Spring Data JPA - @Column Annotation

Last Updated : 18 Mar, 2025

In **Spring Data JPA**, the **@Column annotation** is used to define column-specific attributes for an entity field in the database. It allows developers to customize column names, set length, define nullability, and more. This is essential when working with relational databases like **MySQL, PostgreSQL, and Oracle in Spring Boot applications**. In this article, we will explore **how to use @Column annotation to map entity fields to database columns** with a step-by-step implementation.

Syntax of @Column Annotation

```
@Column(name = "description", nullable = false, length = 512)
private String description;
```

Attributes of @Column:

- **name**: Specifies the column name in the database.
- **length**: Defines the maximum column length.
- **nullable**: Determines if the column allows NULL values.

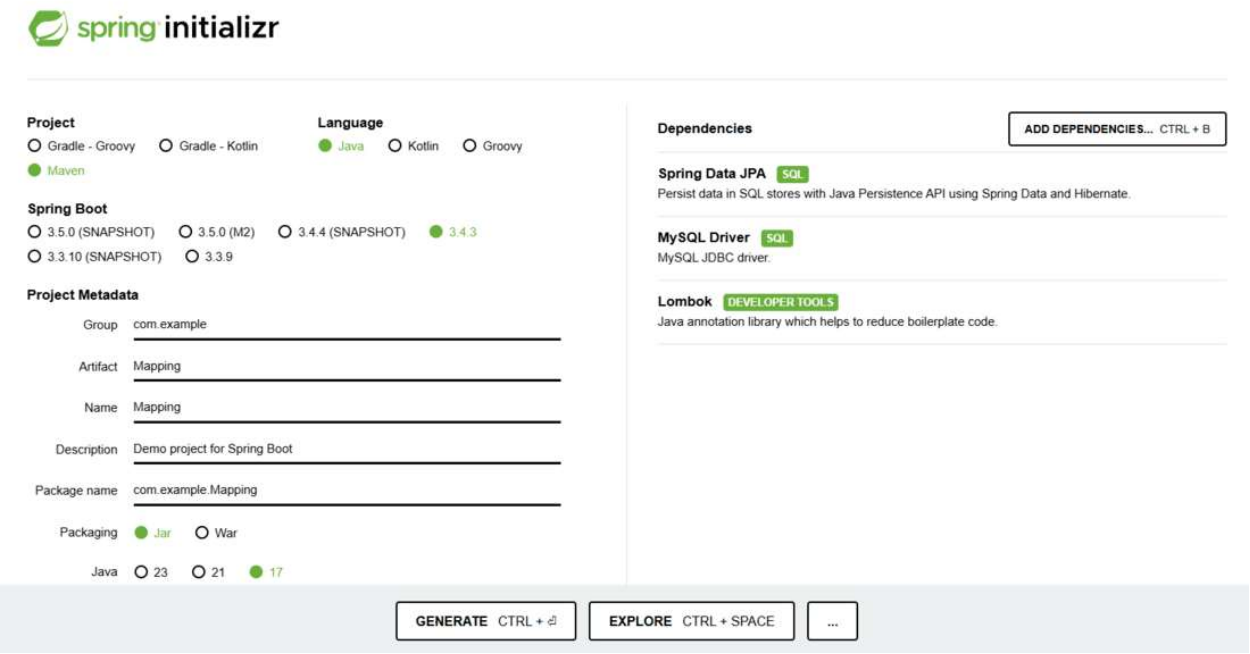
Step-by-Step Implementation

Step 1: Create a Spring Boot Project

Go to Spring Initializr and generate a new project with the following configuration:

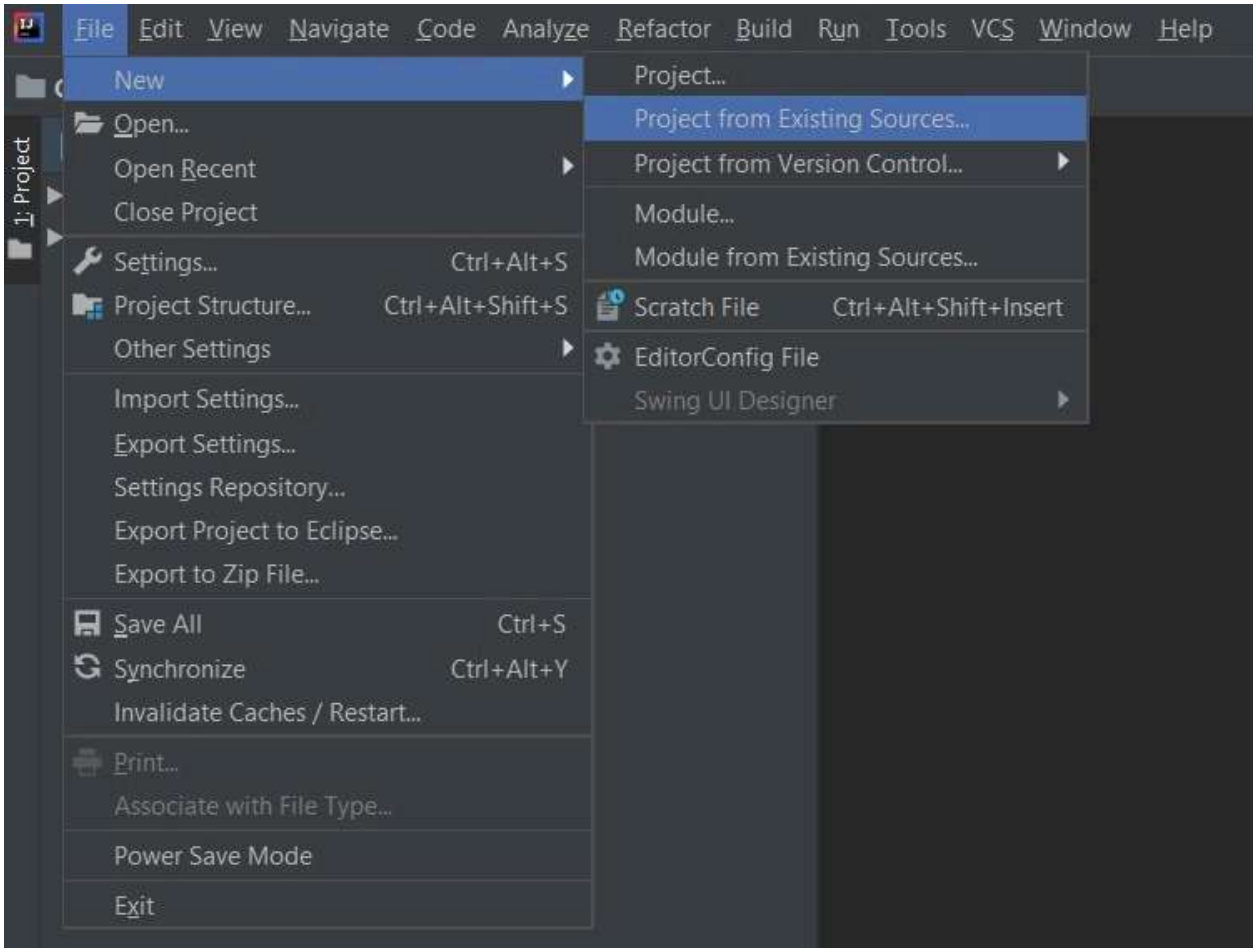
- **Project**: Maven
- **Language**: Java

- **Spring Boot Version**: 3.x (Latest LTS)
- **Dependencies**:
  - Spring Data JPA
  - MySQL Driver
  - Lombok



Click on Generate which will download the starter project.

Extract the zip file. Now open a suitable IDE and then go to **File > New > Project from Existing Sources** and select pom.xml. Click on import changes on prompt and wait for the project to sync as pictorially depicted below as follows:

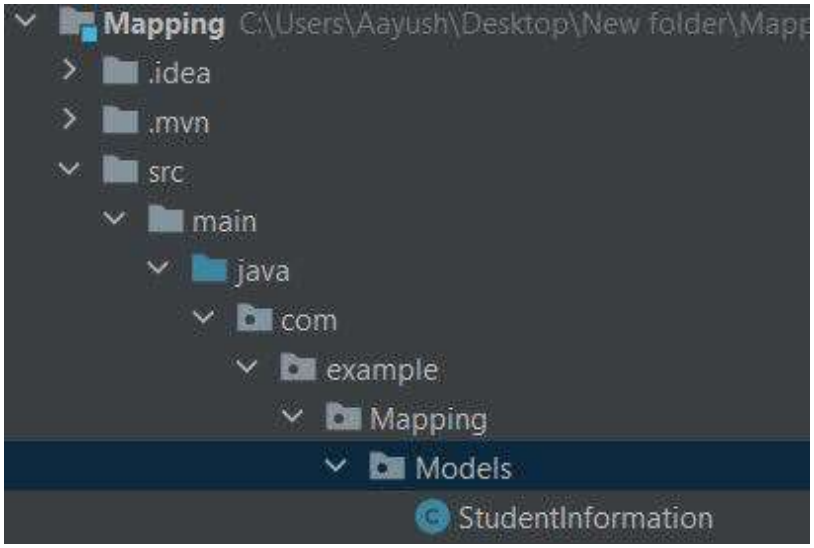


Step 2: Configure Database in application.properties

Adding the necessary properties in the **application.properties** file. (mapping is the database name)

```
spring.datasource.url=jdbc:mysql://localhost:3306/mapping
spring.datasource.username=root
spring.datasource.password=yourpassword
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Project Structrure:



Step 3: Create the Entity Class

Inside **src/main/java/com/example/model/,** create **StudentInformation.java**:

```
import jakarta.persistence.*;
import lombok.*;

@Entity
@Table(name = "student")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class StudentInformation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int rollno;

    @Column(name = "name", nullable = true, length = 255)
    private String name;

    @Column(name = "student_name", nullable = true, length = 255)
    private String studentName;
}
```

Step 4: Run the Application

Run the main class using your IDE or command line:

```
: 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.transacti
: Initialized JPA EntityManagerFactory for persistence unit 'default'
: spring.jpa.open-in-view is enabled by default. Therefore, database queries m
: Tomcat started on port(s): 8080 (http) with context path ''
: Started MappingApplication in 6.117 seconds (JVM running for 6.762)
```

Spring Boot will automatically generate the table Student with the mapped columns in MySQL.

Step 5: Verify the Database Table

Execute the following SQL query in MySQL to check the table structure:

DESC Student;

```
mysql> use mapping;
Database changed
mysql> show tables;
+-------------------+
| Tables_in_mapping |
+-------------------+
| student           |
+-------------------+
1 row in set (0.01 sec)

mysql> desc student;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| rollno       | int          | NO   | PRI | NULL    | auto_increment |
| name         | varchar(255) | YES  |     | NULL    |                |
| student_name | varchar(255) | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```