Week 3 - Spring Core\_Maven

**Exercise 1: Configuring a Basic Spring Application**

Scenario:

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

Steps:

1. Set Up a Spring Project:

o Create a Maven project named LibraryManagement.

o Add Spring Core dependencies in the pom.xml file.

2. Configure the Application Context:

o Create an XML configuration file named applicationContext.xml in the src/main/resources directory.

o Define beans for BookService and BookRepository in the XML file.

3. Define Service and Repository Classes:

o Create a package com.library.service and add a class BookService.

o Create a package com.library.repository and add a class BookRepository.

4. Run the Application:

o Create a main class to load the Spring context and test the configuration.

**Exercise 2: Implementing Dependency Injection**

Scenario:

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

Steps:

1. Modify the XML Configuration:

o Update applicationContext.xml to wire BookRepository into BookService.

2. Update the BookService Class:

o Ensure that BookService class has a setter method for BookRepository.

3. Test the Configuration:

o Run the LibraryManagementApplication main class to verify the dependency injection.

**Code:**

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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>library\_management</artifactId>

<version>1.0-SNAPSHOT</version>

<name>library\_management</name>

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- AspectJ -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

</dependencies>

<build>

<pluginManagement>

<plugins>

<plugin>

<artifactId>maven-clean-plugin</artifactId>

<version>3.1.0</version>

</plugin>

<plugin>

<artifactId>maven-resources-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

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

<version>3.8.0</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.1</version>

</plugin>

<plugin>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-install-plugin</artifactId>

<version>2.5.2</version>

</plugin>

<plugin>

<artifactId>maven-deploy-plugin</artifactId>

<version>2.8.2</version>

</plugin>

<plugin>

<artifactId>maven-site-plugin</artifactId>

<version>3.7.1</version>

</plugin>

<plugin>

<artifactId>maven-project-info-reports-plugin</artifactId>

<version>3.0.0</version>

</plugin>

</plugins>

</pluginManagement>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.example.LibraryApp</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

applicationContent.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

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

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

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

<aop:aspectj-autoproxy />

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

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

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

</bean>

</beans>

LibraryApp.java

package com.example;

import com.example.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {

public static void main(String[] args) {

// Load Spring context from applicationContext.xml

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

// Get BookService bean

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

// Use the service

bookService.addBook("The Great Gatsby");

// Close context if needed (for older Spring versions)

((ClassPathXmlApplicationContext) context).close();

}

}

LoggingAspect.java

package com.example.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.example.service.\*.\*(..))")

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object proceed = joinPoint.proceed(); // execute method

long executionTime = System.currentTimeMillis() - start;

System.out.println(joinPoint.getSignature() + " executed in " + executionTime + "ms");

return proceed;

}

}

BookRepository.java

package com.example.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.out.println("BookRepository: Saving book - " + bookName);

}

}

BookService.java

package com.example.service;

import com.example.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

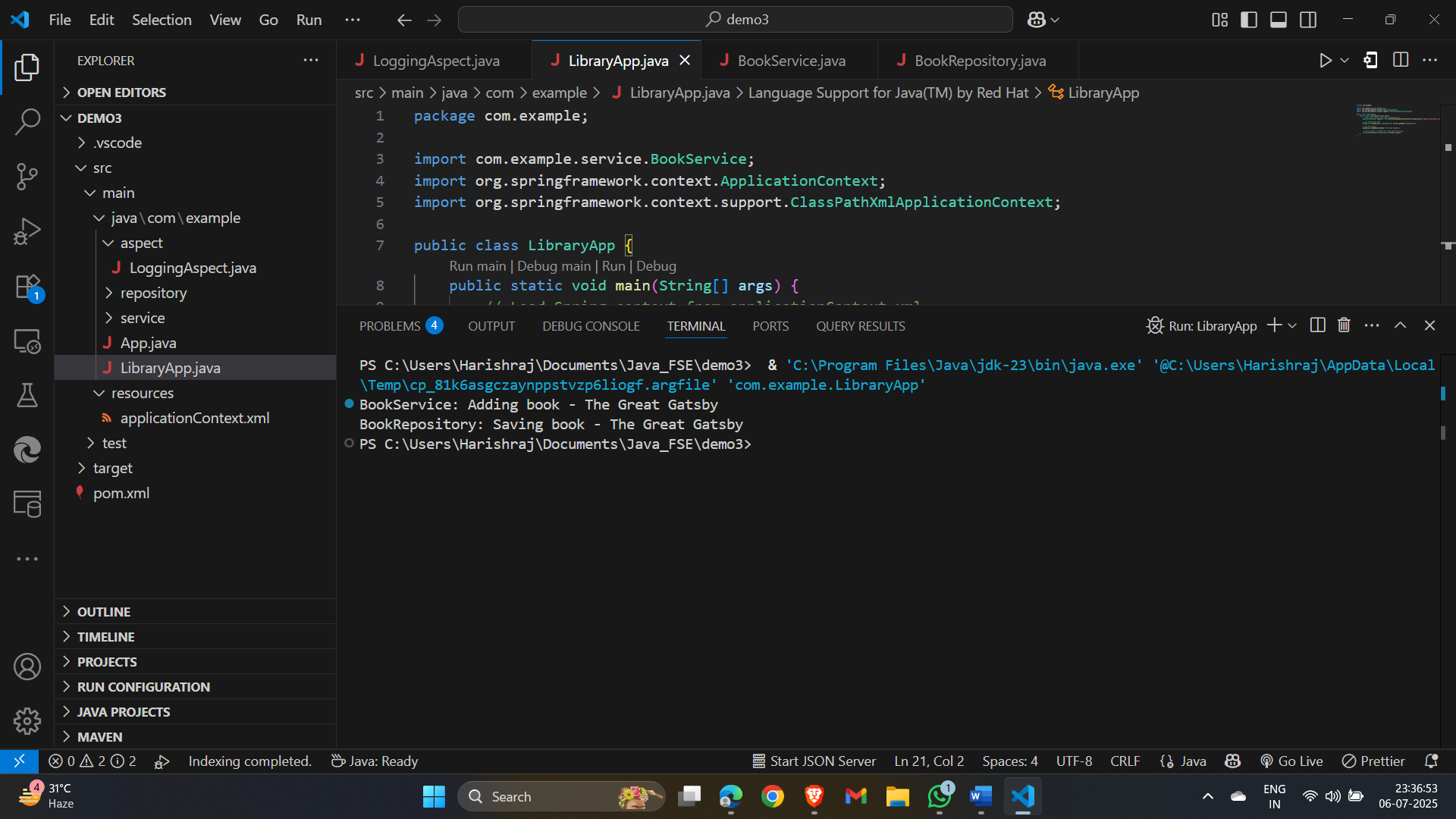
System.out.println("BookService: Adding book - " + bookName);

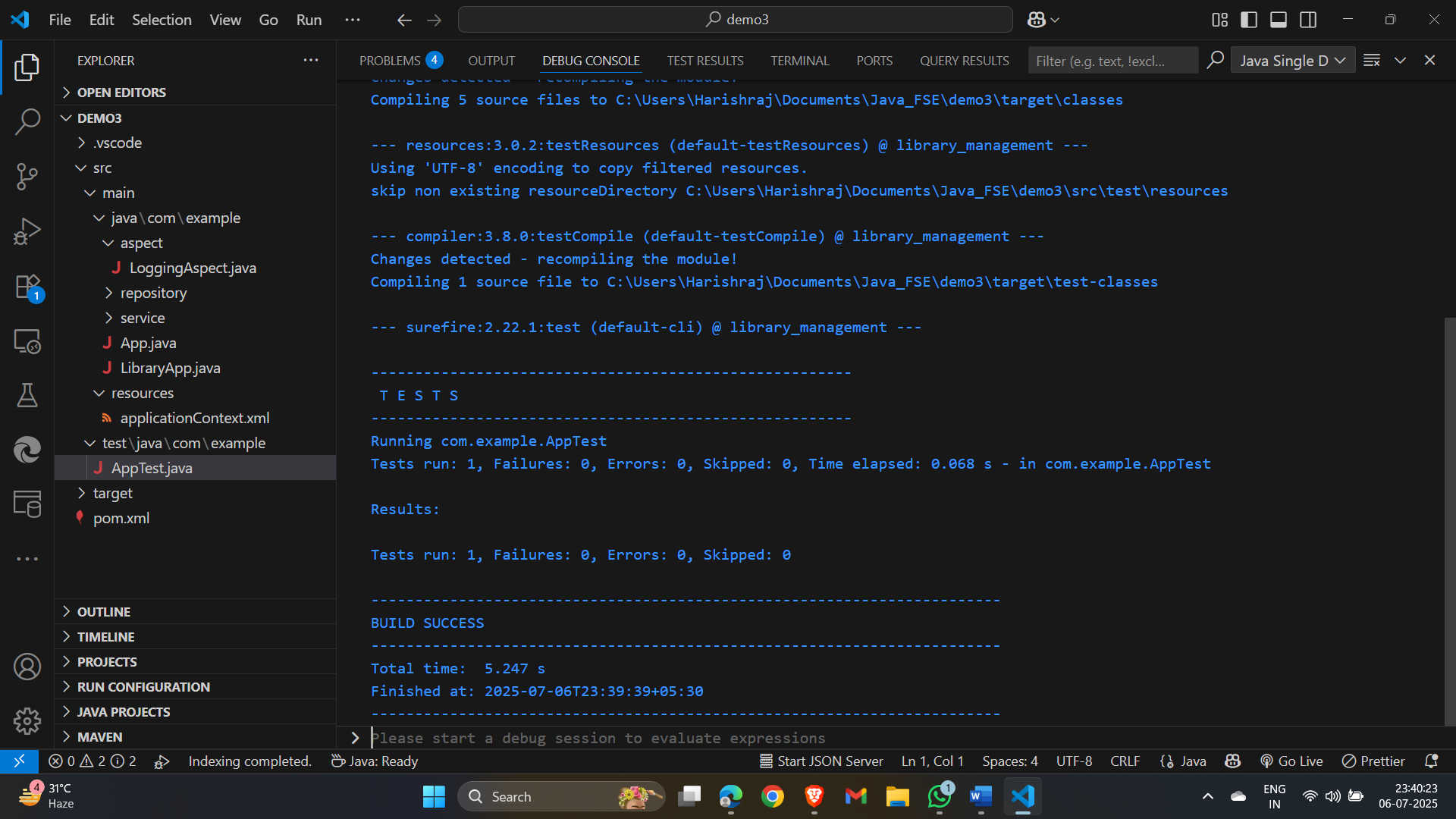
bookRepository.saveBook(bookName);

}

}

Output:





**Exercise 4: Creating and Configuring a Maven Project**

Scenario:

You need to set up a new Maven project for the library management application and add Spring dependencies.

Steps:

1. Create a New Maven Project:

o Create a new Maven project named LibraryManagement.

2. Add Spring Dependencies in pom.xml:

o Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.

3. Configure Maven Plugins:

o Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**Code:**

AppConfig.java  
package com.example;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

public BookRepository bookRepository() {

return new BookRepository();

}

@Bean

public BookService bookService() {

BookService service = new BookService();

service.setBookRepository(bookRepository());

return service;

}

}

BookRepository.java

package com.example;

public class BookRepository {

public void save(String book) {

System.out.println("BookRepository: Saving book - " + book);

}

}

BookService.java

package com.example;

public class BookService {

private BookRepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String book) {

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

bookRepository.save(book);

}

}

LibraryManagementApplication.java

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class LibraryManagementApplication {

public static void main(String[] args) {

// Load Spring context from AppConfig

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

// Get the BookService bean

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

// Call a method to test

bookService.addBook("The Spring Journey");

}

}