Spring Data JPA - @Id Annotation

Last Updated : 19 Mar, 2025

Spring Data JPA is a important part of Spring Boot applications, providing an abstraction over **JPA** (Java Persistence API) and simplifying database interactions. JPA is a specification that defines a standard way to interact with relational databases in Java, while Hibernate is one of the most widely used implementations of JPA. One of the essential annotations in JPA is **@Id,** which is used to mark a field as the primary key of an entity.

In this article, we will explore the **@Id annotation in Spring Data JPA**, how it works, and how to use it in a Spring Boot application with Spring Boot, MySQL, and Hibernate ORM.

Use of @Id Annotation in JPA

The **@Id annotation** is part of **javax.persistence.Id,** which marks a field as the primary key of an entity. It is used by Hibernate and Spring Data JPA for object-relational mapping. When combined with @GeneratedValue, it enables automatic ID generation.

Syntax:

```
import jakarta.persistence.*;

@Entity
public class ExampleEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
}
```

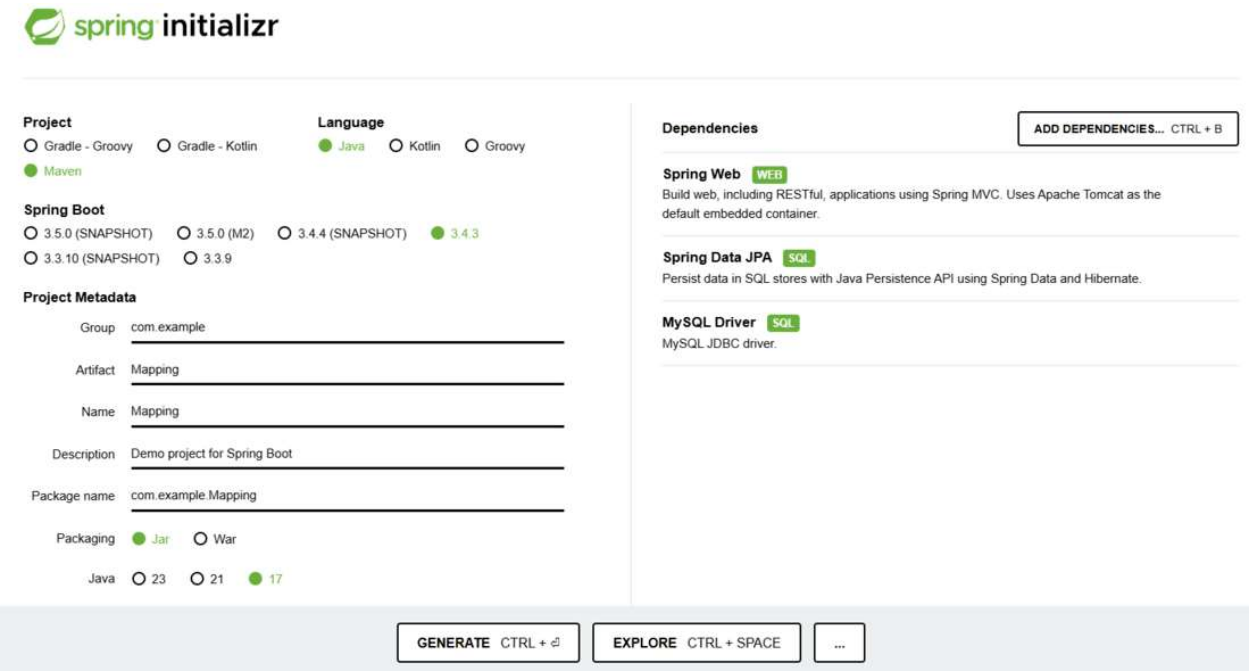
Creating a Spring Boot Application with @Id Annotation

We will now create a Spring Boot application that demonstrates how to use @Id with Spring Data JPA and MySQL.

Step 1: Create a Spring Boot Project

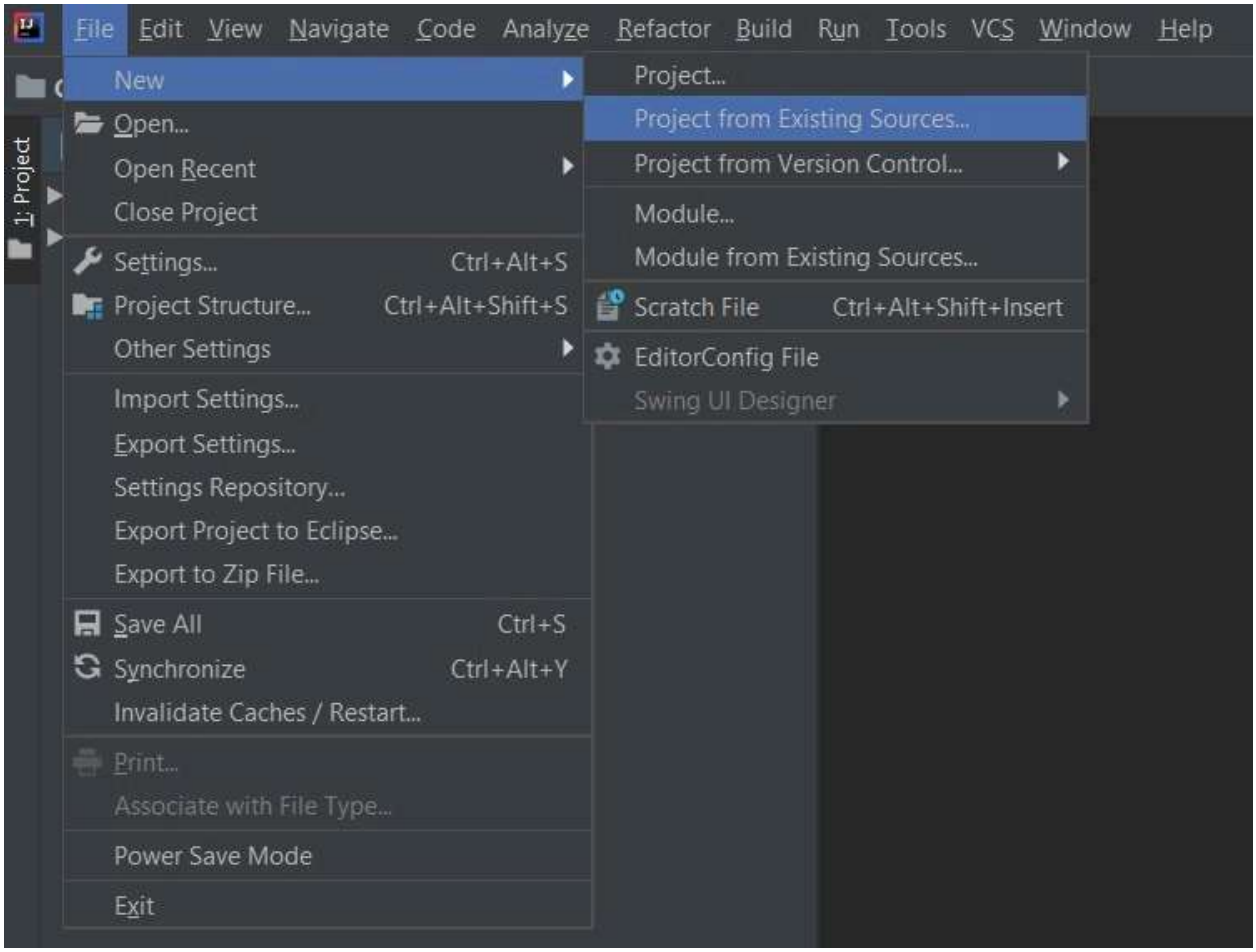
Go to Spring Initializr and configure the project with the following:

- **Project**: Maven
- **Language**: Java
- **Spring Boot Version**: 3.x (Latest stable version)
- **Dependencies**: Spring Web, Spring Data JPA, MySQL Driver



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:

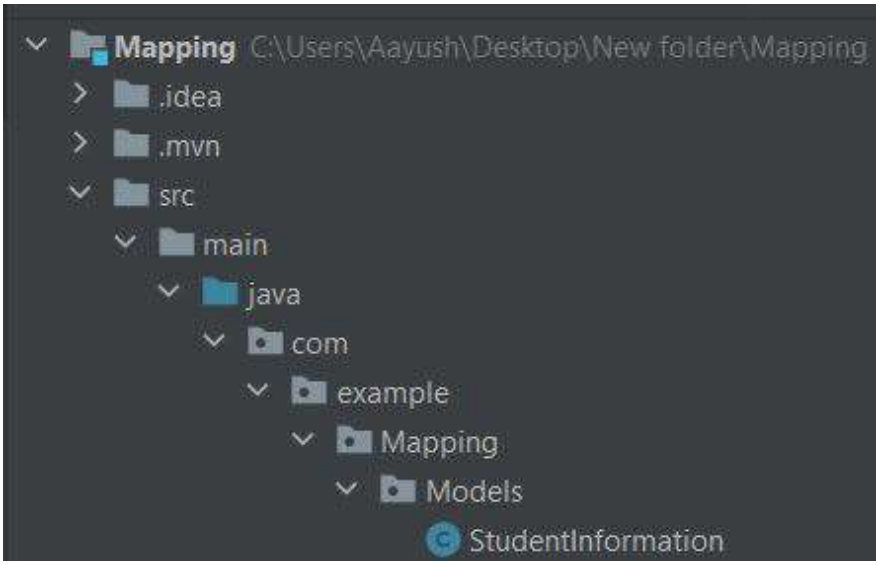


Step 2: Configure application.properties

In **src/main/resources/application.properties**, add the database connection settings. (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.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
```

Project Structure:



Step 3: Create the Entity Class

Create a new file **StudentInformation.java** in **src/main/java/com/example/mapping/.**

```
package com.example.mapping;

import jakarta.persistence.*;

@Entity
@Table(name = "Student")
public class StudentInformation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int rollno;
    private String name;

    public StudentInformation() {}

    public StudentInformation(int rollno, String name) {
        this.rollno = rollno;
        this.name = name;
    }

    public int getRollno() { return rollno; }
    public void setRollno(int rollno) { this.rollno = rollno; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

Step 4: Create the Repository Interface

Create **StudentRepository.java** in **src/main/java/com/example/mapping/.**

```
package com.example.mapping;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface StudentRepository extends JpaRepository<StudentInformation,
Integer> {
}
```

Step 5: Create the Main Application Class

Create **MappingApplication.java** in **src/main/java/com/example/mapping/.**

```
package com.example.mapping;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MappingApplication {
    public static void main(String[] args) {
        SpringApplication.run(MappingApplication.class, args);
    }
}
```

Running the Application

Terminal Output:

```
: 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)
```

Database Output:

- Start MySQL Server
- Open command prompt and run "**mysql -u root -p**". Enter your MySQL password when prompted.
- Select the database
- Check if the Table Exists

```
mysql> create database mapping;
Query OK, 1 row affected (0.01 sec)

mysql> use mapping;
Database changed
mysql> show tables;
+-------------------+
| Tables_in_mapping |
+-------------------+
| student           |
+-------------------+
1 row in set (0.00 sec)

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