

Search...

[Advance Java Course](#)[Java Tutorial](#)[Java Spring](#)[Spring Interview Questions](#)[Java SpringBoot](#)[Sprin](#)

# Spring Data JPA - Insert Data in MySQL Table

Last Updated : 23 Jul, 2025

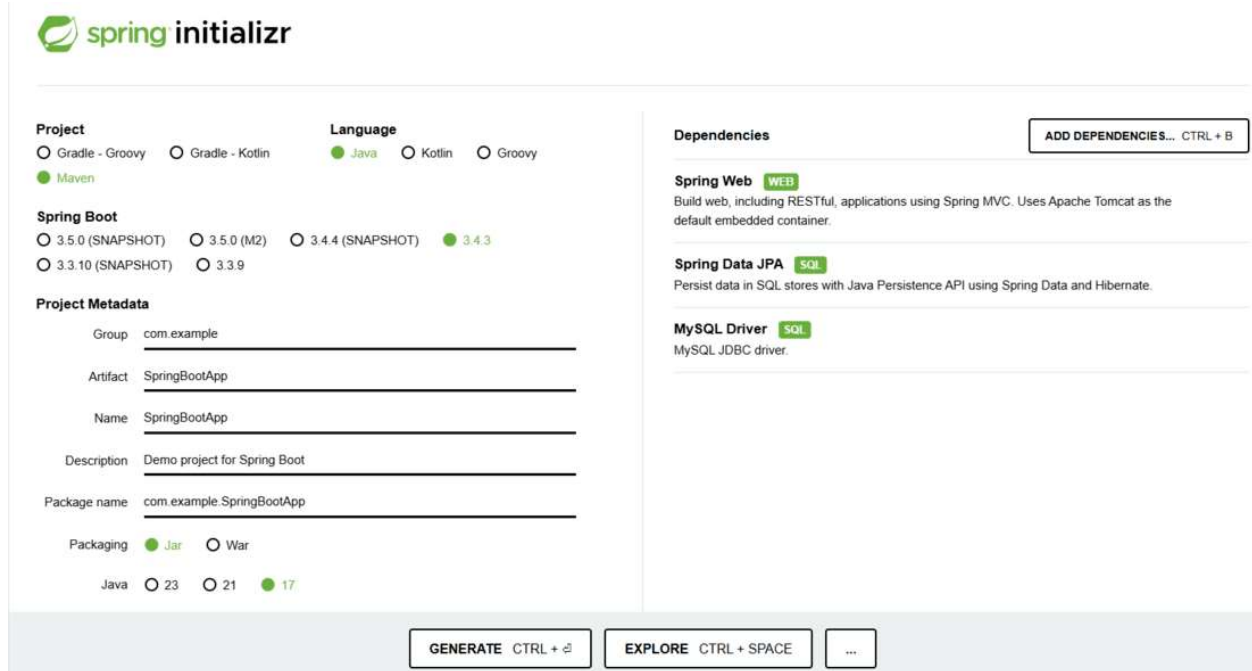
**Spring Data JPA** makes it easy to work with databases in Spring Boot by reducing the need for boilerplate code. It provides built-in methods to perform operations like inserting, updating, and deleting records in a MySQL table. In this article, we will see how to insert data into a MySQL database using Spring Boot, Spring Data JPA, and Hibernate with the **save()** method of **JpaRepository**.

*JpaRepository<>.save() method is used for inserting the values in the MySQL table.*

## Step By Step Implementation

### Step 1: Create a Spring Boot Project

- Go to [Spring Initializr](#).
- Select Spring Boot Version 3.x.
- Add the following dependencies:
  - Spring Web
  - Spring Data JPA
  - MySQL Driver



The image shows the Spring Initializr web interface. It is divided into several sections:

- Project:** Includes radio buttons for **Gradle - Groovy**, **Gradle - Kotlin**, and **Maven** (selected).
- Language:** Includes radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot:** Includes radio buttons for **3.5.0 (SNAPSHOT)**, **3.5.0 (M2)**, **3.4.4 (SNAPSHOT)**, **3.4.3** (selected), and **3.3.10 (SNAPSHOT)**, **3.3.9**.
- Project Metadata:** Includes fields for **Group** (com.example), **Artifact** (SpringBootApplication), **Name** (SpringBootApplication), **Description** (Demo project for Spring Boot), and **Package name** (com.example.SpringBootApplication).
- Packaging:** Includes radio buttons for **Jar** (selected) and **War**.
- Java:** Includes radio buttons for **23**, **21**, and **17** (selected).
- Dependencies:** Includes a button **ADD DEPENDENCIES... CTRL + B** and a list of 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.

At the bottom, there are buttons for **GENERATE CTRL + G**, **EXPLORE CTRL + SPACE**, and a button with three dots.

Click Generate, download the project, and extract it.

## Step 2: Configure pom.xml

Ensure your pom.xml includes the latest Spring Boot version and Java 17+.

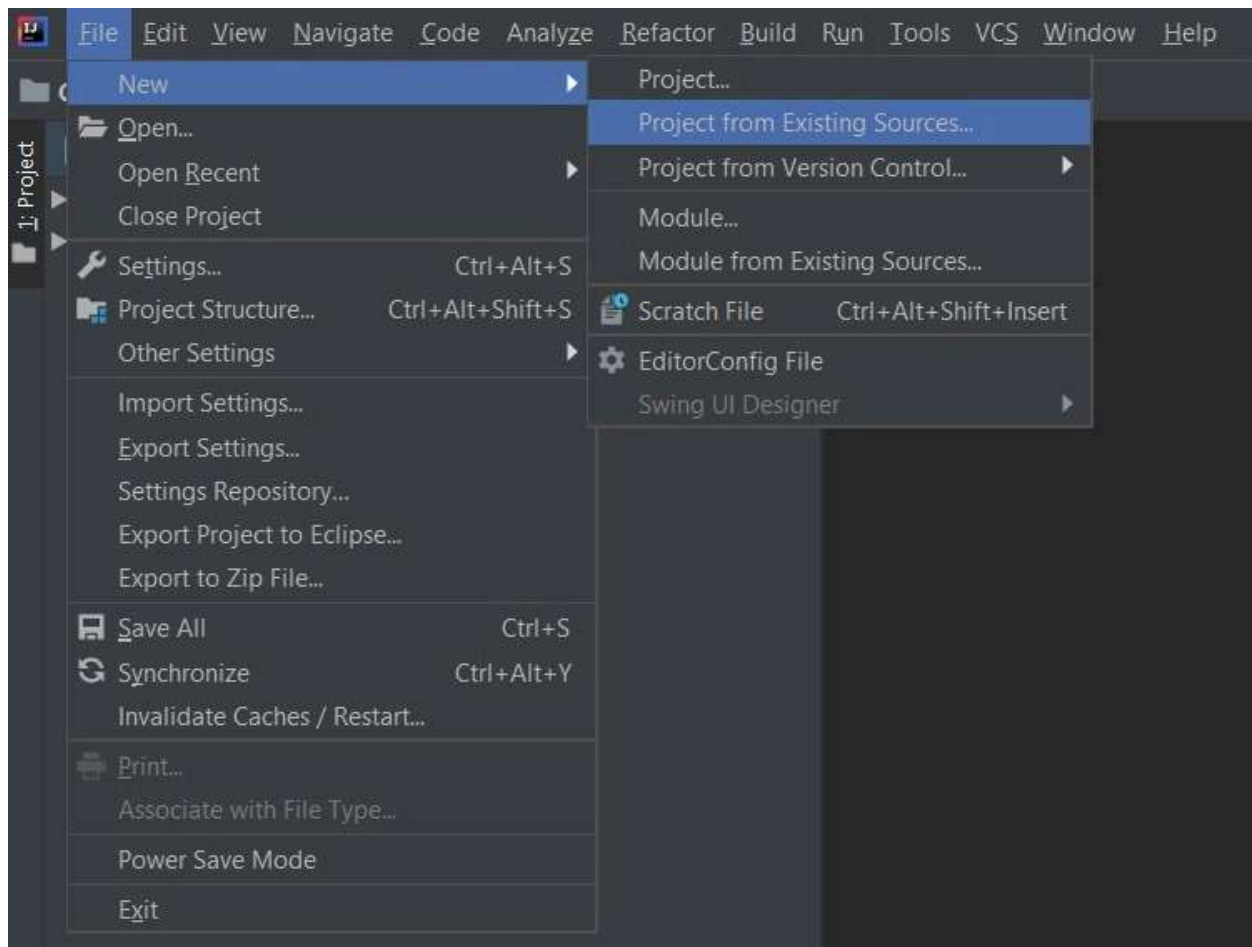
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="https://maven.apache.org/POM/4.0.0"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.0</version> <!-- Latest Spring Boot version -->
  </parent>
  <relativePath/>
  <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-j"va</artifactId>
<scope>runtime</scope>
</dependency>
</dependencies>
<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 and select pom.xml. Click on import changes on prompt and wait for the project to sync



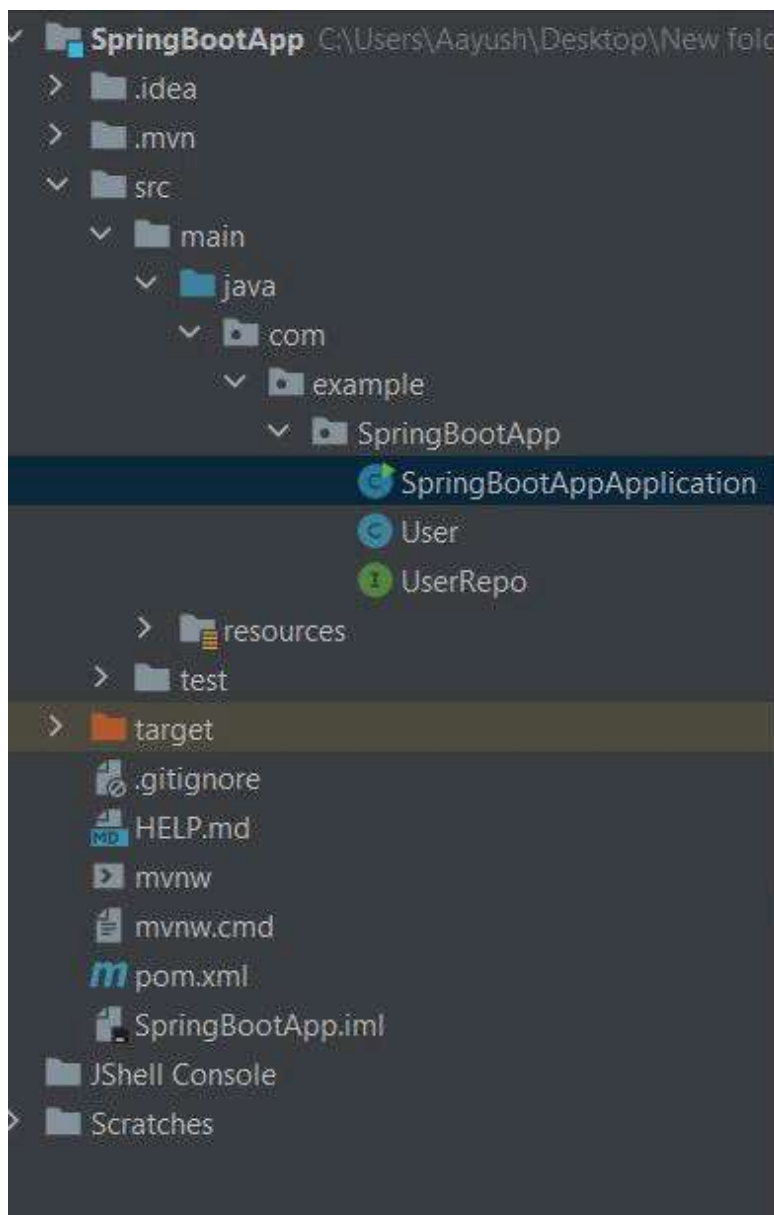
**Note:** In the Import Project for Maven window, make sure you choose the same version of JDK which you selected while creating the project.

### Step 3: Configure application.properties for Database Configuration

application.properties file:

```
spring.datasource.url=jdbc:mysql://localhost:3306/user_db
spring.datasource.username=${DB_USERNAME}
spring.datasource.password=${DB_PASSWORD}
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

**Project Structure:**



## Define the Entity and Repository

### Step 4: Create User Entity

Use Lombok for cleaner code and JPA annotations for table mapping.

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

@Entity
@Table(name = "users")
@Getter
@Setter
```

```
@NoArgsConstructor
@AllArgsConstructor
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(nullable = false)
    private String name;
}
```

## Step 5: Create UserRepository

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

@Repository
public interface UserRepository extends JpaRepository<User, Integer> {
}
```

## Insert Data into MySQL

### Step 6: Create Spring Boot Main Class

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

@SpringBootApplication
public class SpringBootApplication implements CommandLineRunner {

    @Autowired
    private UserRepository userRepository;

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

    @Override
    public void run(String... args) throws Exception {
        User firstUser = new User();
    }
}
```

```
firstUser.setName("Aayush");

userRepository.save(firstUser);
}
```

## Running the Application

The application will start, and data will be inserted into MySQL.

### Output:

```
: Starting SpringBootApplication 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 repository
: 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 SpringBootApplication in 7.005 seconds (JVM running for 7.627
```

### Database Output:

Verify the inserted data by running the SQL query:

```
SELECT * FROM users;
```



```
mysql> show tables;
+-----+
| Tables_in_insertdata |
+-----+
| hibernate_sequence   |
| user                 |
+-----+
2 rows in set (0.01 sec)

mysql> select * from user;
+----+-----+
| id | name  |
+----+-----+
| 1  | Aayush |
+----+-----+
1 row in set (0.00 sec)
```

[Comment](#)[More info](#)[Advertise with us](#)

**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



[Advertise with us](#)

[Company](#)

[Explore](#)