Week3

SPRING CORE AND MAVEN

1.

ApplicationContext.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

<https://www.springframework.org/schema/beans/spring-beans.xsd>">

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

MainApp.java

**package** com.library.app;

**import** com.library.service.BookService;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** MainApp {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("The Great Gatsby");

}

}

BookRepository.java

**package** com.library.repository;

**public** **class** BookRepository {

**public** **void** saveBook(String name) {

System.***out***.println("Book '" + name + "' has been saved to the database.");

}

}

BookService.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter method for Spring to inject dependency

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String name) {

System.***out***.println("Adding book: " + name);

bookRepository.saveBook(name);

}

}

Pom.xml

<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>

<http://maven.apache.org/xsd/maven-4.0.0.xsd>">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Core and Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- (Optional) Logging with SLF4J -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.36</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>1.7.36</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler Plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<encoding>UTF-8</encoding>

<source>1.8</source>

<target>1.8</target>

</configuration>

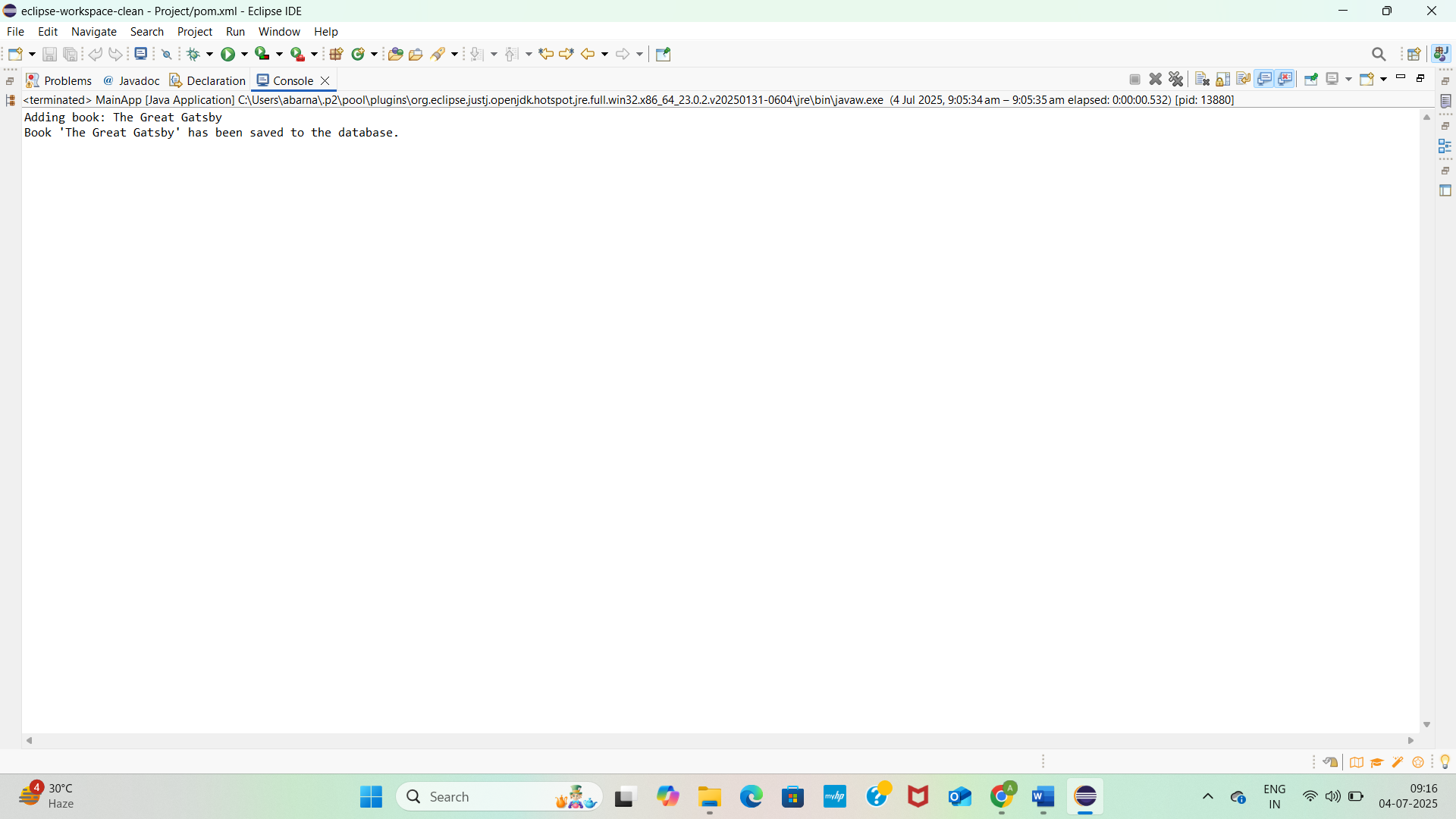
</plugin>

</plugins>

</build>

</project>

OUTPUTSCREENSHOT

2.

ApplicationContext.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

<https://www.springframework.org/schema/beans/spring-beans.xsd>">

<!-- Bean for BookRepository -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<!-- Bean for BookService with Dependency Injection -->

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

BookRepository.java

**package** com.library.repository;

**public** **class** BookRepository {

**public** **void** save(String title) {

System.***out***.println("Book '" + title + "' has been saved to the database.");

}

}

BookService.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for DI

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

System.***out***.println("Adding book: " + title);

bookRepository.save(title);

}

}

MainApp.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for DI

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

System.***out***.println("Adding book: " + title);

bookRepository.save(title);

}

}

Pom.xml

<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>

<http://maven.apache.org/xsd/maven-4.0.0.xsd>">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Core and Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- (Optional) Logging with SLF4J -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.36</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>1.7.36</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler Plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<encoding>UTF-8</encoding>

<source>1.8</source>

<target>1.8</target>

</configuration>

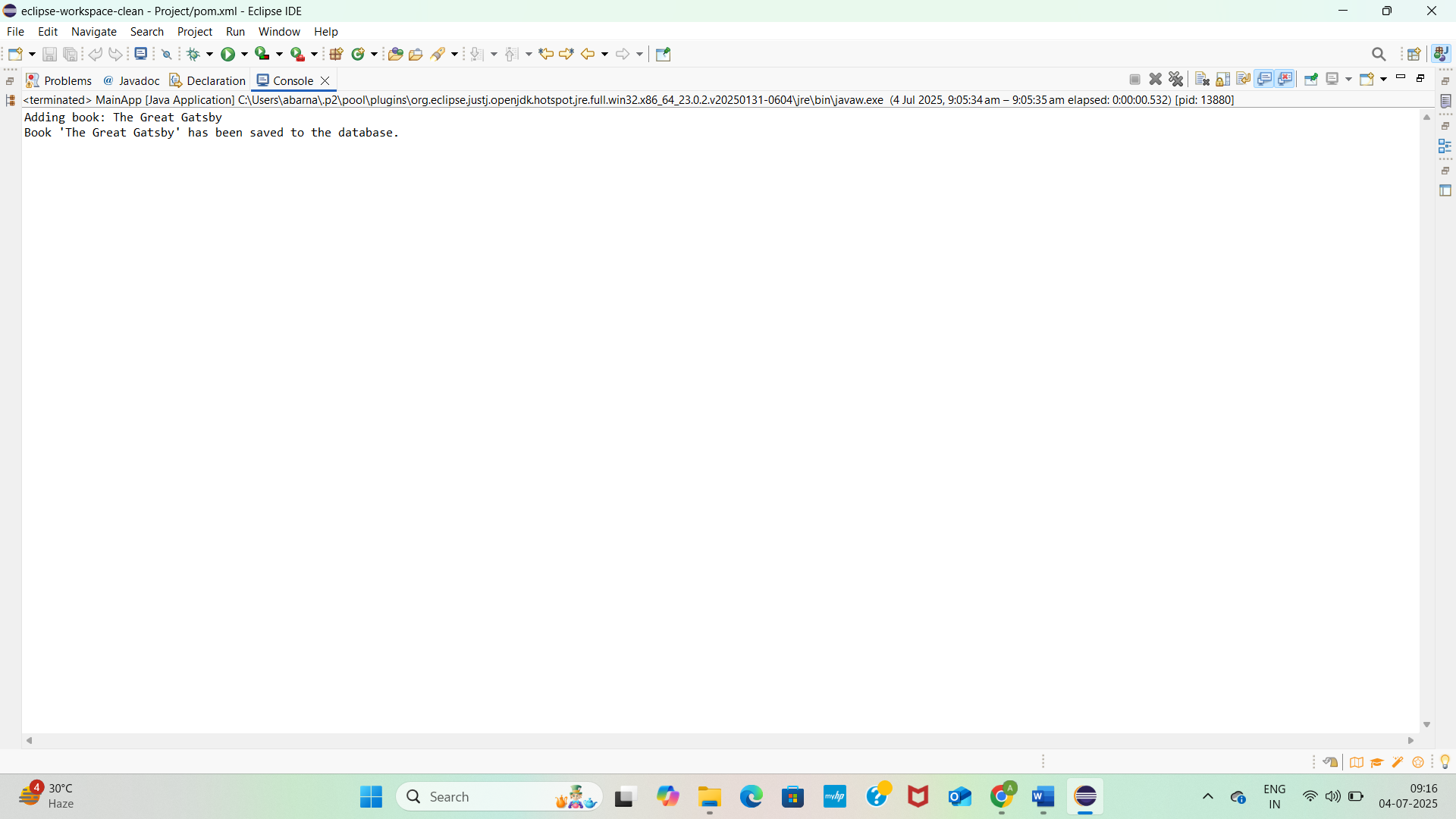
</plugin>

</plugins>

</build>

</project>

OUTPUT SCREENSHOT

3.

ApplicationContext.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans

<https://www.springframework.org/schema/beans/spring-beans.xsd>

<http://www.springframework.org/schema/aop>

<https://www.springframework.org/schema/aop/spring-aop.xsd>">

<!-- Enable AspectJ Auto Proxy -->

<aop:aspectj-autoproxy />

<!-- Repository and Service Beans -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

<!-- Logging Aspect Bean -->

<bean id="loggingAspect" class="com.library.aspect.LoggingAspect" />

</beans>

MainApp.java

**package** com.library.app;

**import** com.library.service.BookService;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** MainApp {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("The Great Gatsby");

}

}

LoggingAspect.java

**package** com.library.aspect;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.Around;

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

@Aspect

**public** **class** LoggingAspect {

@Around("execution(\* com.library.service.\*.\*(..))")

**public** Object logExecutionTime(ProceedingJoinPoint joinPoint) **throws** Throwable {

**long** start = System.*currentTimeMillis*();

Object result = joinPoint.proceed(); // Proceed with method

**long** duration = System.*currentTimeMillis*() - start;

System.***out***.println("[LOG] " + joinPoint.getSignature() + " executed in " + duration + "ms");

**return** result;

}

}

BookRepository.java

**package** com.library.repository;

**public** **class** BookRepository {

**public** **void** save(String title) {

System.***out***.println("Book '" + title + "' has been saved to the database.");

}

}

BookService.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for DI

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

System.***out***.println("Adding book: " + title);

bookRepository.save(title);

}

}

Pom.xml

< 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

<http://maven.apache.org/xsd/maven-4.0.0.xsd>">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Core Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<!-- AspectJ Weaver (Required for @Aspect to work) -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

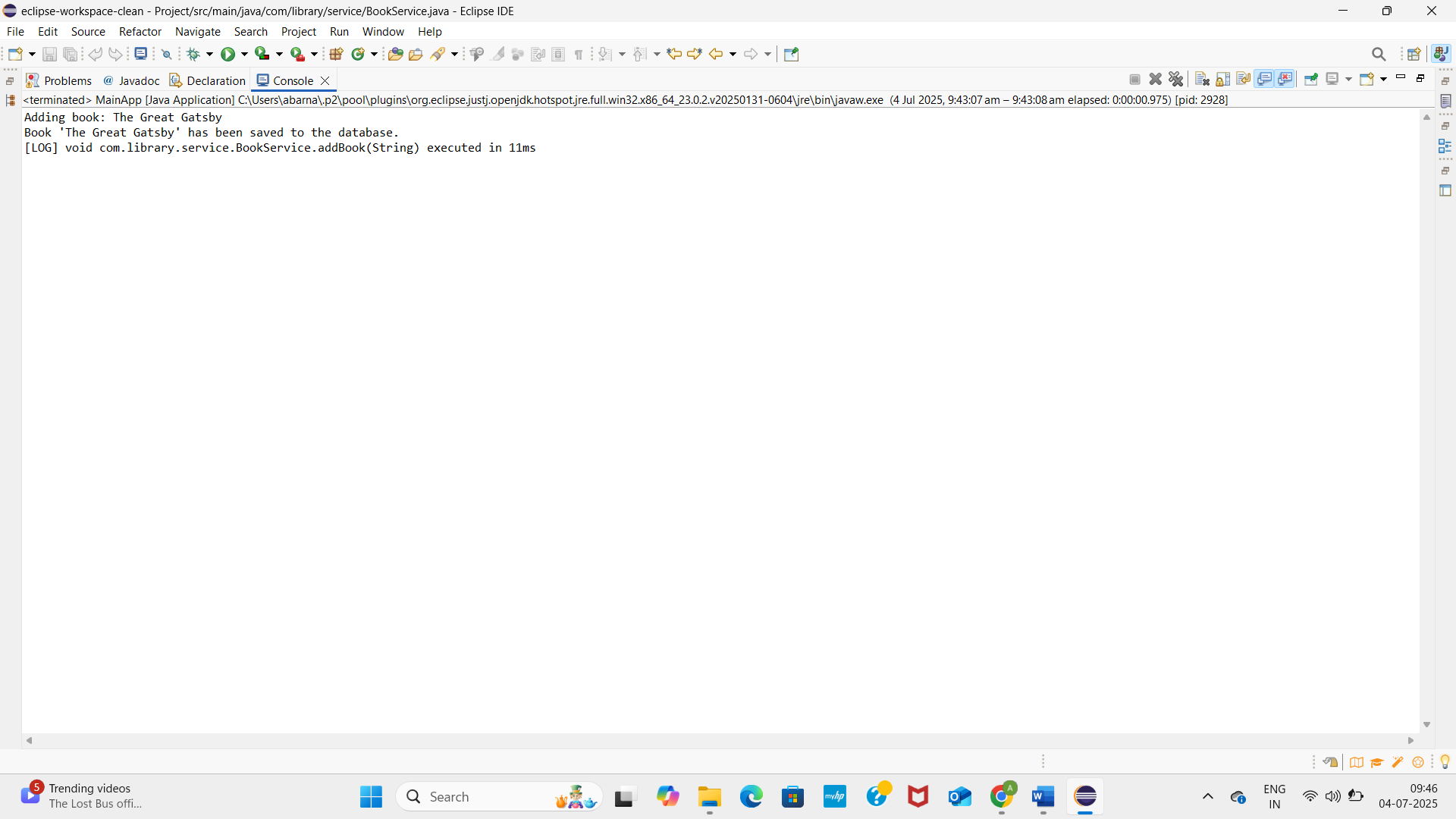
<version>1.9.21</version>

</dependency>

</dependencies>

</project>

OUTPUTSCREENHOT

4.

Pom.xml

<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

<http://maven.apache.org/xsd/maven-4.0.0.xsd>">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<java.version>1.8</java.version>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

ApplicationContext.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans

<https://www.springframework.org/schema/beans/spring-beans.xsd>

<http://www.springframework.org/schema/aop>

<https://www.springframework.org/schema/aop/spring-aop.xsd>">

<!-- Enable AspectJ Auto Proxy -->

<aop:aspectj-autoproxy />

<!-- Repository and Service Beans -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

<!-- Logging Aspect Bean -->

<bean id="loggingAspect" class="com.library.aspect.LoggingAspect" />

</beans>

MainApp.java

**package** com.library.app;

**import** com.library.service.BookService;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** MainApp {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("The Great Gatsby");

}

}

LoggingAspect.java

**package** com.library.aspect;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.Around;

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

@Aspect

**public** **class** LoggingAspect {

@Around("execution(\* com.library.service.\*.\*(..))")

**public** Object logExecutionTime(ProceedingJoinPoint joinPoint) **throws** Throwable {

**long** start = System.*currentTimeMillis*();

Object result = joinPoint.proceed(); // Proceed with method

**long** duration = System.*currentTimeMillis*() - start;

System.***out***.println("[LOG] " + joinPoint.getSignature() + " executed in " + duration + "ms");

**return** result;

}

}

BookRepository.java

**package** com.library.repository;

**public** **class** BookRepository {

**public** **void** save(String title) {

System.***out***.println("Book '" + title + "' has been saved to the database.");

}

}

BookService.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for DI

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

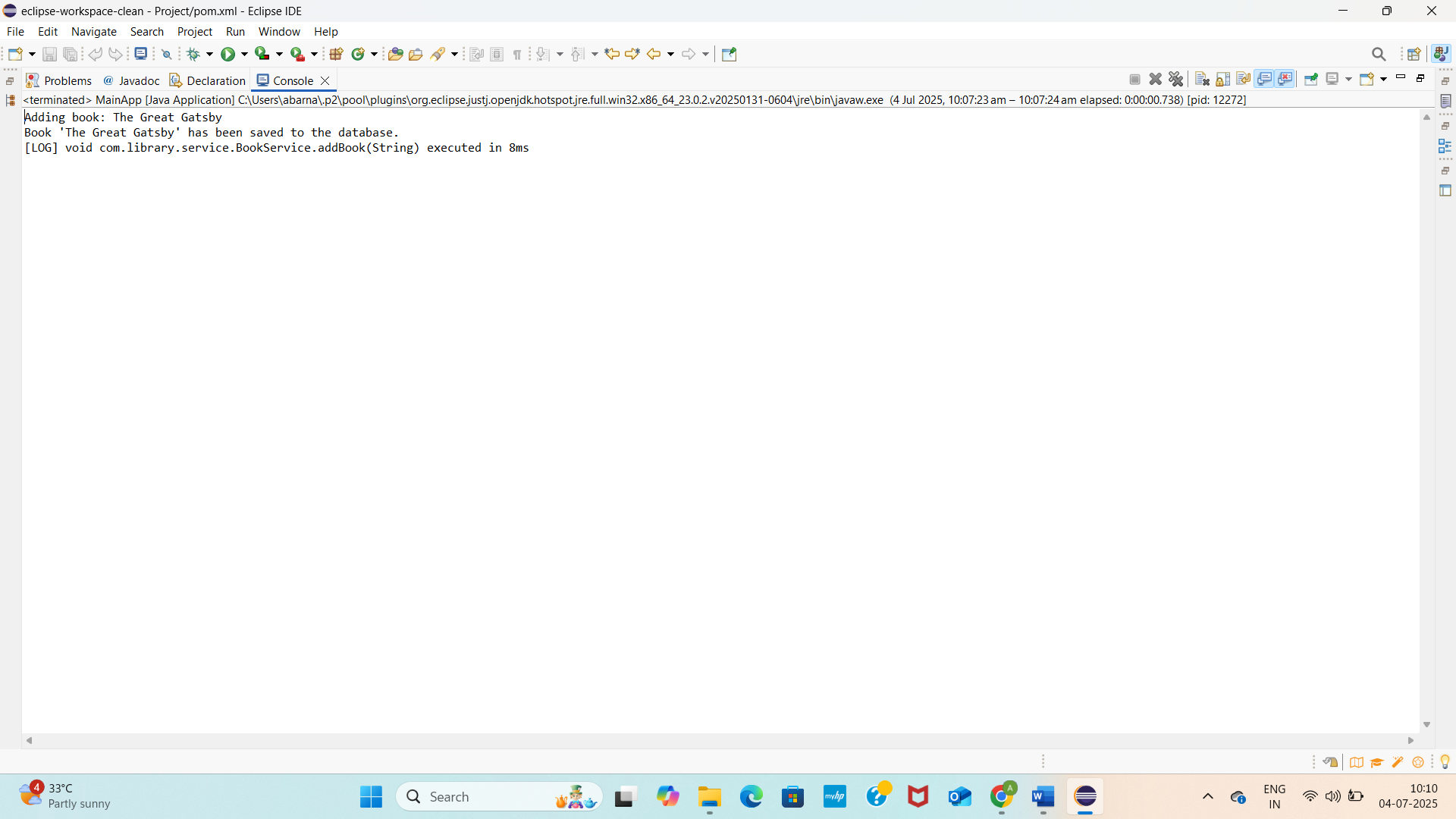
System.***out***.println("Adding book: " + title);

bookRepository.save(title);

}

}

OUTPUTSCREENSHOT

5.

ApplicationContext.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

<https://www.springframework.org/schema/beans/spring-beans.xsd>">

<!-- BookRepository Bean -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<!-- BookService Bean with Dependency Injection -->

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

Pom.xml

<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

<http://maven.apache.org/xsd/maven-4.0.0.xsd>">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<java.version>1.8</java.version>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

MainApp.java

**package** com.library.app;

**import** com.library.service.BookService;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** MainApp {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("The Great Gatsby");

}

}

LoggingAspect.java

**package** com.library.aspect;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.Around;

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

@Aspect

**public** **class** LoggingAspect {

@Around("execution(\* com.library.service.\*.\*(..))")

**public** Object logExecutionTime(ProceedingJoinPoint joinPoint) **throws** Throwable {

**long** start = System.*currentTimeMillis*();

Object result = joinPoint.proceed(); // Proceed with method

**long** duration = System.*currentTimeMillis*() - start;

System.***out***.println("[LOG] " + joinPoint.getSignature() + " executed in " + duration + "ms");

**return** result;

}

}

BookRepository.java

**package** com.library.repository;

**public** **class** BookRepository {

**public** **void** save(String title) {

System.***out***.println("Book '" + title + "' has been saved to the database.");

}

}

BookService.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for DI

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

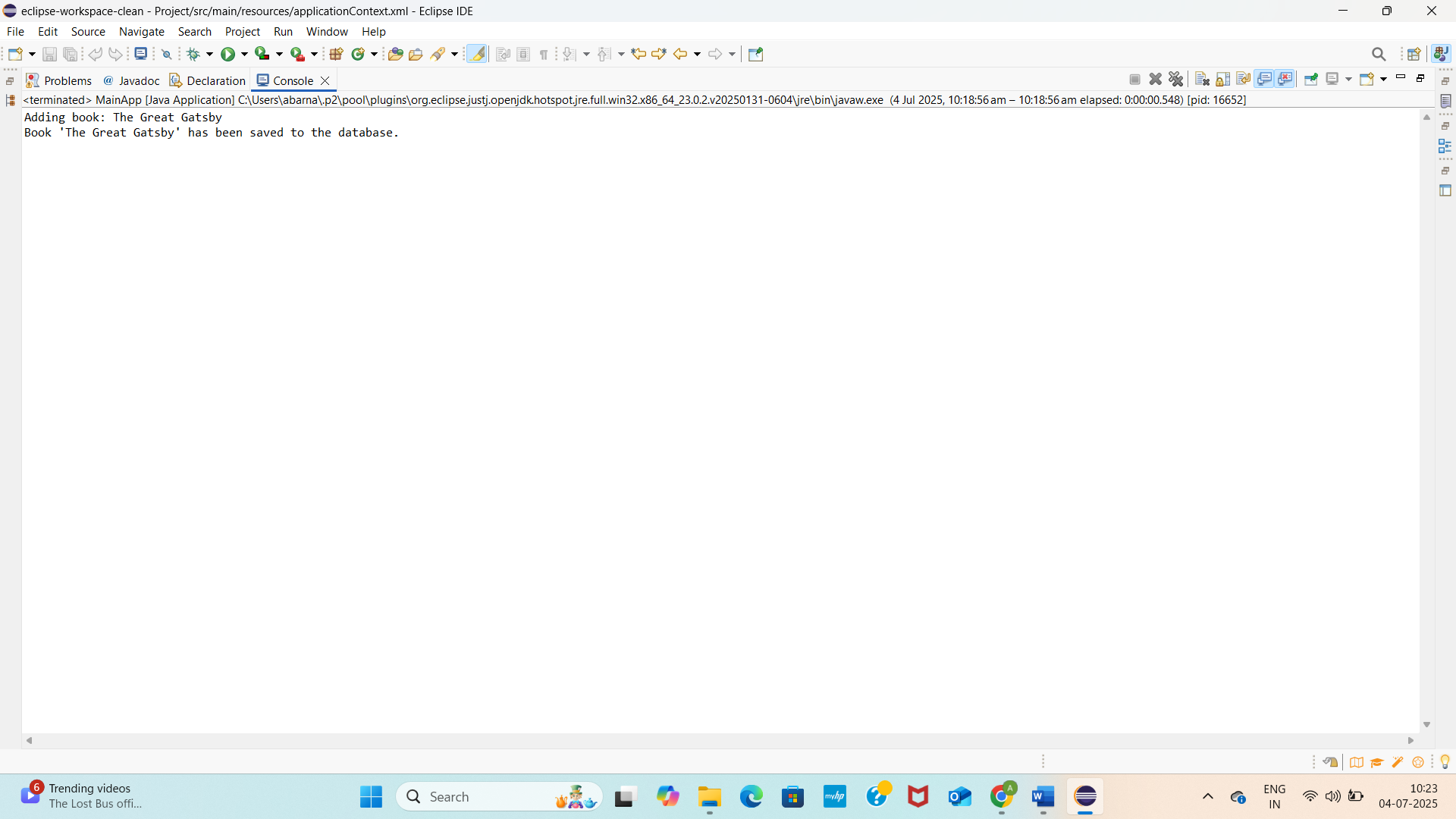
System.***out***.println("Adding book: " + title);

bookRepository.save(title);

}

}

OUTPUT SCREENSHOT



6.

ApplicationContext.xml

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

<beans xmlns="<http://www.springframework.org/schema/beans>"

xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xmlns:context="<http://www.springframework.org/schema/context>"

xsi:schemaLocation="

<http://www.springframework.org/schema/beans>

<https://www.springframework.org/schema/beans/spring-beans.xsd>

<http://www.springframework.org/schema/context>

<https://www.springframework.org/schema/context/spring-context.xsd>">

<!-- Enable component scanning -->

<context:component-scan base-package="com.library" />

</beans>

BookRepository.java

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

public void save(String title) {

System.out.println("Book '" + title + "' has been saved to the database.");

}

}

BookService,java

package com.library.service;

import com.library.repository.BookRepository;

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

import org.springframework.stereotype.Service;

@Service

public class BookService {

private BookRepository bookRepository;

@Autowired

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String title) {

System.out.println("Adding book: " + title);

bookRepository.save(title);

}

}

MainApp.java

**package** com.library.app;

**import** com.library.service.BookService;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** MainApp {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

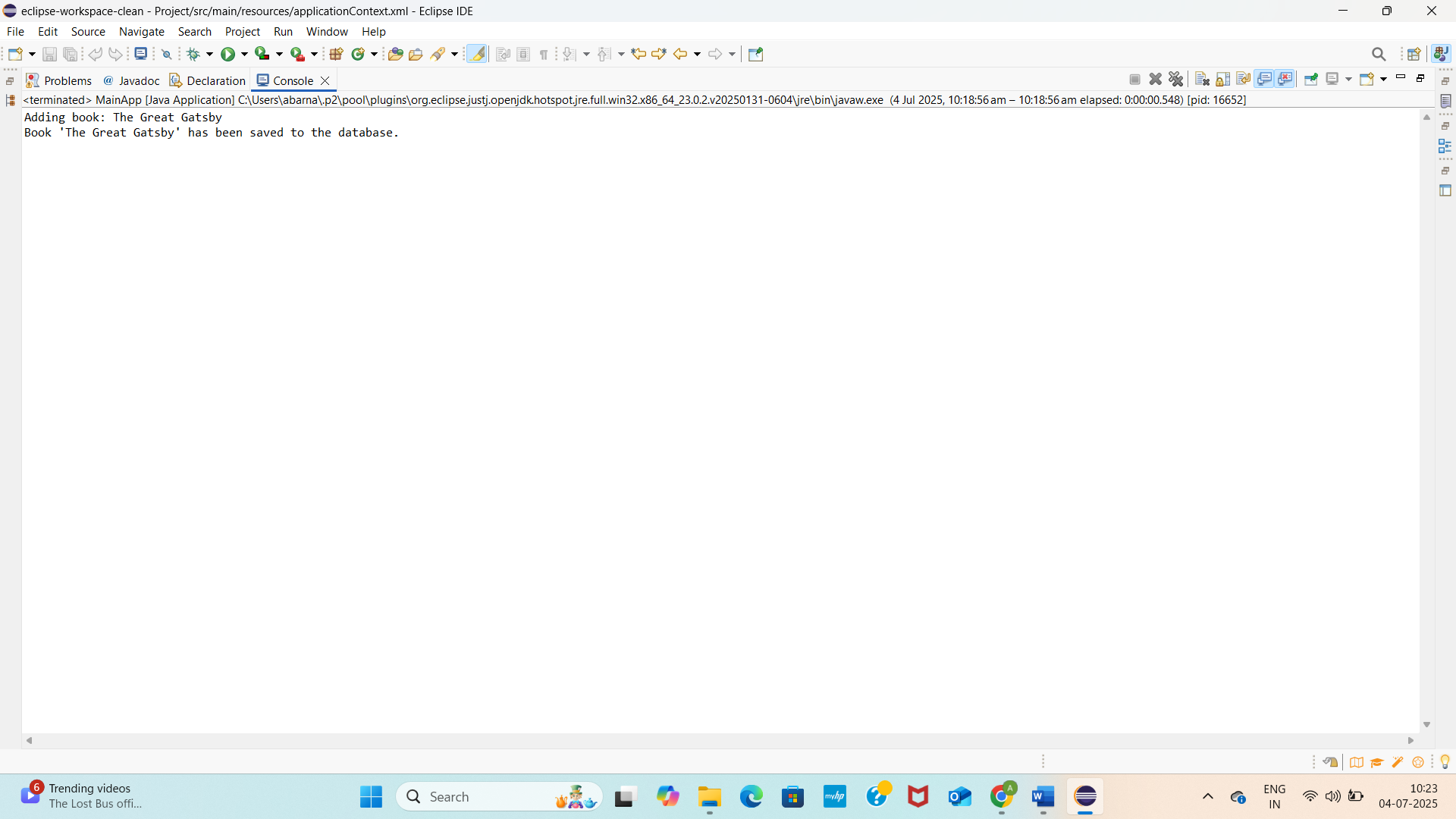
BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("The Great Gatsby");

}

}

OUTPUTSCREENSHOT



7.BookService.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

private String serviceName;

// Constructor injection

public BookService(String serviceName) {

this.serviceName = serviceName;

}

// Setter injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String title) {

System.out.println("[" + serviceName + "] Adding book: " + title);

bookRepository.save(title);

}

}

BookRepository.java

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

public void save(String title) {

System.out.println("Book '" + title + "' has been saved to the database.");

}

}

ApplicationContext.xml

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

<beans xmlns="<http://www.springframework.org/schema/beans>"

xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation="

<http://www.springframework.org/schema/beans>

<https://www.springframework.org/schema/beans/spring-beans.xsd>">

<!-- BookRepository Bean -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<!-- BookService Bean with both constructor and setter injection -->

<bean id="bookService" class="com.library.service.BookService">

<!-- Constructor injection -->

<constructor-arg value="LibraryService"/>

<!-- Setter injection -->

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

|  |  |
| --- | --- |
|  | Mainapp.java |

package [com.library.app](http://com.library.app/);

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

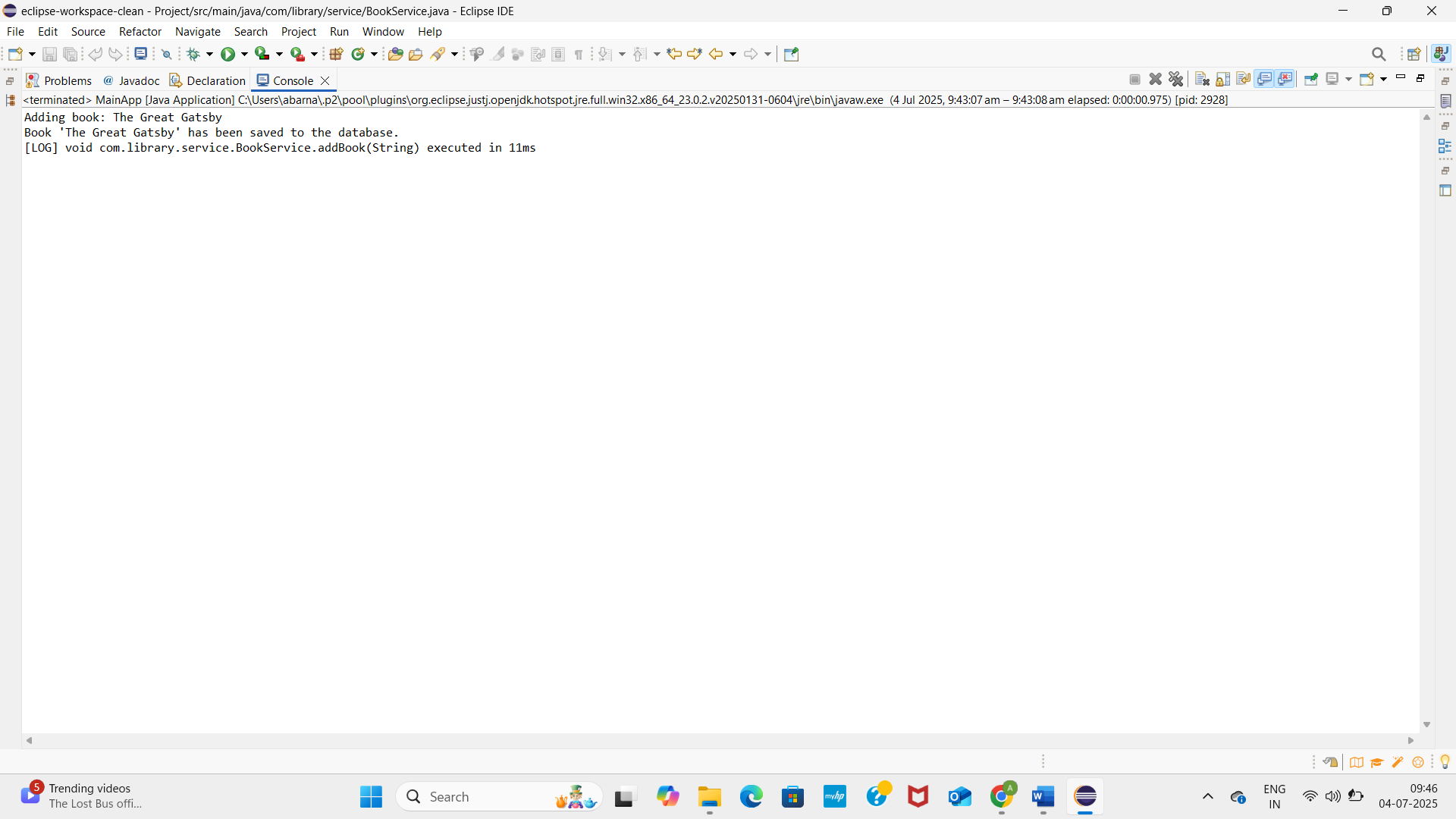
BookService bookService = context.getBean(BookService.class);

bookService.addBook("The Great Gatsby");

}

}

OUTPUTSCREENSHOT



8.applicationContext.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="

<http://www.springframework.org/schema/beans> <https://www.springframework.org/schema/beans/spring-beans.xsd>

<http://www.springframework.org/schema/context> <https://www.springframework.org/schema/context/spring-context.xsd>

<http://www.springframework.org/schema/aop> <https://www.springframework.org/schema/aop/spring-aop.xsd>">

<!-- Enable component scanning -->

<context:component-scan base-package="com.library" />

<!-- Enable AOP auto proxying -->

<aop:aspectj-autoproxy />

</beans>

MainApp.java

**package** com.library.app;

**import** com.library.service.BookService;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** MainApp {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = context.getBean(BookService.**class**);

bookService.addBook("The Great Gatsby");

}

}

LoggingAspect.java

**package** com.library.aspect;

**import** org.aspectj.lang.annotation.After;

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

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

**import** org.springframework.stereotype.Component;

@Aspect

@Component

**public** **class** LoggingAspect {

@Before("execution(\* com.library.service.BookService.addBook(..))")

**public** **void** logBefore() {

System.***out***.println("🔍 [LOG] Method addBook() is about to be called");

}

@After("execution(\* com.library.service.BookService.addBook(..))")

**public** **void** logAfter() {

System.***out***.println("✅ [LOG] Method addBook() has finished execution");

}

}

BookRepository.java

**package** com.library.repository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **class** BookRepository {

**public** **void** save(String title) {

System.***out***.println("💾 Book '" + title + "' has been saved.");

}

}

BookService.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

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

**import** org.springframework.stereotype.Service;

@Service

**public** **class** BookService {

**private** BookRepository bookRepository;

@Autowired

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String title) {

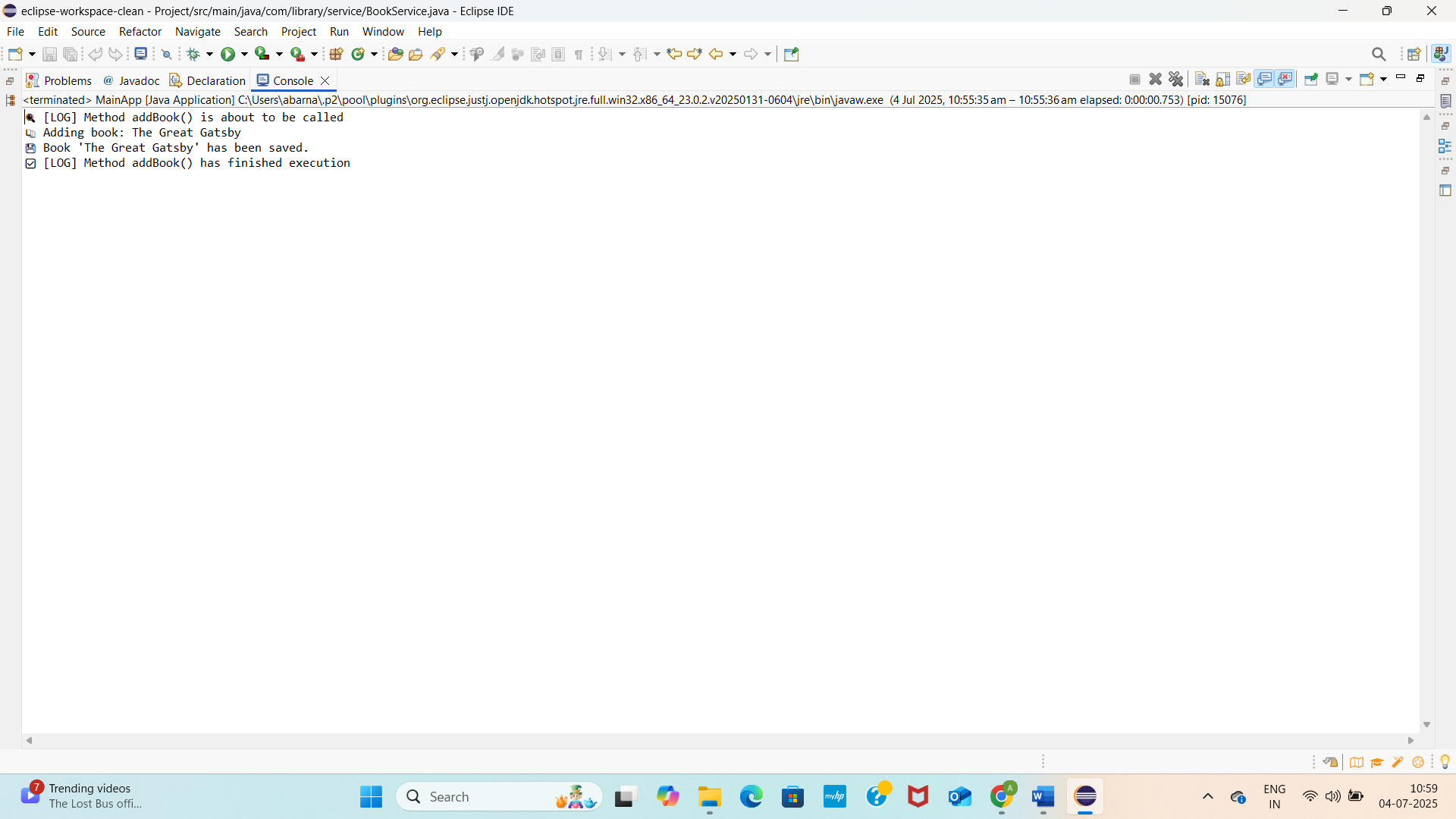
System.***out***.println("📚 Adding book: " + title);

bookRepository.save(title);

}

}

OUTPUTSCREENSHOT



|  |  |
| --- | --- |
|  | 9.  LibraryManagementApplication  **package** com.lms;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  **public** **class** LibraryManagementApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(LibraryManagementApplication.**class**, args);  }  }  BookRestController.java  **package** com.lms.controller;  **import** com.lms.entity.Book;  **import** com.lms.service.BookService;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.web.bind.annotation.\*;  **import** java.util.List;  **import** java.util.Optional;  @RestController  @RequestMapping("/api/books")  **public** **class** BookRestController {  @Autowired  **private** BookService bookService;  @GetMapping  **public** List<Book> getBooks() {  **return** bookService.getAllBooks();  }  @GetMapping("/{id}")  **public** Optional<Book> getBook(@PathVariable Long id) {  **return** bookService.getBookById(id);  }  @PostMapping  **public** Book createBook(@RequestBody Book book) {  **return** bookService.saveBook(book);  }  @PutMapping("/{id}")  **public** Book updateBook(@PathVariable Long id, @RequestBody Book book) {  book.setId(id);  **return** bookService.saveBook(book);  }  @DeleteMapping("/{id}")  **public** **void** deleteBook(@PathVariable Long id) {  bookService.deleteBook(id);  }  }  Book.java  **package** com.lms.entity;  **import** jakarta.persistence.\*;  @Entity  **public** **class** Book {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  **private** Long id;  **private** String title;  **private** String author;  **private** String isbn;  // 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;  }  **public** String getIsbn() {  **return** isbn;  }  **public** **void** setIsbn(String isbn) {  **this**.isbn = isbn;  }  }  BookDataRepository.java  **package** com.lms.repository;  **import** com.lms.entity.Book;  **import** org.springframework.data.jpa.repository.JpaRepository;  **public** **interface** BookDataRepository **extends** JpaRepository<Book, Long> {  }  BookService.java  **package** com.lms.service;  **import** com.lms.entity.Book;  **import** com.lms.repository.BookDataRepository;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** java.util.List;  **import** java.util.Optional;  @Service  **public** **class** BookService {  @Autowired  **private** BookDataRepository bookRepo;  **public** List<Book> getAllBooks() {  **return** bookRepo.findAll();  }  **public** Optional<Book> getBookById(Long id) {  **return** bookRepo.findById(id);  }  **public** Book saveBook(Book book) {  **return** bookRepo.save(book);  }  **public** **void** deleteBook(Long id) {  bookRepo.deleteById(id);  }  }  Pom.xml  <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  <http://maven.apache.org/xsd/maven-4.0.0.xsd>">  <modelVersion>4.0.0</modelVersion>  <groupId>com.lms</groupId>  <artifactId>LibraryManagement</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>LibraryManagement</name>  <description>Library Management System</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>3.2.4</version> <!-- or latest stable version -->  <relativePath/>  </parent>  <properties>  <java.version>17</java.version> <!-- Use 17 or match your setup -->  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <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>  OUTPUTSCREENSHOT    SPRING DATA JPA-QUICK EXAMPLE  Application.properties  # Use H2 in-memory database  spring.datasource.url=jdbc:h2:mem:testdb  spring.datasource.driver-class-name=org.h2.Driver  spring.datasource.username=sa  spring.datasource.password=  # Automatically run schema.sql and data.sql  spring.datasource.initialization-mode=always  spring.jpa.hibernate.ddl-auto=none  spring.jpa.show-sql=true  # Enable H2 console  spring.h2.console.enabled=true  spring.h2.console.path=/h2-console  Data.sql  INSERT INTO country VALUES('IN','India');  INSERT INTO country VALUES('US','United States of America');  Schema.sql  CREATE TABLE country (  co\_code VARCHAR(2) PRIMARY KEY,  co\_name VARCHAR(50)  );  OrmLearnApplication  **package** com.cognizant.ormlearn1;  **import** java.util.List;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.context.ApplicationContext;  **import** com.cognizant.ormlearn1.model.Country;  **import** com.cognizant.ormlearn1.service.CountryService;  @SpringBootApplication  **public** **class** OrmLearnApplication {  **private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(OrmLearnApplication.**class**);  **private** **static** CountryService *countryService*;  **public** **static** **void** main(String[] args) {  ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.**class**, args);  ***LOGGER***.info("Inside main");  *countryService* = context.getBean(CountryService.**class**);  *testGetAllCountries*();  }  **private** **static** **void** testGetAllCountries() {  ***LOGGER***.info("Start");  List<Country> countries = *countryService*.getAllCountries();  ***LOGGER***.debug("countries={}", countries);  ***LOGGER***.info("End");  }  }  Country  **package** com.cognizant.ormlearn1.model;  **import** jakarta.persistence.Column;  **import** jakarta.persistence.Entity;  **import** jakarta.persistence.Id;  **import** jakarta.persistence.Table;  @Entity  @Table(name = "country")  **public** **class** Country {  @Id  @Column(name = "co\_code")  **private** String code;  @Column(name = "co\_name")  **private** String name;  // Getters and Setters  **public** String getCode() {  **return** code;  }  **public** **void** setCode(String code) {  **this**.code = code;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  // toString()  @Override  **public** String toString() {  **return** "Country [code=" + code + ", name=" + name + "]";  }  }  CountryRepository  **package** com.cognizant.ormlearn1.repository;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** org.springframework.stereotype.Repository;  **import** com.cognizant.ormlearn1.model.Country;  @Repository  **public** **interface** CountryRepository **extends** JpaRepository<Country, String> {  }  CountryService  **package** com.cognizant.ormlearn1.service;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** org.springframework.transaction.annotation.Transactional;  **import** com.cognizant.ormlearn1.model.Country;  **import** com.cognizant.ormlearn1.repository.CountryRepository;  @Service  **public** **class** CountryService {  @Autowired  **private** CountryRepository countryRepository;  @Transactional  **public** List<Country> getAllCountries() {  **return** countryRepository.findAll();  }  }  OUTPUTSCREENSHOT      2.IMPLEMENT SERVICES FOR MANAGING COUNTRY  Country.java  **package** com.example.countryapp;  **import** jakarta.persistence.\*;  @Entity  @Table(name = "country")  **public** **class** Country {  @Id  @Column(name = "co\_code", length = 5)  **private** String code;  @Column(name = "co\_name", nullable = **false**)  **private** String name;  // Constructors  **public** Country() {}  **public** Country(String code, String name) {  **this**.code = code;  **this**.name = name;  }  // Getters and Setters  **public** String getCode() { **return** code; }  **public** **void** setCode(String code) { **this**.code = code; }  **public** String getName() { **return** name; }  **public** **void** setName(String name) { **this**.name = name; }  }  CountryAppApplication.java  **package** com.example.countryapp;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  **public** **class** CountryAppApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(CountryAppApplication.**class**, args);  }  }  CountryController.java  **package** com.example.countryapp;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.web.bind.annotation.\*;  **import** java.util.List;  **import** java.util.Optional;  @RestController  @RequestMapping("/api/countries")  **public** **class** CountryController {  @Autowired  **private** CountryService service;  @GetMapping("/{code}")  **public** Optional<Country> getByCode(@PathVariable String code) {  **return** service.getCountryByCode(code);  }  @PostMapping  **public** Country addCountry(@RequestBody Country country) {  **return** service.addCountry(country);  }  @PutMapping  **public** Country updateCountry(@RequestBody Country country) {  **return** service.updateCountry(country);  }  @DeleteMapping("/{code}")  **public** **void** deleteCountry(@PathVariable String code) {  service.deleteCountry(code);  }  @GetMapping("/search")  **public** List<Country> search(@RequestParam("name") String name) {  **return** service.searchByName(name);  }  }  CountryRepository.java  **package** com.example.countryapp;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** java.util.List;  **public** **interface** CountryRepository **extends** JpaRepository<Country, String> {  List<Country> findByNameContainingIgnoreCase(String name);  }  CountryService.java  **package** com.example.countryapp;  **import** java.util.List;  **import** java.util.Optional;  **public** **interface** CountryService {  Optional<Country> getCountryByCode(String code);  Country addCountry(Country country);  Country updateCountry(Country country);  **void** deleteCountry(String code);  List<Country> searchByName(String namePart);  }  CountryServiceImpl.java  **package** com.example.countryapp;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** java.util.List;  **import** java.util.Optional;  @Service  **public** **class** CountryServiceImpl **implements** CountryService {  @Autowired  **private** CountryRepository repository;  @Override  **public** Optional<Country> getCountryByCode(String code) {  **return** repository.findById(code);  }  @Override  **public** Country addCountry(Country country) {  **return** repository.save(country);  }  @Override  **public** Country updateCountry(Country country) {  **return** repository.save(country);  }  @Override  **public** **void** deleteCountry(String code) {  repository.deleteById(code);  }  @Override  **public** List<Country> searchByName(String namePart) {  **return** repository.findByNameContainingIgnoreCase(namePart);  }  }  Data.sql  INSERT INTO country (co\_code, co\_name) VALUES ('AF', 'Afghanistan');  -- rest follows...  Schema.sql  CREATE TABLE country (  co\_code VARCHAR(5) PRIMARY KEY,  co\_name VARCHAR(255) NOT NULL  );  OUTPUTSCREENSHOT      3.DIFFERENCE BETWEEN JPA,HIBERNATE,SPRING DATA JPA  Java Persistence API (JPA)  What it is: A specification (standard) for object-relational mapping (ORM) in Java.  Defined by: JSR 338.  Purpose: Allows you to persist, retrieve, update, and delete Java objects (entities) from a relational database.  Important: JPA is just an interface/contract – it doesn't do anything by itself.  Requires: An implementation like Hibernate, EclipseLink, etc.  Hibernate  What it is: A popular implementation of JPA.  Also: Provides extra features beyond the JPA spec.  Manages: SessionFactory, Sessions, Transactions manually.  You write more boilerplate code (e.g., session management, transactions).  🔸 Hibernate Example:  public Integer addEmployee(Employee employee){  Session session = factory.openSession();  Transaction tx = null;  Integer employeeID = null;  try {  tx = session.beginTransaction();  employeeID = (Integer) session.save(employee);  tx.commit();  } catch (HibernateException e) {  if (tx != null) tx.rollback();  e.printStackTrace();  } finally {  session.close();  }  return employeeID;  }  Spring Data JPA  What it is: A Spring module that simplifies the use of JPA.  Abstraction: It sits on top of JPA and a provider like Hibernate.  Eliminates boilerplate: No need to write code for sessions, transactions, or even queries in most cases.  Built-in repositories: Interfaces like JpaRepository, CrudRepository are provided.  🔸 Spring Data JPA Example:  public interface EmployeeRepository extends JpaRepository<Employee, Integer> { }  @Service  public class EmployeeService {  @Autowired  private EmployeeRepository employeeRepository;  @Transactional  public void addEmployee(Employee employee) {  employeeRepository.save(employee);  }  }  JPA is a Java specification for ORM (defines interfaces, no implementation).  Hibernate is a JPA implementation and ORM tool with extra features.  Spring Data JPA is a Spring abstraction that simplifies JPA and reduces boilerplate code  4.FIND A COUNTRY BASED ON COUNTRY CODE  OrmLearnApplication.java  **package** com.cognizant.springlearn;  **import** com.cognizant.springlearn.model.Country;  **import** com.cognizant.springlearn.service.CountryService;  **import** com.cognizant.springlearn.service.exception.CountryNotFoundException;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.context.ApplicationContext;  @SpringBootApplication  **public** **class** OrmLearnApplication {  **private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(OrmLearnApplication.**class**);  **private** **static** CountryService *countryService*;  **public** **static** **void** main(String[] args) **throws** CountryNotFoundException {  ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.**class**, args);  OrmLearnApplication.*countryService* = context.getBean(CountryService.**class**);  *getCountryByCodeTest*(); // Call the test method  }  **private** **static** **void** getCountryByCodeTest() {  ***LOGGER***.info("Start");  **try** {  Country country = *countryService*.findCountryByCode("IN");  ***LOGGER***.debug("Country: {}", country);  } **catch** (CountryNotFoundException e) {  ***LOGGER***.error("Exception: {}", e.getMessage());  }  ***LOGGER***.info("End");  }  }  Country.java  **package** com.cognizant.springlearn.model;  **import** jakarta.persistence.\*;  @Entity  @Table(name = "country")  **public** **class** Country {  @Id  @Column(name = "co\_code")  **private** String code;  @Column(name = "co\_name")  **private** String name;  **public** Country() {  }  **public** Country(String code, String name) {  **this**.code = code;  **this**.name = name;  }  **public** String getCode() {  **return** code;  }  **public** **void** setCode(String code) {  **this**.code = code;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  @Override  **public** String toString() {  **return** "Country [code=" + code + ", name=" + name + "]";  }  }  CountryRepositor.java  **package** com.cognizant.springlearn.repository;  **import** com.cognizant.springlearn.model.Country;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** org.springframework.stereotype.Repository;  @Repository  **public** **interface** CountryRepository **extends** JpaRepository<Country, String> {  }  CountryService.java  **package** com.cognizant.springlearn.service;  **import** com.cognizant.springlearn.model.Country;  **import** com.cognizant.springlearn.service.exception.CountryNotFoundException;  **import** com.cognizant.springlearn.repository.CountryRepository;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** org.springframework.transaction.annotation.Transactional;  **import** java.util.Optional;  @Service  **public** **class** CountryService {  @Autowired  **private** CountryRepository countryRepository;  @Transactional  **public** Country findCountryByCode(String countryCode) **throws** CountryNotFoundException {  Optional<Country> result = countryRepository.findById(countryCode);  **if** (!result.isPresent()) {  **throw** **new** CountryNotFoundException("Country with code " + countryCode + " not found");  }  **return** result.get();  }  }  CountryNotFoundException.java  **package** com.cognizant.springlearn.service.exception;  **public** **class** CountryNotFoundException **extends** Exception {  **public** CountryNotFoundException(String message) {  **super**(message);  }  }  Application.properties  # Enable H2 Console  spring.h2.console.enabled=true  spring.h2.console.path=/h2-console  # DB Configuration  spring.datasource.url=jdbc:h2:mem:testdb  spring.datasource.driver-class-name=org.h2.Driver  spring.datasource.username=sa  spring.datasource.password=  # Run SQL files  spring.datasource.initialization-mode=always  spring.jpa.hibernate.ddl-auto=none  # Show SQL  spring.jpa.show-sql=true  Data.sql  INSERT INTO country (co\_code, co\_name) VALUES ('IN', 'India');  INSERT INTO country (co\_code, co\_name) VALUES ('US','United States');  INSERT INTO country (co\_code, co\_name) VALUES ('DE', 'Germany');-- rest follows...  Schema.sql  CREATE TABLE country (  co\_code **VARCHAR**(5) **PRIMARY KEY**,  co\_name **VARCHAR**(50) NOT NULL  );  OUTPUTSCREENSHOT      5.DD A NEW COUNTRY  Country.java  **package** com.example.countryapp;  **import** jakarta.persistence.\*;  @Entity  @Table(name = "country")  **public** **class** Country {  @Id  @Column(name = "co\_code", length = 5)  **private** String code;  @Column(name = "co\_name", nullable = **false**)  **private** String name;  // Constructors  **public** Country() {}  **public** Country(String code, String name) {  **this**.code = code;  **this**.name = name;  }  // Getters and Setters  **public** String getCode() {  **return** code;  }  **public** **void** setCode(String code) {  **this**.code = code;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  }  CountryAppApplication.java  **package** com.example.countryapp;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  **public** **class** CountryAppApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(CountryAppApplication.**class**, args);  }  }  CountryNotFoundException.java  **package** com.example.countryapp;  **public** **class** CountryNotFoundException **extends** RuntimeException {  **public** CountryNotFoundException(String message) {  **super**(message);  }  }  CountryRepository.java  **package** com.example.countryapp;  **import** org.springframework.data.jpa.repository.JpaRepository;  **public** **interface** CountryRepository **extends** JpaRepository<Country, String> {  }  CountryService.java  **package** com.example.countryapp;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** org.springframework.transaction.annotation.Transactional;  **import** java.util.Optional;  @Service  **public** **class** CountryService {  @Autowired  **private** CountryRepository countryRepository;  @Transactional  **public** **void** addCountry(Country country) {  countryRepository.save(country);  }  **public** Country findCountryByCode(String code) {  Optional<Country> optional = countryRepository.findById(code);  **if** (optional.isEmpty()) {  **throw** **new** CountryNotFoundException("Country not found: " + code);  }  **return** optional.get();  }  }    OrmLearnApplication.java  **package** com.example.countryapp;  **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** OrmLearnApplication **implements** CommandLineRunner {  @Autowired  **private** CountryService countryService;  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(OrmLearnApplication.**class**, args);  }  @Override  **public** **void** run(String... args) **throws** Exception {  testAddCountry();  }  **private** **void** testAddCountry() {  Country country = **new** Country();  country.setCode("JP");  country.setName("Japan");  countryService.addCountry(country);  Country result = countryService.findCountryByCode("JP");  System.***out***.println("Country added: " + result.getCode() + " - " + result.getName());  }  }  Application.properties  # H2 Database configuration  spring.datasource.url=jdbc:h2:mem:testdb  spring.datasource.driver-class-name=org.h2.Driver  spring.datasource.username=sa  spring.datasource.password=  # Automatically run schema and data scripts  spring.datasource.initialization-mode=always  spring.jpa.hibernate.ddl-auto=none  spring.jpa.show-sql=true  # H2 Console  spring.h2.console.enabled=true  spring.h2.console.path=/h2-console  Schema.sql  CREATE TABLE country (  co\_code **VARCHAR**(5) **PRIMARY KEY**,  co\_name **VARCHAR**(50) NOT NULL  );  Data.sql  INSERT INTO country (co\_code, co\_name) VALUES ('IN', 'India');  INSERT INTO country (co\_code, co\_name) VALUES ('US', 'United States');  OUTPUT SCREENSHOT      SPRING DATA JPA HANDSON 2  DEMONSTRATE IMPLEEMENTATION  Product.java  **package** com.example.demo.entity;  **import** jakarta.persistence.\*;  **import** java.time.LocalDate;  @Entity  **public** **class** Product {  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** Long id;  **private** String name;  **private** Double price;  **private** LocalDate createdDate;  // ✅ Default constructor required by JPA  **public** Product() {}  // ✅ Custom constructor used in DemoApplication  **public** Product(String name, Double price, LocalDate createdDate) {  **this**.name = name;  **this**.price = price;  **this**.createdDate = createdDate;  }  // ✅ Getters and 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; }  **public** LocalDate getCreatedDate() { **return** createdDate; }  **public** **void** setCreatedDate(LocalDate createdDate) { **this**.createdDate = createdDate; }  // ✅ Custom toString for readable output  @Override  **public** String toString() {  **return** id + " - " + name + " - " + price + " - " + createdDate;  }  }  DemoApplication.java  **package** com.example.demo;  **import** com.example.demo.entity.Product;  **import** com.example.demo.repository.ProductRepository;  **import** org.springframework.boot.CommandLineRunner;  **import** org.springframework.context.annotation.Bean;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.boot.SpringApplication;  **import** java.time.LocalDate;  @SpringBootApplication  **public** **class** DemoApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(DemoApplication.**class**, args);  }  @Bean  **public** CommandLineRunner demo(ProductRepository repo) {  **return** args -> {  // Save products  repo.save(**new** Product("Apple iPhone", 799.0, LocalDate.*of*(2023, 5, 20)));  repo.save(**new** Product("Samsung Galaxy", 699.0, LocalDate.*of*(2023, 6, 10)));  repo.save(**new** Product("Nokia Phone", 299.0, LocalDate.*of*(2023, 4, 5)));  repo.save(**new** Product("Google Pixel", 899.0, LocalDate.*of*(2023, 7, 1)));  repo.save(**new** Product("Realme Note", 199.0, LocalDate.*of*(2023, 3, 25)));  System.***out***.println("=== Containing 'Phone' ===");  repo.findByNameContaining("Phone").forEach(System.***out***::println);  System.***out***.println("=== Starting with 'S' ===");  repo.findByNameStartingWith("S").forEach(System.***out***::println);  System.***out***.println("=== Price > 500 ===");  repo.findByPriceGreaterThan(500.0).forEach(System.***out***::println);  System.***out***.println("=== Price < 300 ===");  repo.findByPriceLessThan(300.0).forEach(System.***out***::println);  System.***out***.println("=== Created between May and July ===");  repo.findByCreatedDateBetween(  LocalDate.*of*(2023, 5, 1),  LocalDate.*of*(2023, 7, 31)  ).forEach(System.***out***::println);  System.***out***.println("=== Top 3 Expensive ===");  repo.findTop3ByOrderByPriceDesc().forEach(System.***out***::println);  System.***out***.println("=== Sorted by Name Ascending ===");  repo.findAllByOrderByNameAsc().forEach(System.***out***::println);  };  }  }  ProductRepository.java  **package** com.example.demo.repository;  **import** com.example.demo.entity.Product;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** java.time.LocalDate;  **import** java.util.List;  **public** **interface** ProductRepository **extends** JpaRepository<Product, Long> {  // Search by containing text  List<Product> findByNameContaining(String keyword);  // Filter by starting text  List<Product> findByNameStartingWith(String prefix);  // Filter by price >, <  List<Product> findByPriceGreaterThan(Double price);  List<Product> findByPriceLessThan(Double price);  // Between dates  List<Product> findByCreatedDateBetween(LocalDate start, LocalDate end);  // Top N  List<Product> findTop3ByOrderByPriceDesc(); // Top 3 expensive products  // Sorting (in name order)  List<Product> findAllByOrderByNameAsc();  }  Application.properties  spring.datasource.url=jdbc:h2:mem:testdb  spring.datasource.driver-class-name=org.h2.Driver  spring.datasource.username=sa  spring.datasource.password=  spring.jpa.hibernate.ddl-auto=create  spring.jpa.show-sql=true  spring.h2.console.enabled=true  OUTPUTSCREENSHOT        DEMONSTRATE IMPLEMENTATION OF O/R MAPPING  OrmDemoApplication.java  **package** com.cognizant.ormdemo;  **import** jakarta.persistence.EntityManager;  **import** jakarta.persistence.PersistenceContext;  **import** org.springframework.boot.CommandLineRunner;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.context.annotation.Bean;  **import** com.cognizant.ormdemo.entity.Course;  **import** com.cognizant.ormdemo.entity.Department;  **import** com.cognizant.ormdemo.entity.Student;  **import** java.util.Set;  @SpringBootApplication  **public** **class** OrmDemoApplication {  **public** **static** **void** main(String[] args) {  System.***out***.println("🚀 Main method started");  SpringApplication.*run*(OrmDemoApplication.**class**, args);  }  @PersistenceContext  **private** EntityManager entityManager;  @Bean  **public** CommandLineRunner demo() {  **return** args -> {  System.***out***.println("✅ Demo method running...");  Department dept = **new** Department();  dept.setName("Computer Science");  entityManager.persist(dept);  Course c1 = **new** Course();  c1.setCourseName("Java");  Course c2 = **new** Course();  c2.setCourseName("Spring Boot");  entityManager.persist(c1);  entityManager.persist(c2);  Student s1 = **new** Student();  s1.setName("Abarna");  s1.setDepartment(dept);  s1.setCourses(Set.*of*(c1, c2));  entityManager.persist(s1);  System.***out***.println("✅ Inserted Student: " + s1.getName());  System.***out***.println("✅ Department: " + dept.getName());  System.***out***.println("✅ Courses: " + c1.getCourseName() + ", " + c2.getCourseName());  };  }  }  Couse.java  **package** com.cognizant.ormdemo.entity;  **import** jakarta.persistence.\*;  **import** java.util.Set;  @Entity  **public** **class** Course {  @Id  @GeneratedValue  **private** Long id;  **private** String courseName;  @ManyToMany(mappedBy = "courses")  **private** Set<Student> students;  // ✅ ADD THIS  **public** **void** setCourseName(String courseName) {  **this**.courseName = courseName;  }  **public** String getCourseName() {  **return** courseName;  }  }  Department.java  **package** com.cognizant.ormdemo.entity;  **import** jakarta.persistence.\*;  **import** java.util.List;  @Entity  **public** **class** Department {  @Id  @GeneratedValue  **private** Long id;  **private** String name;  @OneToMany(mappedBy = "department", fetch = FetchType.***LAZY***)  **private** List<Student> students;  // ✅ ADD THIS  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getName() {  **return** name;  }  }  Student.java  **package** com.cognizant.ormdemo.entity;  **import** jakarta.persistence.\*;  **import** java.util.Set;  @Entity  **public** **class** Student {  @Id  @GeneratedValue  **private** Long id;  **private** String name;  @ManyToOne(fetch = FetchType.***EAGER***)  @JoinColumn(name = "department\_id")  **private** Department department;  @ManyToMany  @JoinTable(  name = "student\_course",  joinColumns = @JoinColumn(name = "student\_id"),  inverseJoinColumns = @JoinColumn(name = "course\_id")  )  **private** Set<Course> courses;  // ✅ Setters  **public** **void** setName(String name) { **this**.name = name; }  **public** **void** setDepartment(Department department) { **this**.department = department; }  **public** **void** setCourses(Set<Course> courses) { **this**.courses = courses; }  // ✅ Getter for console output  **public** String getName() { **return** name; }  }  Application.properties  # Use H2 in-memory database  spring.datasource.url=jdbc:h2:mem:testdb  spring.datasource.driver-class-name=org.h2.Driver  spring.datasource.username=sa  spring.datasource.password=  # Recreate DB fresh every run  spring.jpa.hibernate.ddl-auto=create-drop  # Show generated SQL  spring.jpa.show-sql=true  # Enable H2 console at /h2-console  spring.h2.console.enabled=true  spring.h2.console.path=/h2-console  OUTPUTSCREENHOT    3.SPRING DATA JPA HANDSON  **Hibernate Query Language (HQL)**, **JPQL**, **Native Query**, @Query annotation, and **Criteria Query**, along with examples and comparisons based on the references you provided.  HQL example  // Retrieve all employees  String hql = "FROM Employee e WHERE e.salary > :minSalary";  Query query = session.createQuery(hql);  query.setParameter("minSalary", 50000);  List<Employee> employees = query.list();  JPQL Example (with @Query annotation)  public interface EmployeeRepository extends JpaRepository<Employee, Long> {    @Query("SELECT e FROM Employee e WHERE e.salary > :minSalary")  List<Employee> findHighEarners(@Param("minSalary") Double salary);  }  HQL with fetch keyword (Join Fetch)  String hql = "SELECT e FROM Employee e JOIN FETCH e.department WHERE e.id = :id";  Query query = session.createQuery(hql);  query.setParameter("id", 1L);  Employee emp = (Employee) query.uniqueResult();  Aggregate Functions in HQL  String hql = "SELECT AVG(e.salary) FROM Employee e";  Double avgSalary = (Double) session.createQuery(hql).uniqueResult();  Native SQL Query (Using @Query with nativeQuery=true)  @Query(value = "SELECT \* FROM employee WHERE salary > ?1", nativeQuery = true)  List<Employee> findEmployeesWithHighSalary(Double salary); **Need and Benefits of Criteria Query****✅ Why use Criteria API?**  1. **Dynamic Query Construction**: Useful when query conditions are built programmatically. 2. **Type-safe**: Errors can be caught at compile-time rather than runtime. 3. **Avoids hard-coded strings**: Enhances refactorability and maintainability. 4. **Support for complex queries**: Makes combining multiple conditions easier.   Criteria Query Example  CriteriaBuilder cb = entityManager.getCriteriaBuilder();  CriteriaQuery<Employee> cq = cb.createQuery(Employee.class);    Root<Employee> employee = cq.from(Employee.class);  cq.select(employee).where(cb.greaterThan(employee.get("salary"), 50000));    TypedQuery<Employee> query = entityManager.createQuery(cq);  List<Employee> result = query.getResultList(); |

|  |  |
| --- | --- |
|  |  |