

Making Java easy to learn

Java Technology and Beyond



You are here ▶ Home > java >

How To Save Data Into Database Using Spring Data JPA : Step By Step Tutorial

java Spring Spring Boot Spring Data JPA by dev5003 - May 13, 2021 0

If we are developing a project using Spring Boot, we take help of Spring Data JPA to solve our database operations need. In this article we will go through "How to Save Data into Database Using Spring Data JPA : Step by Step Tutorial". If we are developing a project using Spring boot, saving data into the database is the mandatory topic to learn. Furthermore, we will go through step by step in this article. Also, we will save bulk data in a batch. Now let's start our exercise as "How to Save Data into Database Using Spring Data JPA : Step by Step Tutorial".

Table of Contents (Click on links below to navigate) [hide]

- 1 Software Used
- 2 Step #1 : Creating Starter Project using STS
- 3 Step#2 : Writing codes to implement bulk save operation
 - 3.1 Updating application.properties
 - 3.2 New Classes/Interfaces
 - 3.2.1 Employee.java
 - 3.2.2 EmployeeRepository.java(Interface)
 - 3.2.3 DBOperationRunner.java
- 4 Step#3: Running the application
- 5 Step#4: Verify saved records in the Database

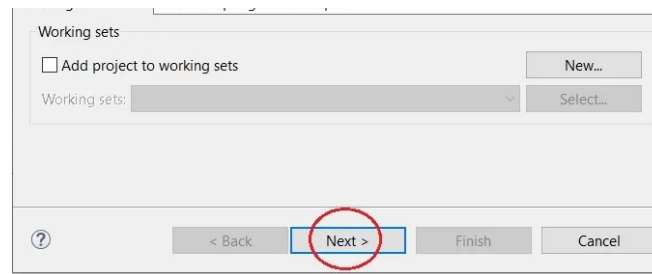
Software Used

- ♦ STS (Spring Tool Suite) : (download link: <https://spring.io/tools>)
- ♦ MySQL Database
- ♦ JDK 8 (Extremely Tested on JDK9, JDK11, JDK 14 & JDK15)

Step #1 : Creating Starter Project using STS

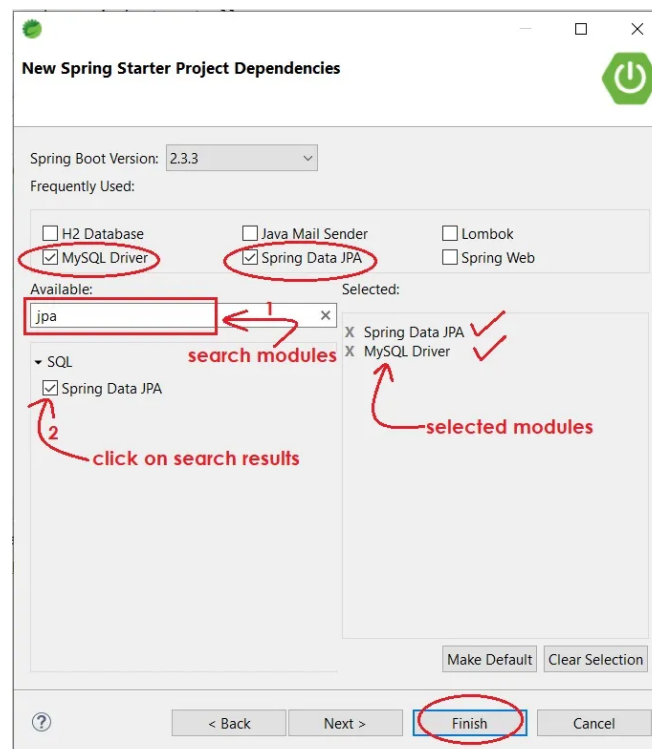
In order to create a Spring Boot Starter Project, we will use STS(Spring Tool Suite). Open your STS and Go to File >> New >> Spring Starter Project -> Then enter your project name in the Name field as shown below. It's "SpringBootDataJPAwithMySQL" in my case as shown below in the screenshot. Next, enter the value of Group field which will become the parent package name of your src folder. Then click on 'Next'.

The screenshot shows the 'New Spring Starter Project' dialog box in Spring Tool Suite (STS). The 'Service URL' is set to 'https://start.spring.io'. The 'Name' field contains 'SpringBootDataJPAwithMySQL', highlighted with a red box and a red arrow pointing to it with the label 'Project Name'. The 'Group' field contains 'com.dev.springboot.data.jpa', also highlighted with a red box and a red arrow pointing to it with the label 'src Package Structure'. Other fields include 'Location' (D:\Workspace-Spring-Tool-Suite-4-4.6.2.RELEASE\SpringBootDataJ...), 'Type' (Maven), 'Packaging' (Jar), 'Java Version' (8), 'Language' (Java), 'Artifact' (SpringBootDataJPAwithMySQL), 'Version' (0.0.1-SNAPSHOT), 'Description' (Demo project for Spring Boot), and 'Package' (com.dev.springboot.data.jpa).

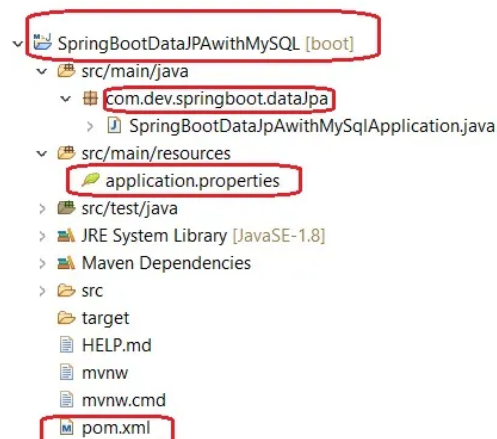


Since we are going to save data into our MySQL database, we need to add some starter dependencies in our project. These dependencies are nothing but required dependent jars. Spring Boot offers some small starter projects/modules to support these dependencies. Hence, we have to select dependent starter projects which are already provided by Spring boot. Search MySQL string in Available field. Select 'MySQL Driver' from the results. Additionally, search JPA string in Available field. Then select 'Spring Data JPA' from the results. Now make sure that both selected modules came under Selected field.

In case you selected wrong module, you can cancel it by clicking on cross sign at Selected field. Then click on 'Finish'. Here selected modules (dependent projects) will be added into pom.xml as a dependency & all required jars will get downloaded automatically. Needless to say, 'MySQL Driver' and 'Spring Data JPA' are the dependencies to support database operations using MySQL DB.



After clicking on 'Finish', you will see Project is created in Package Explorer accordingly as shown below.



Step#2 : Writing codes to implement bulk save operation

In order to implement our functionality, we will update application.properties file & write 2 new classes & 1 interface to implement our functionality.

Updating application.properties

Here, we will update application.properties file mainly for database properties accordingly as below.

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mytestdb
spring.datasource.username=root
spring.datasource.password=devs
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

New Classes/Interfaces

We will create new files under our package 'com.dev.springboot.dataJpa'. Since it is very small functionality & for testing purpose only, briefly, we will create all classes & interfaces in a single package only. But in real time project we should create separate packages for different purpose accordingly.

We will create 2 new classes & one Interface as below.

Employee.java

We will create an Entity class that will map to DB table. It will have all the required fields, getters/setters & constructors annotated with @Entity as below.

```
package com.dev.springboot.dataJpa;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Employee {

    @Id
    private Integer empId;
    private String empName;
    private Double empSal;
    private String empDept;

    public Employee() {

    }

    public Employee(Integer empId, String empName, Double empSal, String empDept) {
        super();
        this.empId = empId;
        this.empName = empName;
        this.empSal = empSal;
        this.empDept = empDept;
    }

    public Integer getEmpId() {
        return empId;
    }

    public void setEmpId(Integer empId) {
        this.empId = empId;
    }
}
```

```

    }
    public String getEmpName() {
        return empName;
    }
    public void setEmpName(String empName) {
        this.empName = empName;
    }
    public Double getEmpSal() {
        return empSal;
    }
    public void setEmpSal(Double empSal) {
        this.empSal = empSal;
    }
    public String getEmpDept() {
        return empDept;
    }
    public void setEmpDept(String empDept) {
        this.empDept = empDept;
    }
}

```

♥ **Note :** You can still minimize the boilerplate code from this Entity class if you use Lombok API. Further the above class code will reduce to its minimum as below. If you are not familiar with applying Lombok, kindly visit our [internal article on Lombok](#).

```

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Employee {

    @Id
    private Integer empId;
    private String empName;
    private Double empSal;
    private String empDept;
}

```

EmployeeRepository.java(Interface)

An interface which extends from JpaRepository<Employee,Integer>. JpaRepository has multiple DB operation methods. For bulk save operation, we will use method saveAll () which will return All Employee data as a List without declaring it in our EmployeeRepository. Of course, this method will be available in our EmployeeRepository through inheritance only.

```

package com.dev.springboot.dataJpa;

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

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

```

DBOperationRunner.java

This is a Runner class with run() method similar to a thread which is used to run a functionality only once. In fact, It implements CommandLineRunner interface provided by Spring boot. DBOperationRunner class will have @Component so that Spring container creates object of it & our functionality can be run. Further, we will autowire EmployeeRepository via @Autowired in this class so that we can use saveAll() method from the EmployeeRepository interface through JpaRepository.

```
package com.dev.springboot.dataJpa;

import java.util.Arrays;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class DBOperationRunner implements CommandLineRunner {

    @Autowired
    EmployeeRepository eRepo;

    @Override
    public void run(String... args) throws Exception {

        eRepo.saveAll(Arrays.asList(

            new Employee(1001, "James", 2599.5, "HR"),
            new Employee(1002, "Elizabeth", 2999.0, "Admin"),
            new Employee(1003, "Robert", 2699.5, "Testing"),
            new Employee(1004, "Victoria", 3000.5, "Development"),
            new Employee(1005, "David", 2650.5, "QA"),
            new Employee(1006, "Isabel", 2590.0, "Support"),
            new Employee(1007, "Michael", 3599.75, "Development"),
            new Employee(1008, "Maria", 2499.0, "Finance"),
            new Employee(1009, "Thomas", 2799.25, "HR"),
            new Employee(1010, "Maria", 2899.5, "Development"))

        );

        // *** Below method List.of(...) will work for JDK 9 onwards***
        // It will not work in Java 8

        /*
        eRepo.saveAll(List.of(


            new Employee(1001, "James", 2599.5, "HR"),
            new Employee(1002, "Elizabeth", 2999.0, "Admin"),
            new Employee(1003, "Robert", 2699.5, "Testing"),
            new Employee(1004, "Victoria", 3000.5, "Development"),
            new Employee(1005, "David", 2650.5, "QA"),
            new Employee(1006, "Isabel", 2590.0, "Support"),
            new Employee(1007, "Michael", 3599.75, "Development"),
            new Employee(1008, "Maria", 2499.0, "Finance"),
            new Employee(1009, "Thomas", 2799.25, "HR"),
            new Employee(1010, "Maria", 2899.5, "Development"))

        );

        System.out.println("-----All Data saved into Database-----");
    }

}
```

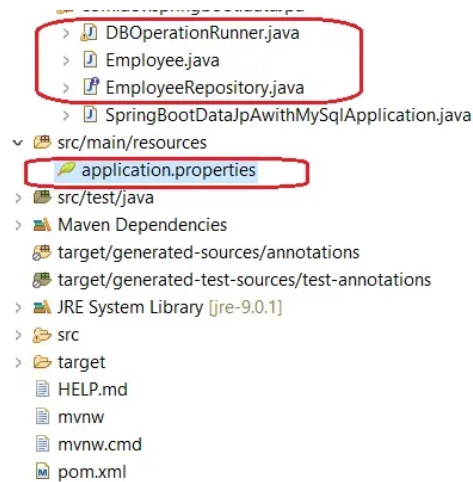
After creating all the files, at last your project structure should look like below screenshot.



```

SpringBootDataJPAwithMySQL [boot]
└── src/main/java
    └── com.dev.springboot.dataJpa

```



Step#3: Running the application

In the end, to run the application right click on the project and then select "Run As" >> "Spring Boot App". Simultaneously, observe the console to see database queries(something like below screenshot) & message "—All Data saved into Database—".

```
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_dept as emp_dept2_0_0_,
-----All Data saved into Database-----
```

Step#4: Verify saved records in the Database

Open MySQL database software, select database 'mytestdb' and then execute the query 'SELECT * FROM EMPLOYEE'. You will see the below output immediately.

emp_id	emp_dept	emp_name	emp_sal
1001	HR	James	2599.5
1002	Admin	Elizabeth	2999
1003	Testing	Robert	2699.5
1004	Development	Victoria	3000.5
1005	QA	David	2650.5
1006	Support	Isabel	2590
1007	Development	Michael	3599.75
1008	Finance	Maria	2499
1009	HR	Thomas	2799.25
1010	Development	Maria	2899.5

That's it. How Simple is that in Spring Boot !. For More details on Spring Data JPA, kindly visit spring.io.

Furthermore, for trending hands-on topics on Spring Boot kindly visit <https://javatechonline.com/blogs/> and <https://javatechonline.com/spring-boot/>

Spring Boot Batch Example CSV to MySQL Using JPA

June 30, 2022

In "java"

Spring Transaction Annotations With Examples

August 11, 2022

In "java"

Spring Batch Tutorial

June 22, 2022

In "java"

Me gusta

Tagged [how to save data in database using spring boot](#) [How to Save Data into Database Using Spring Data JPA: Step by Step Tutorial](#) [How to Save list of object in spring boot?](#) [how to store data in database using spring boot](#) [jpa example](#) [jpa in spring boot](#) [jpa repository example](#) [jpa repository in spring boot](#) [jpa repository save](#) [jpa repository spring boot](#) [jpa save](#) [jpa save update](#) [jpa spring boot](#) [jparepository spring boot](#) [list.of not working in Java 8](#) [save data in database using spring boot](#) [save data in database using spring boot jpa](#) [save file in database spring boot](#) [spring boot crud example](#) [spring boot data jpa](#) [spring boot jpa](#) [spring boot jpa example](#) [spring boot jpa repository](#) [spring boot jpa save](#) [spring boot save data to database](#) [spring boot save form data to database](#) [spring boot save to database](#) [spring data](#) [spring data jpa](#) [spring data jpa example](#) [spring data jpa repository](#) [spring data jpa save example](#) [spring data jpa save or update example](#) [spring data jpa tutorial](#) [spring data tutorial](#) [spring jpa](#) [spring jpa example](#) [spring jpa hibernate](#) [spring jpa tutorial](#) [spring save to database](#) [springboot save](#)

< **Previous article**

JVM Architecture & Class Loaders Java

Next article >

Default Method in Interface

Leave a Reply

Name *

Email Address *

Website

☐ Yes, add me to your mailing list

Post Comment

Comment *

FOLLOW US

RECENTLY PUBLISHED POSTS

- » **Spring Transaction Annotations With Examples**
- » **Spring Scheduling Annotations With Examples**
- » **Spring Security Annotations With Examples**
- » **Spring Boot Interview Questions & Answers**
- » **Java Interface**
- » **Spring Security Without WebSecurityConfigurerAdapter**
- » **Spring Boot Batch Example CSV to MySQL Using JPA**
- » **Spring Batch Tutorial**
- » **Java 14 Features**
- » **Java Features After Java 8**

DO YOU HAVE A QUERY ?

Submit your Query

Email *

Subscribe to receive Updates !

© 2022

Powered by javatechonline.com | <https://javatechonline.com>

[Home](#) [About](#) [Privacy Policy](#) [Terms & Conditions](#) [Disclaimer](#) [Contact Us](#)