**7) Aim: Basic SpringBoot Application with Spring DataJPA**

**Description:**

In this experiment, we will create a Spring Boot application that connects to a MySQL database and uses Spring Data JPA to perform basic database operations. The application will allow inserting and retrieving StudentDemo records through a RESTful API.

* **StudentDemo.java**–The entity class representing StudentDemo.
* **StudentDemoRepository.java**–TheJPArepositoryinterfacefordatabaseoperations.
* **StudentDemoController.java**–RESTcontrollerforhandlingHTTPrequests.
* **StudentDemoApplication.java**–Mainapplicationclassforbootstrappingtheapplication.
* **application.properties**–Configuration file for database and server.
* **pom.xml** –Maven configuration file for dependencies.

# Program:

**StudentDemoApplication.java**

package com.example;

importorg.springframework.boot.CommandLineRunner; import org.springframework.boot.SpringApplication;

importorg.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean; @SpringBootApplication

publicclassStudentDemoApplication{

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

}

@Bean

CommandLineRunnerinitDatabase(StudentDemoRepositoryrepo){ return args -> {

repo.save(newStudentDemo(1,"Virat")); repo.save(new StudentDemo(2, "Kohli")); repo.save(new StudentDemo(3, "Siri")); System.*out*.println("Students inserted!");

};

}

}

# StudentDemo.java

package com.example;

importjakarta.persistence.Entity; import jakarta.persistence.Id; @Entity

publicclassStudentDemo{

@Id privateintsno;

privateStringsname;

publicStudentDemo(){}

publicStudentDemo(intsno,Stringsname){ this.sno = sno;

this.sname=sname;

}

publicintgetSno() {returnsno;}

publicvoidsetSno(intsno){this.sno=sno;} public String getSname() { return sname; }

publicvoidsetSname(Stringsname){this.sname= sname;}

}

# application.properties

spring.application.name=StudentDemo

server.port=8264

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

spring.datasource.username=root

spring.datasource.password=Alekhya@2004

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

spring.jpa.show-sql=true

//spring.jpa.properties.hibernate.format\_sql=true

//spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# StudentDemoController.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** StudentDemoController {

**private** **final** StudentDemoRepository repo;

**public** StudentDemoController(StudentDemoRepository repo) {

**this**.repo=repo;

}

@PostMapping

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

**return** repo.save(student);

}

@GetMapping

**public** List<StudentDemo> getAllStudents(){

**return** repo.findAll();

}

}

# StudentDemoRepository.java(Interface)

package com.example;

importorg.springframework.data.jpa.repository.JpaRepository;

publicinterface StudentDemoRepositoryextendsJpaRepository<StudentDemo,Integer>{

}

# pom.xml

<?xmlversion="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>3.5.4</version>

<relativePath/><!--lookupparentfromrepository -->

</parent>

<groupId>com</groupId>

<artifactId>StudentDemoApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>StudentDemo</name>

<description>DemoprojectforSpring 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>

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot- starter-data-jpa -->

<dependency>

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

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

<version>3.5.2</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

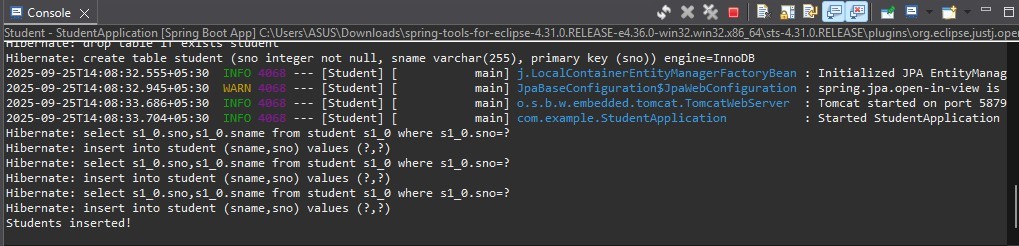
</plugin>

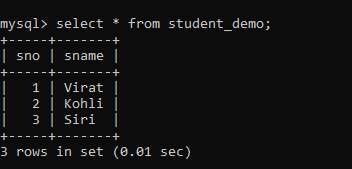
</plugins>

</build>

</project>

**Output:**

****

****

**8)Aim: Pagination and Sorting in Spring Data JPA**

**Description:**

In this experiment, we will create a Spring Boot application that demonstrates how to paginate and sort database records using Spring Data JPA. We will use a Book entity with sample data, a JPA repository interface for database operations, and a REST controller to handle requests. Pagination parameters (page, size) and sorting parameters (sortBy, direction) will be passed via URL query parameters to retrieve data in a paginated and sorted manner.

# Program: application.properties

spring.application.name=Book

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

spring.datasource.username=root

spring.datasource.password=Alekhya@2004

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

spring.jpa.show-sql=true

server.port=6951

# BookApplication.java

package com.example;

importorg.springframework.boot.SpringApplication;

importorg.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication publicclassBookApplication{

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

}

}

# 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) {

this.title = title;

this.author = author;

}

@Override

public String toString() {

return "Book{id=" + id + ", title='" + title + "', author='" + author + "'}";

}

// getters and setters

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; }

}

# BRepository(Interface)

package com.example;

importorg.springframework.data.jpa.repository.JpaRepository;

publicinterfaceBookRepositoryextendsJpaRepository<Book, Long>{

}

# BookController.java

package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.data.domain.Page;

importorg.springframework.data.domain.PageRequest; import org.springframework.data.domain.Pageable; import org.springframework.data.domain.Sort;

importorg.springframework.web.bind.annotation.\*; @RestController

@RequestMapping("/books") publicclassBookController{

@Autowired

privateBookRepositorybookRepository; @GetMapping("/init")

publicStringinitData(){

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

bookRepository.save(new Book("Spring Boot Basics", "John")); bookRepository.save(new Book("Java Programming", "Alice")); bookRepository.save(new Book("Hibernate in Action","Bob")); bookRepository.save(newBook("MicroservicesGuide","Carol")); bookRepository.save(new Book("Data Structures", "Davidraj"));}

return"Samplebooksadded!";

}

@GetMapping

public Page<Book> getBooks( @RequestParam(defaultValue = "0") int page, @RequestParam(defaultValue = "3") int size, @RequestParam(defaultValue = "title") String sortBy, @RequestParam(defaultValue="asc")Stringdirection

){

Sortsort=direction.equalsIgnoreCase("asc")?

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

Pageablepageable=PageRequest.*of*(page,size,sort); return bookRepository.findAll(pageable);

}

}

# pom.xml

<?xmlversion="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/><!--lookupparentfromrepository -->

</parent>

<groupId>com.example</groupId>

<artifactId>BookApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Book</name>

<description>DemoprojectforSpring Boot</description>

<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>

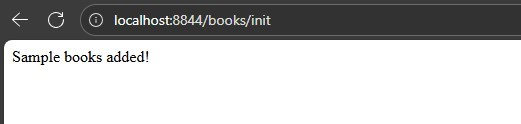
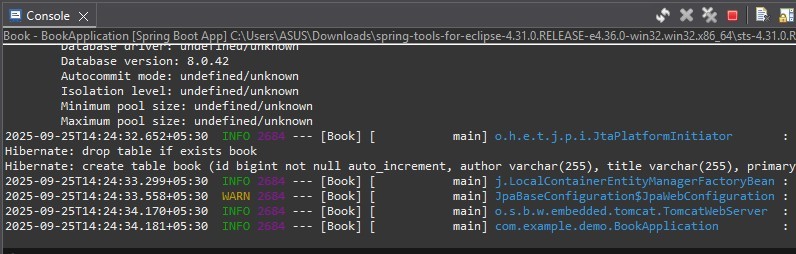
</plugin>

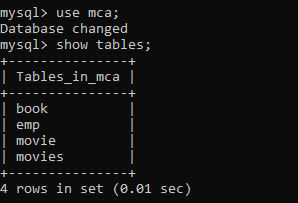
</plugins>

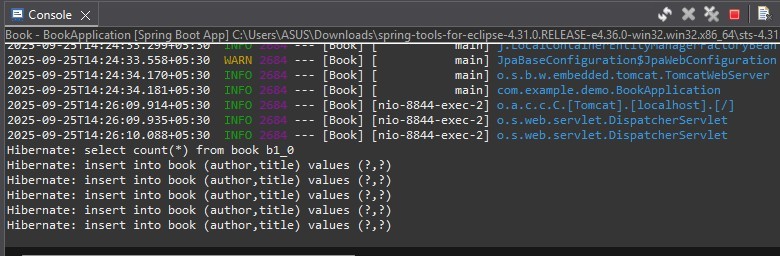
</build>

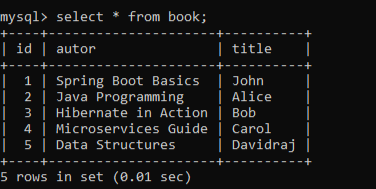
</project>

**Output:**

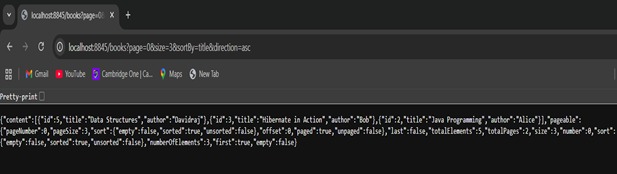
****

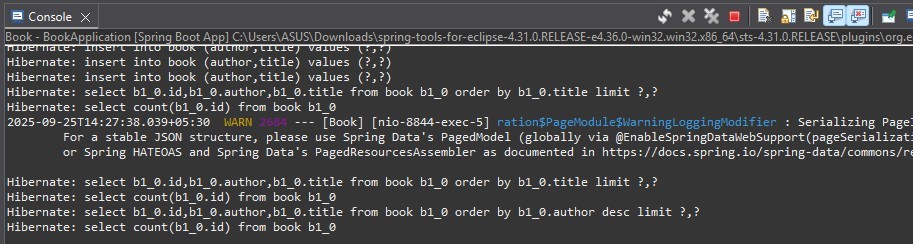
****

****



****

****

****

**9) Aim: Implementing AOP for Logging with Spring Data JPA**

**Description:**

In this experiment,we create a SpringBootapplication to manage products.The application includes:

* **Entity**–Product with id,name,and price.
* **Repository**–ProductRepository for database operations.
* **Service**–ProductService to handle businesslogic.
* **Controller**–ProductController for RESTAPIs.
* **Aspect**–LoggingAspect to log method calls in ProductService.
* **Database**–H2in-memoryDBor MySQL.
* **DependencyManagement**–Managed via Maven(pom.xml).

This demonstrates the use of **SpringDataJPA**,**SpringAOP**,and**RESTfulAPIdevelopment**.

# Program:

**ProductRepository.java(Interface)**

**package**com.example;

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

**publicinterface**ProductRepository**extends**JpaRepository<Product,Long>{

}

# ProductService.java

package com.example;

importorg.springframework.stereotype.Service; import java.util.List;

@Service

publicclassProductService{

privatefinalProductRepository repo;

publicProductService(ProductRepositoryrepo){ this.repo = repo;

}

publicProductsaveProduct(Productproduct){ return repo.save(product);

}

publicList<Product>getAllProducts(){ return repo.findAll();

}

}

**application.properties**

spring.application.name=product

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

spring.datasource.username=root

spring.datasource.password=Alekhya@2004

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

spring.jpa.show-sql=true

server.port=8000

# Product.java

package com.example.; importjakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType; import jakarta.persistence.Id;

@Entity

publicclassProduct{ @Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private long id;

privateStringname;

privatedoubleprice; public Product() {}

publicProduct(Stringname,doubleprice){ this.name = name;

this.price= price;}

//getters&setters

publicLong getId(){

return id;}

publicvoidsetId(Longid){

this.id=id;}public String getName() {return name; }

publicvoidsetName(Stringname){

this.name=name;}

public double getPrice() {

return price; }

publicvoidsetPrice(doubleprice){

this.price=price;}

}

# ProductController.java

package com.example;

importorg.springframework.web.bind.annotation.\*; import java.util.List;

@RestController @RequestMapping("/products") publicclassProductController{

privatefinalProductService service;

publicProductController(ProductServiceservice){ this.service = service;

}

@PostMapping("/add")

publicProductaddProduct(@RequestBodyProductproduct){ return service.saveProduct(product);

}

@GetMapping("/all")

publicList<Product>getAllProducts(){ return service.getAllProducts();

}

}

# LoggingAspect.java

package com.example; importorg.aspectj.lang.JoinPoint;

importorg.aspectj.lang.annotation.Aspect; importorg.aspectj.lang.annotation.Before;

importorg.springframework.stereotype.Component;

@Aspect @Component

publicclassLoggingAspect{

// Logs before executing any ProductService method @Before("execution(\*com.example.demo.ProductService.\*(..))") public void logBefore(JoinPoint joinPoint) {

System.out.println(">>>Enteringmethod:"+joinPoint.getSignature().getName());

}

}

# main.java

package com.example;

importorg.springframework.boot.CommandLineRunner; import org.springframework.boot.SpringApplication;

importorg.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean; @SpringBootApplication

publicclassProductApplication{

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

}

@Bean

CommandLineRunnerrunner(ProductRepositoryrepo){ return args -> {

repo.save(newProduct("Laptop", 55000));

repo.save(newProduct("Mobile", 20000));

repo.save(newProduct("Tablet",30000));

repo.save(newProduct("Mouse",35000));

};

}

}

# pom.xml

<?xmlversion="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>3.5.4</version>

<relativePath/><!--lookupparentfromrepository -->

</parent>

<groupId>com.example</groupId>

<artifactId>ProductApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>product</name>

<description>DemoprojectforSpring 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>

<!--SpringAOP-->

<dependency>

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

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

</dependency>

<!--Lombok(optional,justtoreduceboilerplate)-->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

</dependency>

<!--H2Database(in-memory,noneedforMySQLsetup)-->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

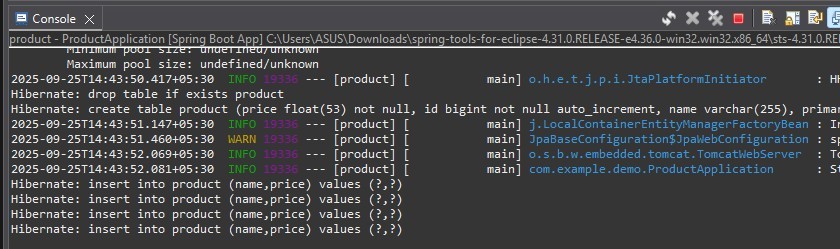
</plugin>

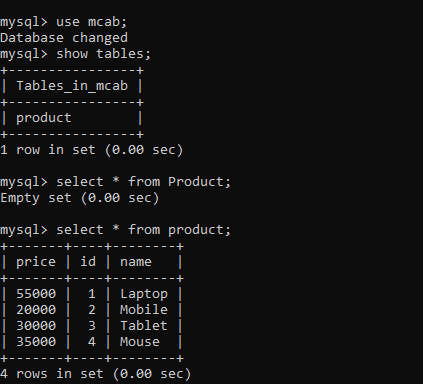
</plugins>

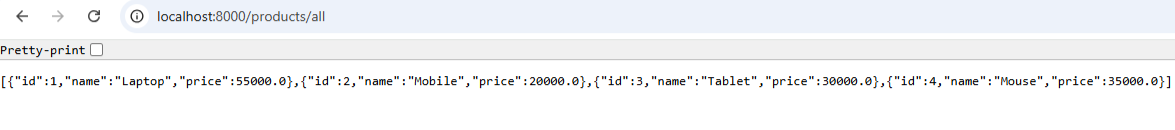
</build>

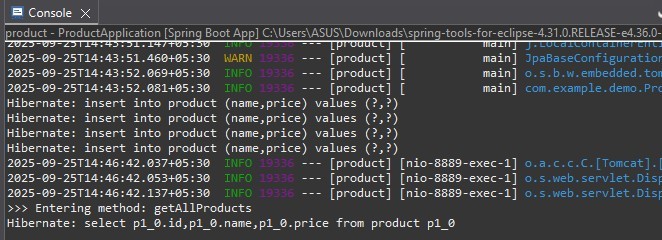
</project>

**Output:**

****

****

****

****