**7)**

**Aim:**

**Basic Spring Boot Application with Spring Data JPA 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 student records through a RESTful API.

* **Student.java** – The entity class representing students.
* **StudentRepository.java** – The JPA repository interface for database operations.
* **StudentController.java** – REST controller for handling HTTP requests.
* **StudentApplication.java** – Main application class for bootstrapping the application.
* **application.properties** – Configuration file for database and server.
* **pom.xml** – Maven configuration file for dependencies.

Program:

# SqlDemoProjectApplication.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** SqlDemoProjectApplication {

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

}

@Bean

CommandLineRunner initDatabase(StudentRepository repo) {

**return** args -> {

repo.save(**new** Student(1, "Billa"));

repo.save(**new** Student(2, "Remo")); // ✅use different id System.***out***.println("Students Inserted!");

};

}

}

**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) {

**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; }

}

# application.properties

spring.application.name=Student server.port=9650

spring.datasource.url=jdbc:mysql://localhost:3306/mcab spring.datasource.username=root spring.datasource.password=Vara97#jaya spring.jpa.hibernate.ddl-auto=create-drop spring.jpa.show-sql=true

# StudentController.java

**package** com.example;

**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;

**import** java.util.List; @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 (Interface)

**package** com.example;

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

**public interface** StudentRepository **extends** JpaRepository<Student,Integer>{

}

# 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>3.5.4</version>

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

</parent>

<groupId>com</groupId>

<artifactId>StudentApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Student</name>

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

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

</scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

<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>

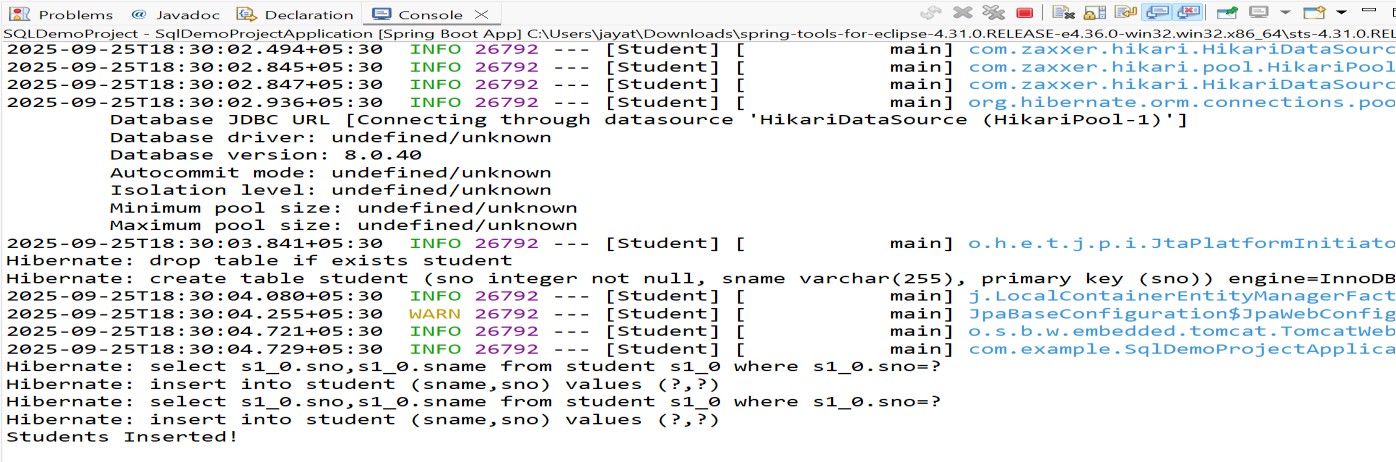
</plugin>

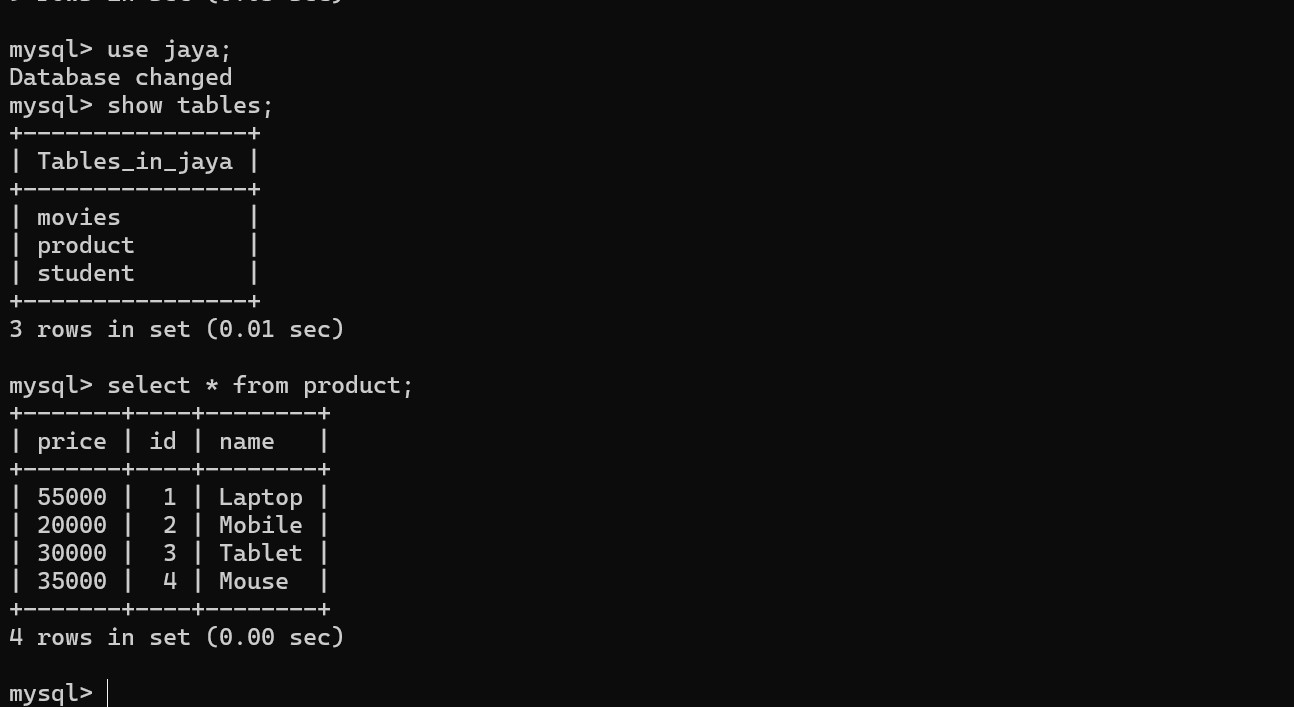
</plugins>

</build>

</project>

**Output:**

****

****

**8)**

**Aim:**

**Basic Spring Boot Application with 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/bookrecords spring.datasource.username=root spring.datasource.password=Vara97#jaya spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true server.port=8842

# 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);

}

}

# 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;

}

}

# BookRepository (Interface)

**package** com.example;

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

**public interface** BookRepository **extends** JpaRepository<Book,Long> {

}

# BookController.java

**package** com.example;

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

**import** org.springframework.data.domain.Page;

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

**import** org.springframework.web.bind.annotation.GetMapping; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RequestParam; **import** org.springframework.web.bind.annotation.RestController; @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", "John")); bookRepository.save(**new** Book("Java Programming", "Alice")); bookRepository.save(**new** Book("Hibernate in Action", "Bob")); bookRepository.save(**new** Book("Microservices Guide", "Carol")); bookRepository.save(**new** Book("Data Structures", "Davidraj"));

}

**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);

}}

# 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>PaginationandSortingApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Book</name>

<description>Demo project for Spring 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>

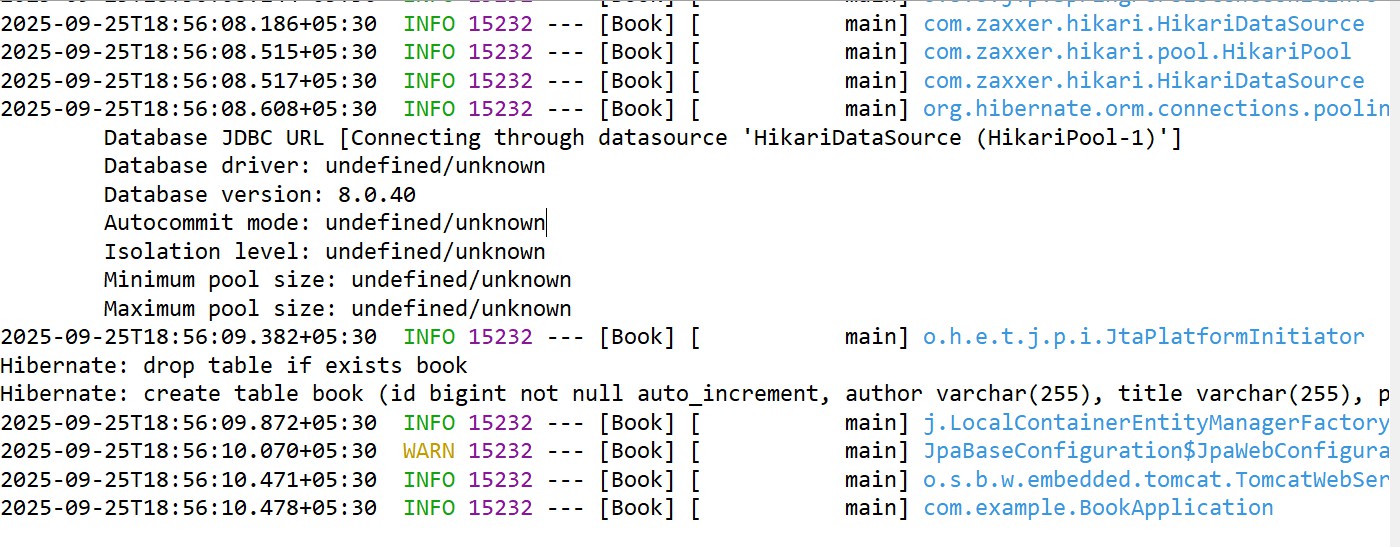
</plugin>

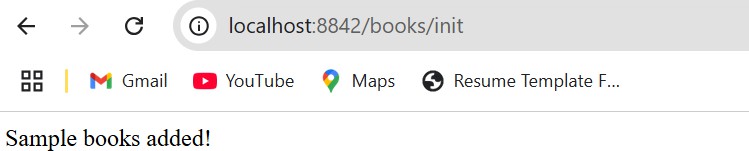
</plugins>

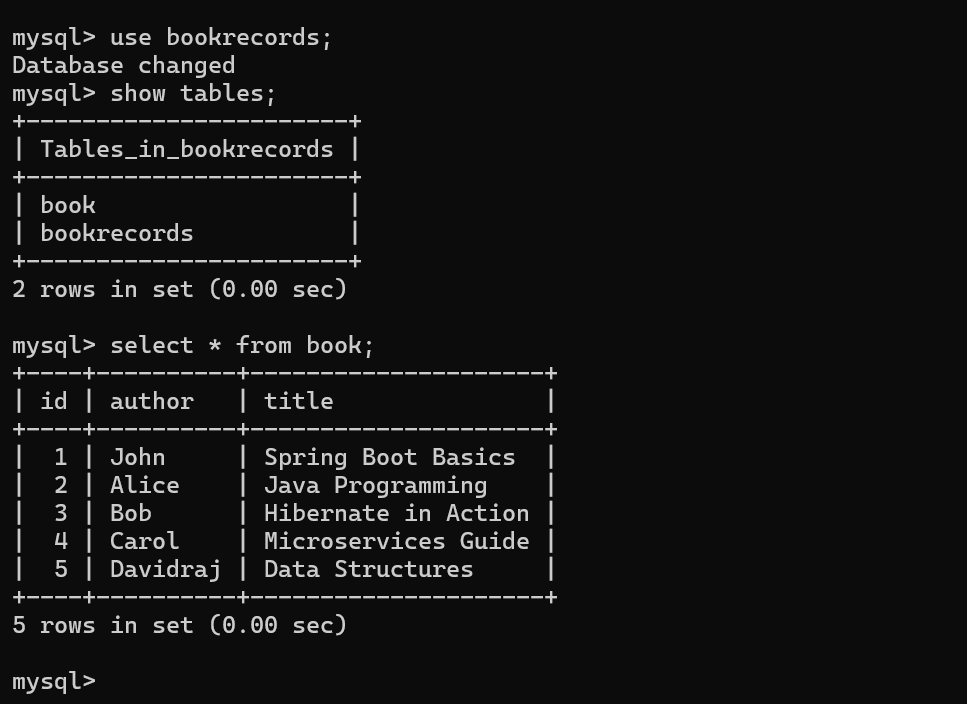
</build>

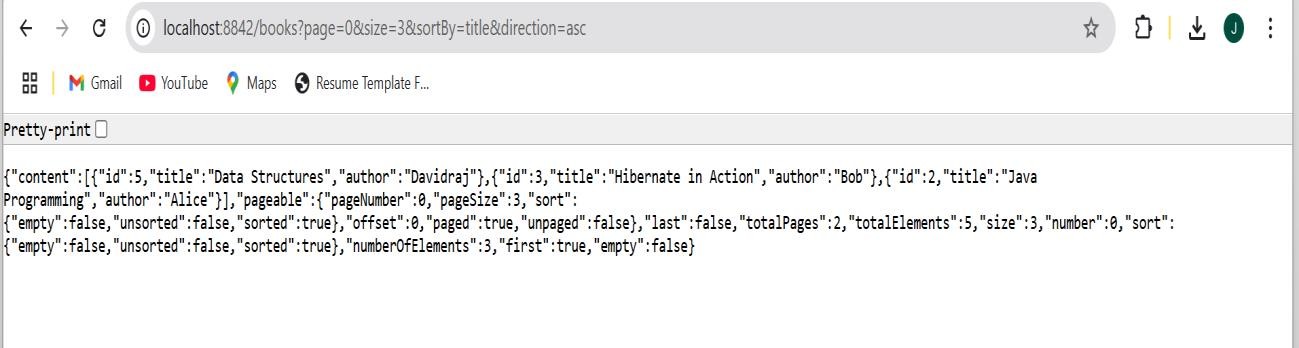
</project>

**Output:**

****

****

****



****

**9)**

**Aim:**

**Implementing AOP for Logging with Spring Data JPA Description:**

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

* **Entity** – Product with id, name, and price.
* **Repository** – ProductRepository for database operations.
* **Service** – ProductService to handle business logic.
* **Controller** – ProductController for REST APIs.
* **Aspect** – LoggingAspect to log method calls in ProductService.
* **Database** – H2 in-memory DB or MySQL.
* **Dependency Management** – Managed via Maven (pom.xml).

This demonstrates the use of **Spring Data JPA**, **Spring AOP**, and **RESTful API development**.

# Program:

**ProductRepository.java (Interface)**

**package** com.example;

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

**public interface** ProductRepository **extends** JpaRepository<Product, Long> {

}

# ProductService.java

package com.example;

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

@Service

public class ProductService {

private final ProductRepository repo;

public ProductService(ProductRepository repo) {

this.repo = repo;

}

public Product saveProduct(Product product) { return repo.save(product);

}

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

}

}

**application.properties** spring.application.name=Product spring.datasource.url=jdbc:mysql://localhost:3306/jaya spring.datasource.username=root spring.datasource.password=Vara97#jaya spring.jpa.hibernate.ddl-auto=create-drop spring.jpa.show-sql=true

server.port=1234

# Product.java

package com.example;

import jakarta.persistence.Entity;

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

@Entity

public class Product { @Id

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

private String name; private double price; public Product() {}

public Product(String name, double price) { this.name = name;

this.price = price;

}

// getters & setters

public Long getId() { return id; }

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

public void setName(String name) { this.name = name; } public double getPrice() { return price; }

public void setPrice(double price) { this.price = price; }

}

# ProductController.java

package com.example;

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

@RestController @RequestMapping("/products") public class ProductController {

private final ProductService service;

public ProductController(ProductService service) { this.service = service;

}

@PostMapping("/add")

public Product addProduct(@RequestBody Product product) { return service.saveProduct(product);

}

@GetMapping("/all")

public List<Product> getAllProducts() { return service.getAllProducts();

}

}

# LoggingAspect.java

package com.example;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect; import org.aspectj.lang.annotation.Before;

import org.springframework.stereotype.Component;

@Aspect @Component

public class LoggingAspect {

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

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

}

}

# ProductApplication.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** ProductApplication {

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

}

@Bean

CommandLineRunner runner(ProductRepository repo) {

**return** args -> {

repo.save(**new** Product("Laptop", 55000));

repo.save(**new** Product("Mobile", 20000));

repo.save(**new** Product("Tablet", 30000));

repo.save(**new** Product("Mouse", 35000));

};

}

}

# 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>3.5.4</version>

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

</parent>

<groupId>com</groupId>

<artifactId>productApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>product</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>

<!-- Spring AOP -->

<dependency>

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

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

</dependency>

<!-- Lombok (optional, just to reduce boilerplate) -->

<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>

<!-- H2 Database (in-memory, no need for MySQL setup) -->

<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>

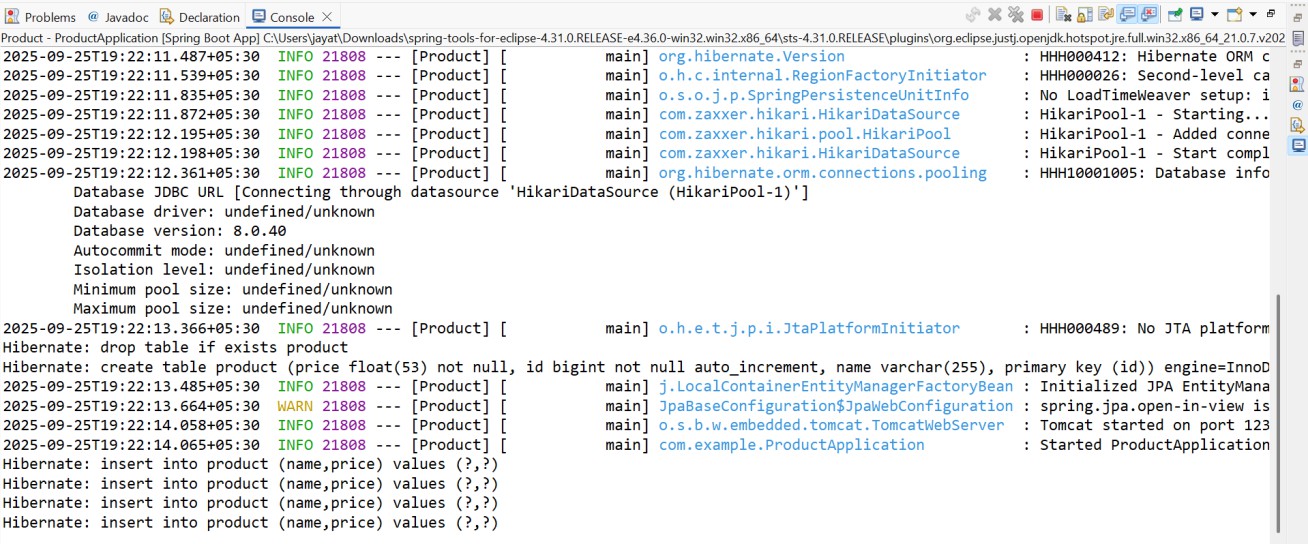
</plugin>

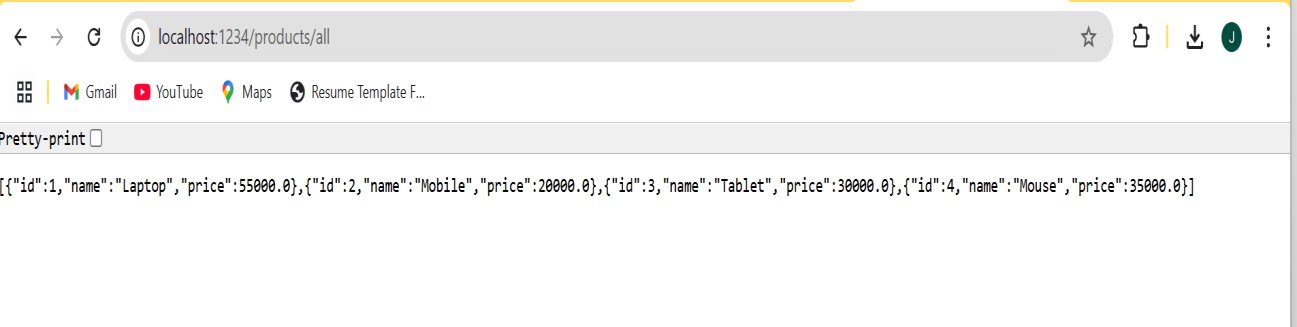
</plugins>

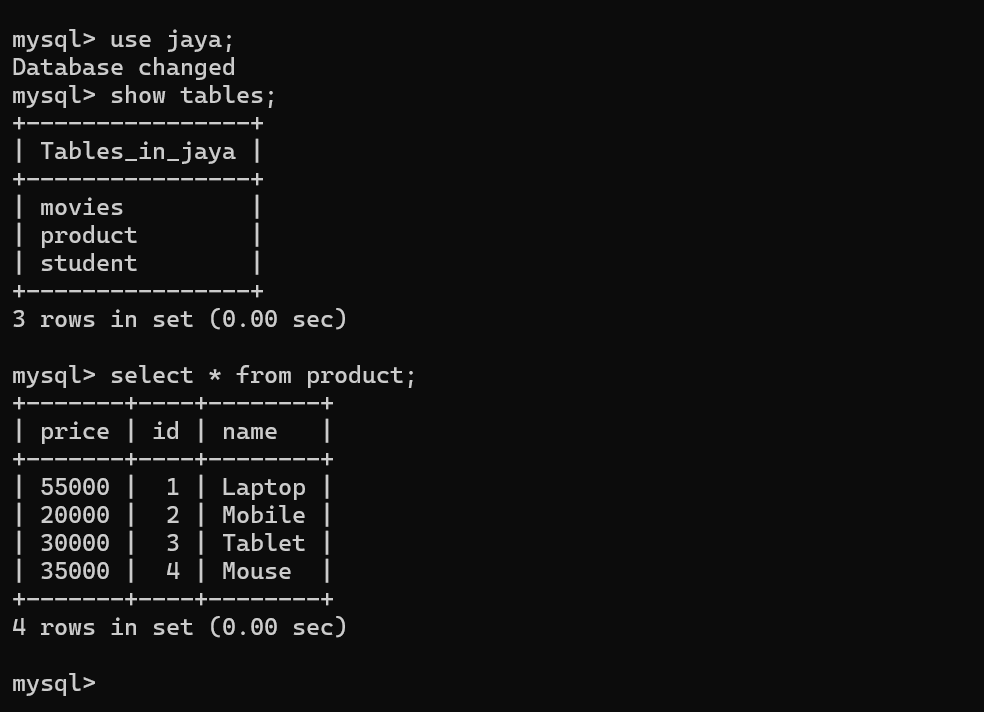
</build>

</project>

**Output:**

****

****

****