.datenaissance,e1_0.name from etudiant e1_0 where e1_0.id=?
Hibernate: select count(*) from etudiant e1_0
AVANT Count:5
Hibernate: select e1_0.id,e1_0.cne,e1_0.datenaissance,e1_0.name from etudiant e1_0 where e1_0.id=?
Hibernate: delete from etudiant where id=?
Hibernate: select count(*) from etudiant e1_0
APRES Count:4

```