Spring Data JPA - Delete Records From MySQL

Last Updated : 23 Jul, 2025

Spring Boot simplifies database operations using **Spring Data JPA**, which provides built-in methods for **CRUD (Create, Read, Update, Delete)** operations. In modern application development, **data manipulation** is a critical task, and Java Persistence API (JPA) simplifies this process. **Java persistence API is like an interface that contains most of the methods that are used for data manipulation in the MySQL table and hibernate is the implementation of these methods**, So we don't have to create any method for inserting the data in the MySQL, deletion the records from the table.

Let's understand how to **delete the records from Mysql using Spring Data JPA**. The most efficient way to delete the records is with the help of the **primary key**. Because the primary key uniquely identifies each record of the table. We can use the JPA method **deleteById()** to delete the record of the particular primary key. This approach ensures efficient data management and simplifies database operations in the Spring Boot applications.

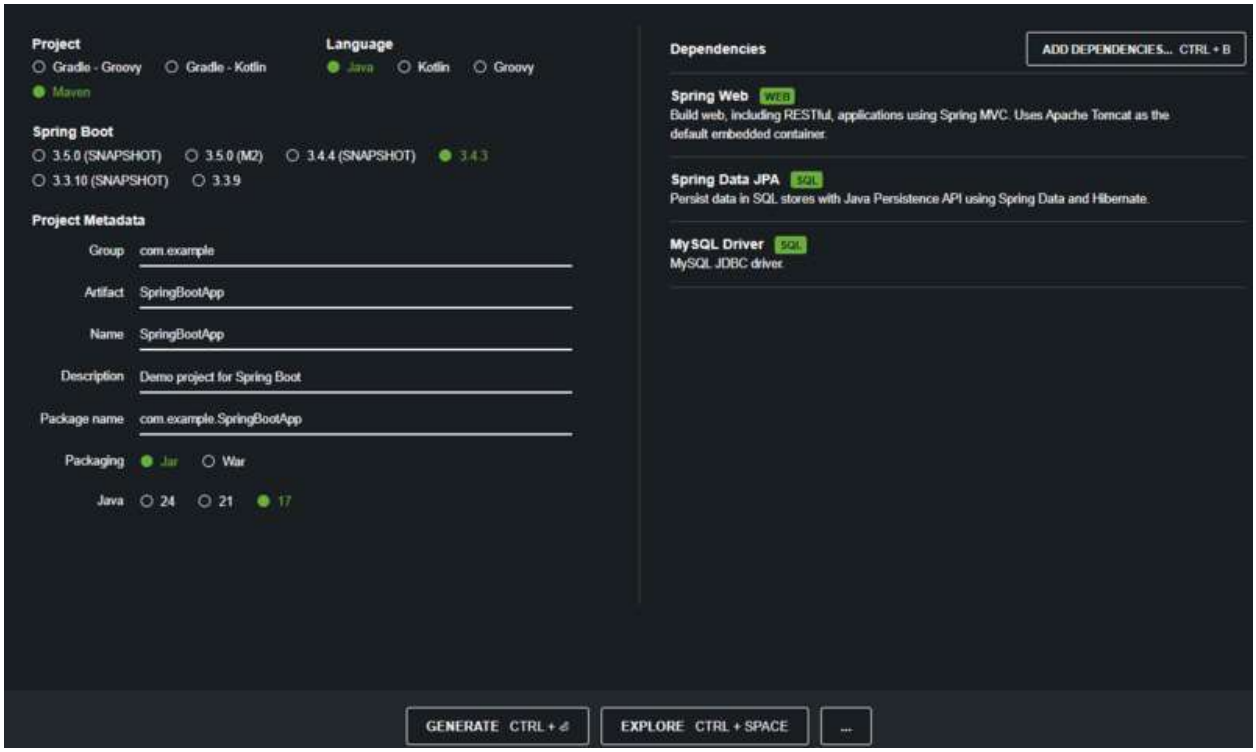
Step By Step Implementation

Step 1: Create a Spring Boot Project

- Go to Spring Initializr
- Fill in the project details:
  - **Project:** Maven Project
  - **Language:** Java
  - **Spring Boot:** 3.x.x (Latest stable version)

- **Group:** com.example
- **Artifact:** SpringBootApp
- **Name:** SpringBootApp
- **PackageName:** com.example
- **Packaging:** Jar
- **Java Version:** 17

- Add the following dependencies:
  - **Spring Data JPA**
  - **Spring Web**
  - **MYSQL Driver**
- Click on **Generate** to download the project as a ZIP file.



Step 2: Adding Dependencies

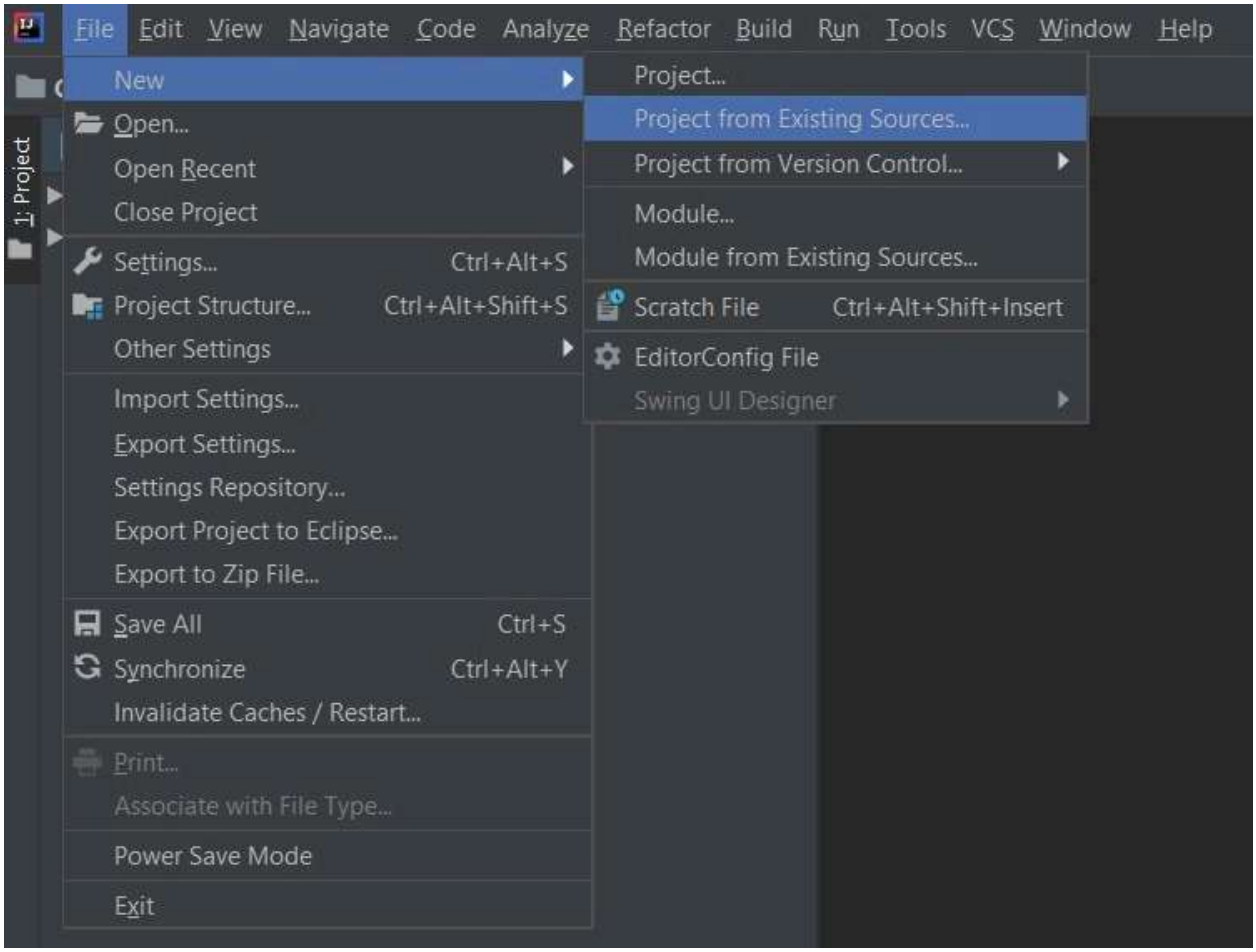
The pom.xml file contains the project dependencies and configuration.

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="https://maven.apache.org/POM/4.0.0"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
```

```
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.4.3</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>SpringBootApp</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>SpringBootApp</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Extract the zip file. Now open a suitable IDE and then go to File->New->Project from existing sources->Springbootapp and select pom.xml. Click on import changes on prompt and wait for the project to sync.



> ***Note**: In the Import Project for Maven window, make sure you choose the same version of JDK which you selected while creating the project.*

Step 3: Configure Database in (application.properties)

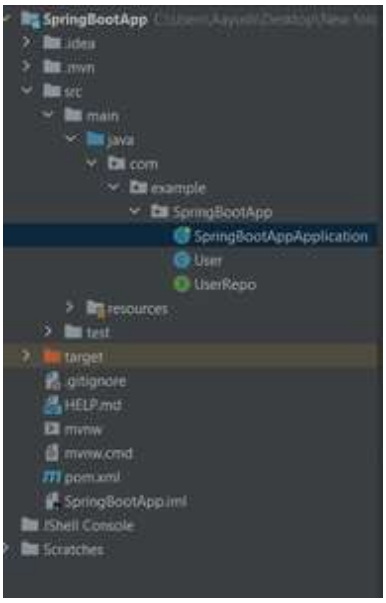
For database operation we have to configure the Spring application with the database also it is required to add configuration of the database before executing the Spring project. All the configuration is added in the file application.properties of the Spring project.

application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/user
spring.datasource.username=Your Username
spring.datasource.password=Your Password
spring.jpa.hibernate.ddl-auto=update
```

Project Structure:

The Project structure is as follow:



Step 4: Create the Entity Class

Create a User entity class that represents a table in the MySQL database.

User.java:

```
import jakarta.persistence.*;

@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    int id;
    String name;
    User() {}
    User(int id, String name)
    {
        this.id = id;
        this.name = name;
    }
}
```

Step 5: Create the Repository Interface

Create a repository interface that extends JpaRepository to perform CRUD operations.

UserRepo:

```
import org.springframework.data.jpa.repository.JpaRepository;

interface UserRepo extends JpaRepository<User,Integer> {

}
```

Step 6: Implement the Main Application Class

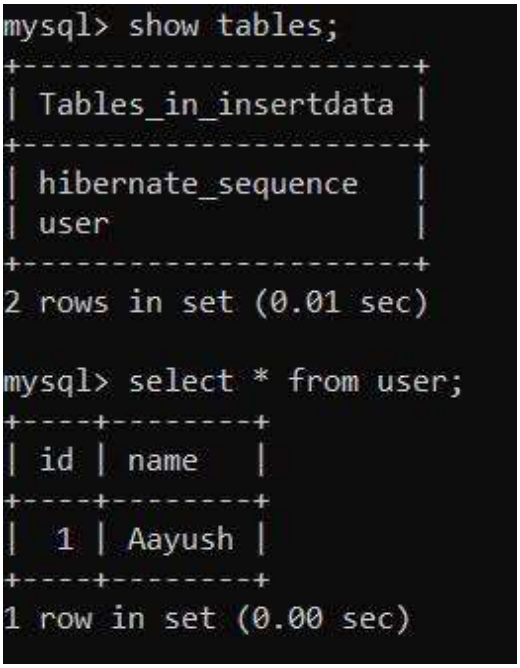
The main application class implements CommandLineRunner to execute code at startup. Here, we insert a record and then delete it using the deleteById() method.

SpringBootAppApplication.java:

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.util.Assert;

@SpringBootApplication
public class SpringBootAppApplication implements CommandLineRunner {

    @Autowired
    UserRepo ob;

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

    @Override
    public void run(String... args) throws Exception {
        // Inserting the data in the MySQL table.
        User first = new User(1, "Aayush");
         // Save the record before deleting it
        ob.save(first);
        System.out.println("Record inserted successfully.");

        // Deleting the record with id 1
        if (ob.existsById(1)) {
             // Check if the record exists
            ob.deleteById(1);
            System.out.println("Record deleted successfully.");
        } else {
            System.out.println("Record not found.");
        }
```

```
    }
}
```

Previous table output:



Step 7: Run the Application

Ensure MySQL is running and the user database exists. Run the application using the following command

mvn spring-boot:run

Output:

```
: Starting SpringBootAppApplication using Java 16.0.2 on LAPTOP-0P6DDSCR w
: No active profile set, falling back to default profiles: default
: Bootstrapping Spring Data JPA repositories in DEFAULT mode.
: Finished Spring Data repository scanning in 71 ms. Found 1 JPA repositor
: Tomcat initialized with port(s): 8080 (http)
: Starting service [Tomcat]
: Starting Servlet engine: [Apache Tomcat/9.0.55]
: Initializing Spring embedded WebApplicationContext
: Root WebApplicationContext: initialization completed in 2579 ms
: HHH000204: Processing PersistenceUnitInfo [name: default]
: HHH000412: Hibernate ORM core version 5.6.1.Final
: HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
: HikariPool-1 - Starting...
: HikariPool-1 - Start completed.
: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.trans
: Initialized JPA EntityManagerFactory for persistence unit 'default'
: spring.jpa.open-in-view is enabled by default. Therefore, database queri
: Tomcat started on port(s): 8080 (http) with context path ''
: Started SpringBootAppApplication in 7.005 seconds (JVM running for 7.627
```

Database Output:

```
mysql> select * from user;
Empty set (0.00 sec)

mysql>
```

Comment | More info | Advertise with us

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305