**7) AIM: Basic Spring Boot Application with Spring Data JPA.**

**Description :**

This is a basic Spring Boot application that demonstrates Auto-Wiring and Bean Scopes.

• TheStudent class depends on the Marks class, and Spring automatically injects

Marks into Student using @Autowired.

• @Componentisused to register both classes (Student and Marks) as Spring

Beans.

• Inthemainclass (FourthprogramApplication), we retrieve the Student bean from

the ApplicationContext and call its methods along with the Marks methods

**Student.java**

package com.example;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

*@Entity*

public class Student {

*@Id*

private int sno;

private String sname;

public Student() {}

public Student(int sno, String sname) {

super();

this.sno = sno;

this.sname = sname;

}

public int getSno() {

return sno;

}

public void setSno(int sno) {

this.sno = sno;

}

public String getSname() {

return sname;

}

public void setSname(String sname) {

this.sname = sname;

}

}

**StudentsApplication.java**

package com.example;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

*@SpringBootApplication*

public class StudentsApplication {

public static void main(String[] args) {

SpringApplication.*run*(StudentsApplication.class, args);

}

*@Bean*

CommandLineRunner initDatabase(StudentRepository repo) {

return args -> {

repo.save(new Student(1, "Rakesh kumar"));

repo.save(new Student(2, "Murali"));

repo.save(new Student(3, "vamsi"));

System.***out***.println("Students inserted!");

};

}

}

**StudentController.java**

package com.example;

import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

*@RestController*

public class StudentController {

private final StudentRepository repo;

public StudentController(StudentRepository repo) {

this.repo = repo;

}

*@PostMapping*

public Student addStudent(*@RequestBody* Student student) {

return repo.save(student);

}

*@GetMapping*

public List<Student> getAllStudents() {

return repo.findAll();

}

}

**StudentRepository.java**

package com.example;

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

public interface StudentRepository extends JpaRepository<Student, Integer> {

}

**application.properties**

spring.application.name=Students

server.port=8543

spring.datasource.url=jdbc:mysql://localhost:3306/demo

spring.datasource.username=root

spring.datasource.password=Revs@123

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://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>3.5.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com</groupId>

<artifactId>StudentsApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Students</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

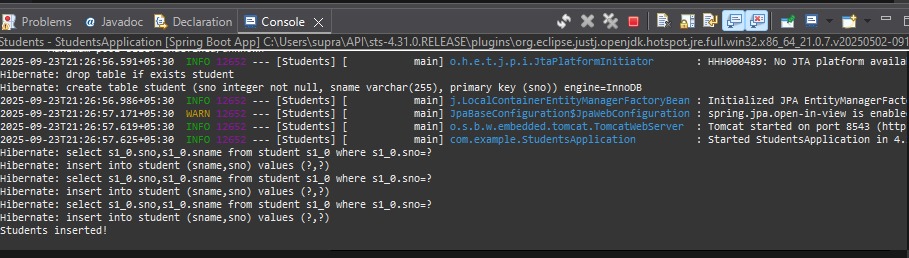
</plugin>

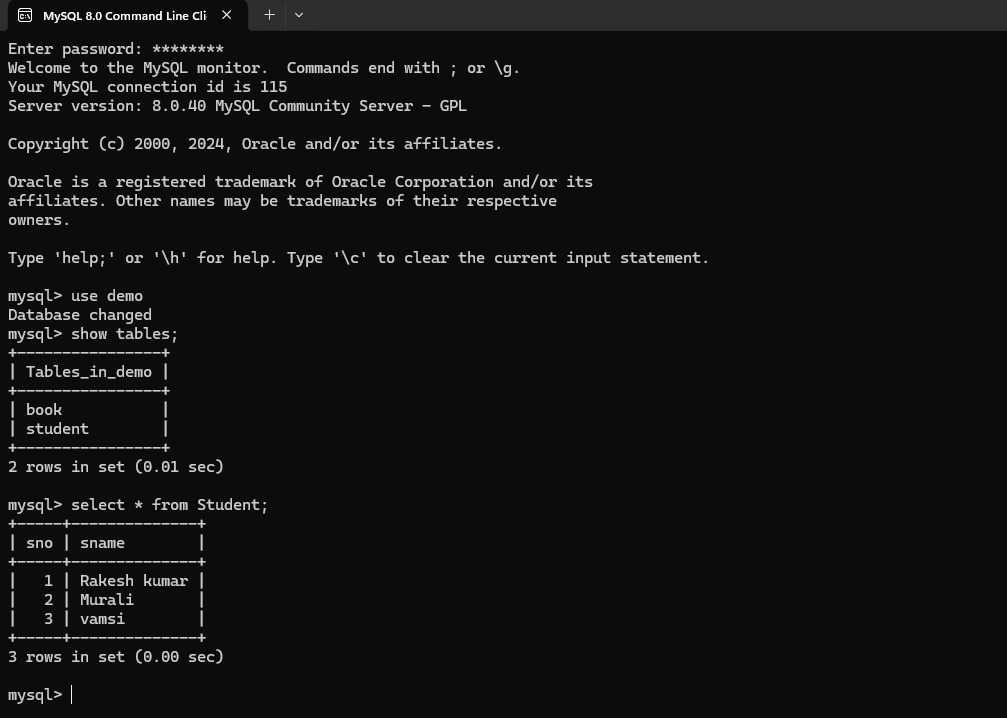
</plugins>

</build>

</project>

**Output:**





**8) AIM: Pagination and Sorting in Spring Data JPA.**

**Description:**

This program explains the difference between Singleton and Prototype bean scopes in

Spring Boot

• Singleton Scope (@Scope("singleton")): Only one instance of the bean is created

for the entire Spring container. Any request for that bean will return the same

instance.

• Prototype Scope (@Scope("prototype")): A new instance of the bean is created

every time it is requested from the container.

• SingletonBean is shared across requests. Changes in one reference affect the

other.

• PrototypeBean creates new objects each time, so changes in one object do not

affect another

**Book.java**

package com.example;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

*@Entity*

public class Book {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private long id;

private String title;

private String author;

public Book() {}

public Book( String title, String author) {

super();

this.title = title;

this.author = author;

}

*@Override*

public String toString() {

return "Book [id=" + id + ", title=" + title + ", author=" + author + "]";

}

public long getId() {return id;}

public void setId(long id) {this.id = id;}

public String getTitle() {return title;}

public void setTitle(String title) {this.title = title;}

public String getAuthor() {return author;}

public void setAuthor(String author) {this.author = author;}

}

**BookApplication.java**

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class BookApplication {

public static void main(String[] args) {

SpringApplication.*run*(BookApplication.class, args);

}

}

**BookController.java**

package com.example;

import org.springframework.data.domain.Page;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.data.domain.Sort;

import org.springframework.web.bind.annotation.\*;

*@RestController*

*@RequestMapping*("/books")

public class BookController {

*@Autowired*

private BookRepository bookRepository;

*@GetMapping*("/init")

public String initData() {

if (bookRepository.count() == 0) {

bookRepository.save(new Book("Spring Boot Basics", "Johnardan"));

bookRepository.save(new Book("Spring Boot Basics", "Eleven"));

bookRepository.save(new Book("Spring Boot Basics", "Will"));

bookRepository.save(new Book("Spring Boot Basics", "Steve"));

bookRepository.save(new Book("Spring Boot Basics", "Dustiban")); }

return "Sample books added!";

}

*@GetMapping*

public Page<Book> getBooks(

*@RequestParam*(defaultValue = "0") int page,

*@RequestParam*(defaultValue = "3") int size,

*@RequestParam*(defaultValue = "title") String sortBy,

*@RequestParam*(defaultValue = "asc") String direction ){

Sort sort = direction.equalsIgnoreCase("asc") ?

Sort.*by*(sortBy).ascending() :

Sort.*by*(sortBy).descending();

Pageable pageable = PageRequest.*of*(page, size, sort);

return bookRepository.findAll(pageable);

}

}

**BookRepository.java**

package com.example;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**application.properties**

spring.application.name=Book

server.port=8754

spring.datasource.url=jdbc:mysql://localhost:3306/demo

spring.datasource.username=root

spring.datasource.password=Revs@123

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://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>3.5.6</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com</groupId>

<artifactId>BookApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Book</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

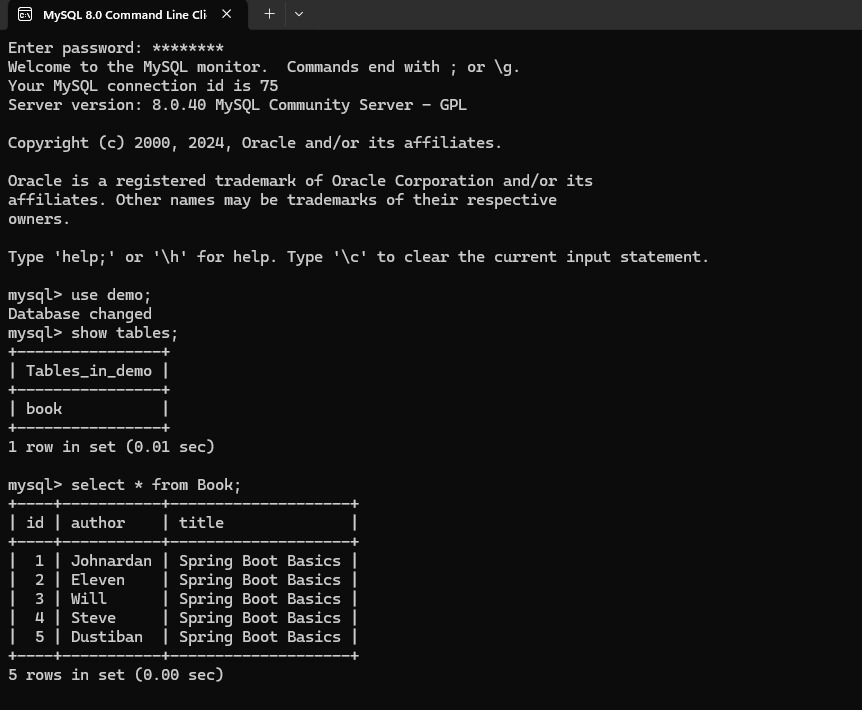
</plugin>

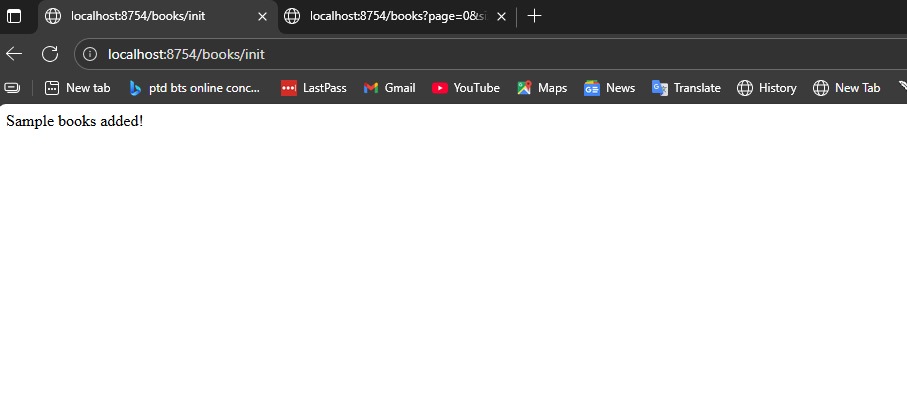
</plugins>

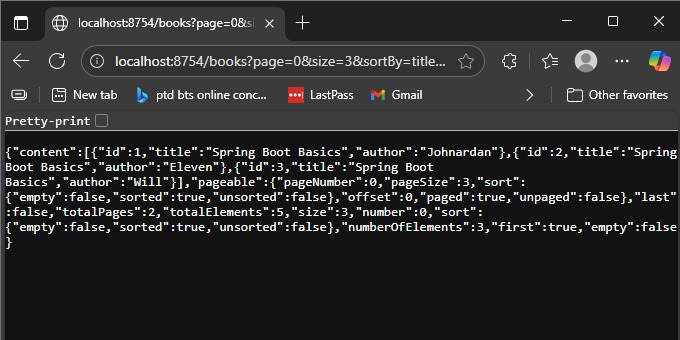
</build>

</project>

**Output:**







**9.AIM: .**

**Description:**