Spring Data JPA - @Table Annotation

Last Updated : 28 Mar, 2025

Spring Data JPA is a powerful framework that simplifies database interactions in Spring Boot applications. The **@Table annotation in JPA** (Java Persistence API) is used to specify the table name in the database and ensure proper mapping between Java entities and database tables. This is especially useful when:

- The database table name differs from the entity class name.
- We need to ensure uniqueness across multiple columns.
- Working with multi-schema databases (for example, legacy systems).

In this article, we will explore **how to use the @Table annotation in Spring Data JPA with an example**.

@Table Annotation

The @Table annotation in JPA is used to define the database table mapping for an entity. It allows customization of:

- Table name (default is the entity class name)
- Catalog and schema (useful for multi-database environments)
- Unique constraints on specific columns

Syntax:

```
import jakarta.persistence.*;

@Entity
@Table(name = "student")       // Custom table name
public class Student {
    // Fields and methods
}
```

In this example, the table is explicitly named "student" instead of relying on the default entity name.

Attributes of @Table Annotation

The @Table annotation provides the following attributes:

Attribute	Description
name	It defines the table name (default: entity class name).
catalog	It specifies the database catalog.
schema	It defines the database schema.
uniqueConstraints	It enforces unique constraints on specific columns.

Example of @Table with Unique Constraint

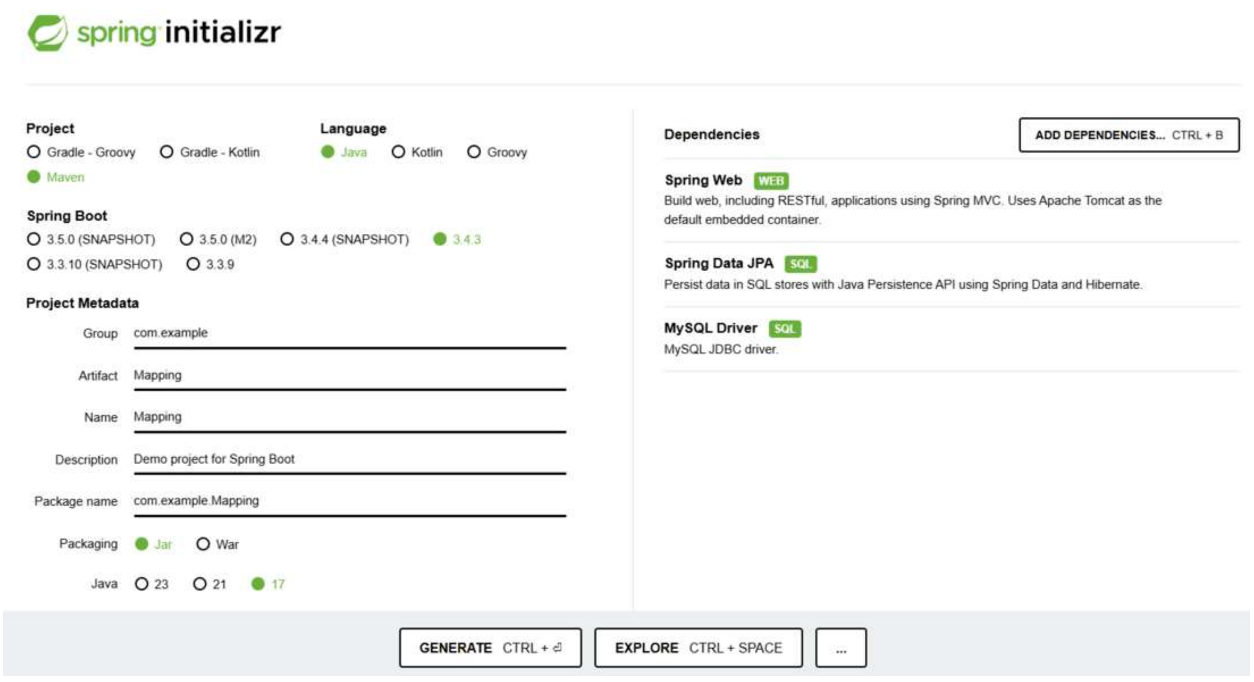
```
@Entity
@Table(name = "EMPLOYEE", uniqueConstraints = {
@UniqueConstraint(columnNames = "email") })
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(nullable = false, unique = true)
    private String email;
}
```

In this example, the email field must be unique across all rows in the EMPLOYEE table.

Step-by-Step Implementation

Step 1: Create a Spring Boot Project

- Go to Spring Initializr
- Fill in the details:
  - **Project:** Maven
  - **Language**: Java
  - **Spring Boot Version**: 3.x.x (or latest stable version)
  - **Packaging**: JAR
  - **Java Version**: 17 or later
  - **Dependencies**: Spring Web, Spring Data JPA, MySQL Driver



Project: Gradle - Groovy, Gradle - Kotlin, [✓ Maven]

Language: [✓ Java], Kotlin, Groovy

Spring Boot: 3.5.0 (SNAPSHOT), 3.5.0 (M2), 3.4.4 (SNAPSHOT), [✓ 3.4.3], 3.3.10 (SNAPSHOT), 3.3.9

Project Metadata

Group	com.example
Artifact	Mapping
Name	Mapping
Description	Demo project for Spring Boot
Package name	com.example.Mapping
Packaging	[✓ Jar] War
Java	23 21 [✓ 17]

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

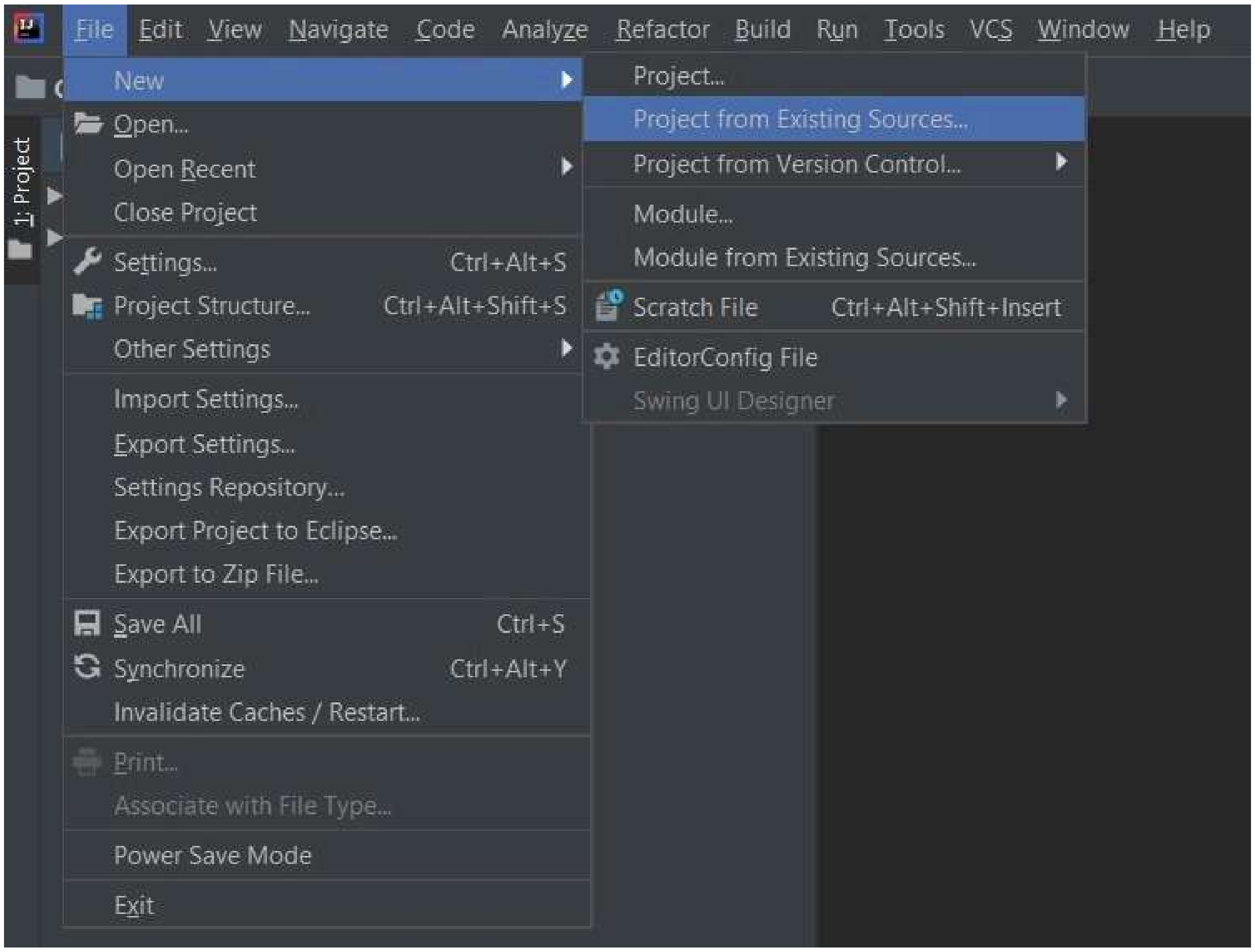
Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.

Click on Generate, download, and extract the project.

Step 2: Import the Project into Your IDE

- Extract the zip file.
- Open your preferred IDE (IntelliJ, Eclipse, VS Code). Here, we are using IntelliJ IDE.
- Now open a suitable IDE and then go to **File > New > Project from Existing Sources** and select pom.xml.
- Ensure `pom.xml` is recognized and dependencies are downloaded.



Step 3: Configure Database 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=your_password *spring.jpa.hibernate.ddl-auto=update*

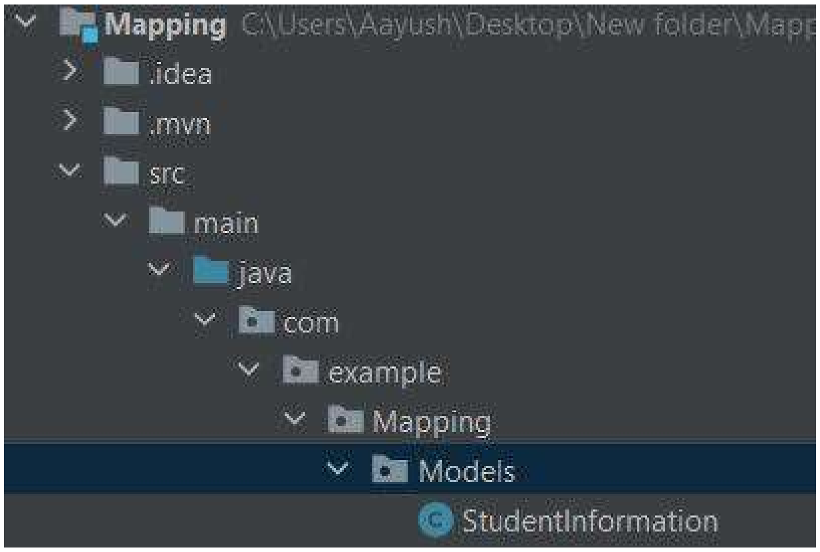
Best Practice: Avoid hardcoding credentials. Use environment variables:

```
spring:
  datasource:
    username: ${DB_USER}
    password: ${DB_PASS}
```

Step 4: Create the Entity Class

Project Structure:

Create a model folder in the project folder and make a StudentInformation class.



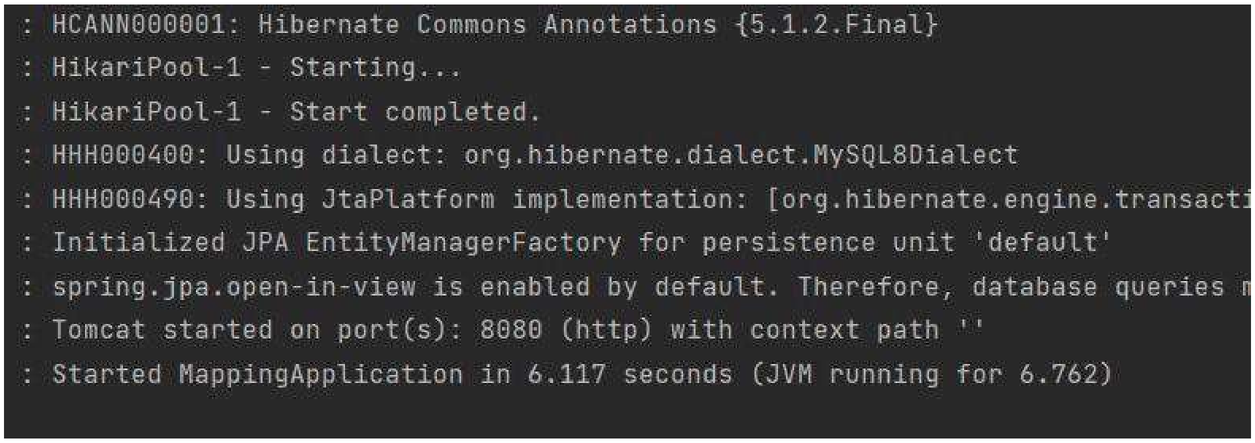
```
Mapping C:\Users\Aayush\Desktop\New folder\Mapp
  .idea
  .mvn
  src
    main
      java
        com
          example
            Mapping
              Models
                StudentInformation
```

StudentInformation.java:

```
package com.example.mapping.models;

import jakarta.persistence.*;

@Entity
@Table(name = "student")     // Custom table name
public class StudentInformation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int rollno;

    private String name;

    // Default constructor
    public StudentInformation() {}

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

    // Getters and Setters
    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;

    }
}
```

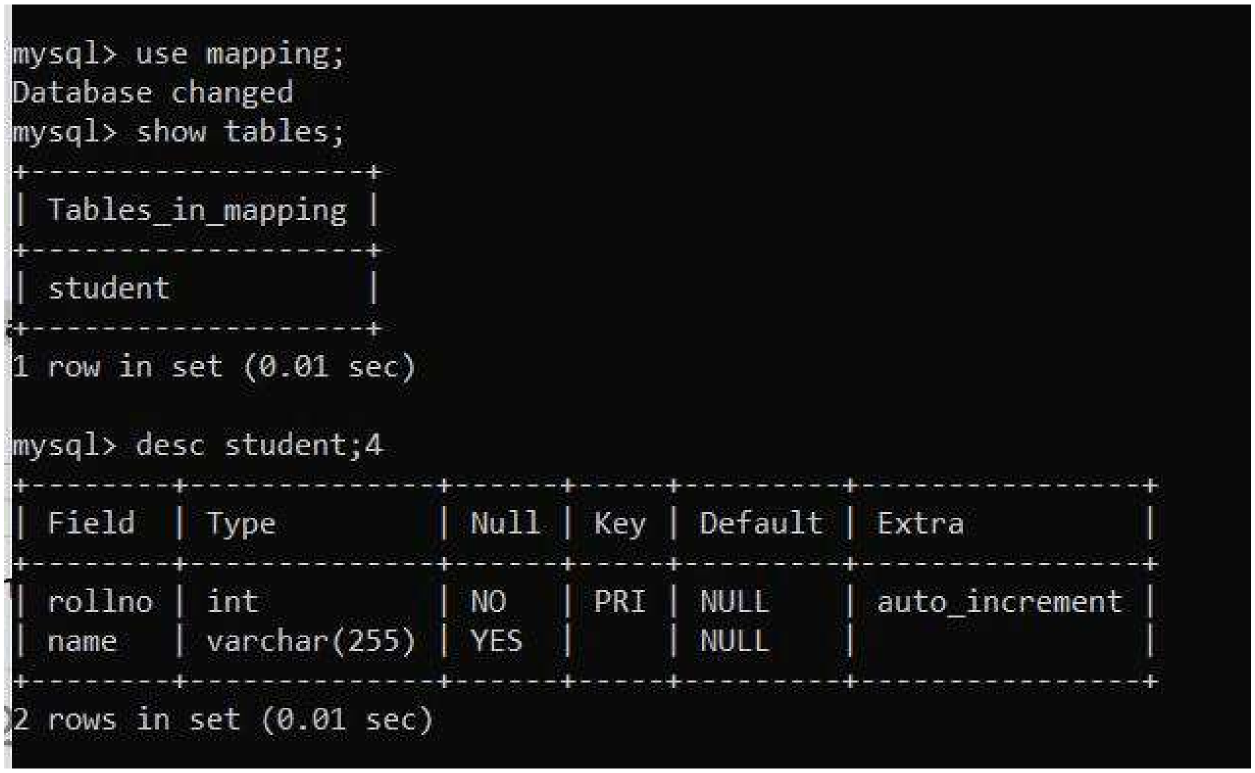
Step 5: Running the Application

Run the main application class.



```
: 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:



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

mysql> desc student;4
+--------+--------------+------+-----+---------+----------------+
| 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)
```