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

MainApp.java

package com.library.main;  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import com.library.service.BookService;  
  
public class MainApp {  
 public static void main(String[] args) {  
 // Load Spring context from XML  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // Get bean and use it  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.addBook("The Alchemist");  
 }  
}

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  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Define BookRepository bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- Define BookService bean with dependency injection -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>\

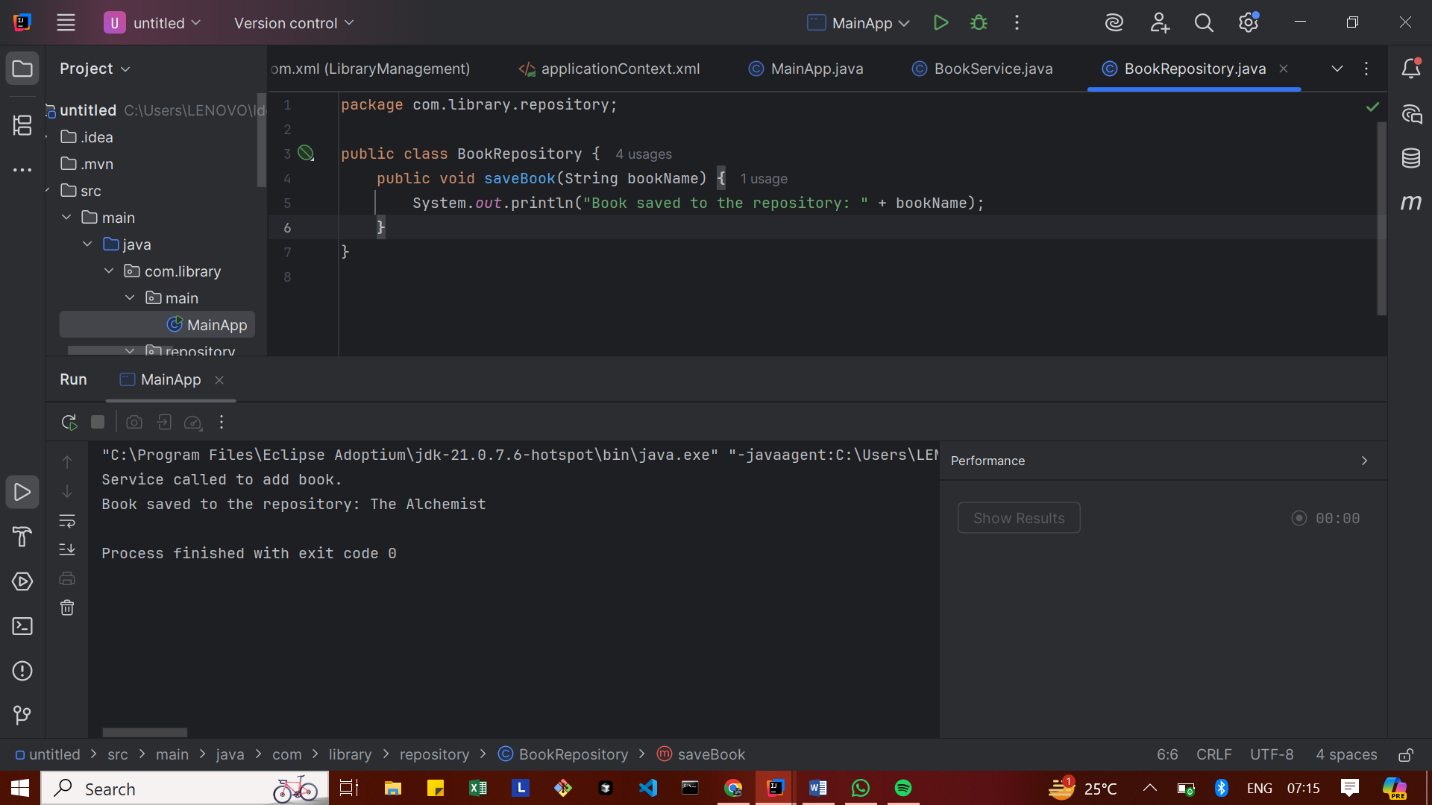
BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter for Spring to inject the dependency  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Service called to add book.");  
 bookRepository.saveBook(bookName);  
 }  
}

BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String bookName) {  
 System.*out*.println("Book saved to the repository: " + bookName);  
 }  
}

**OUTPUT**

****

**Exercise 2: Implementing Dependency Injection**

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  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Define BookRepository bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- Define BookService bean and inject dependency -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

BookService.java

package com.library.service;  
  
import com.library.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("Service called to add book.");  
 bookRepository.save(bookName);  
 }  
}

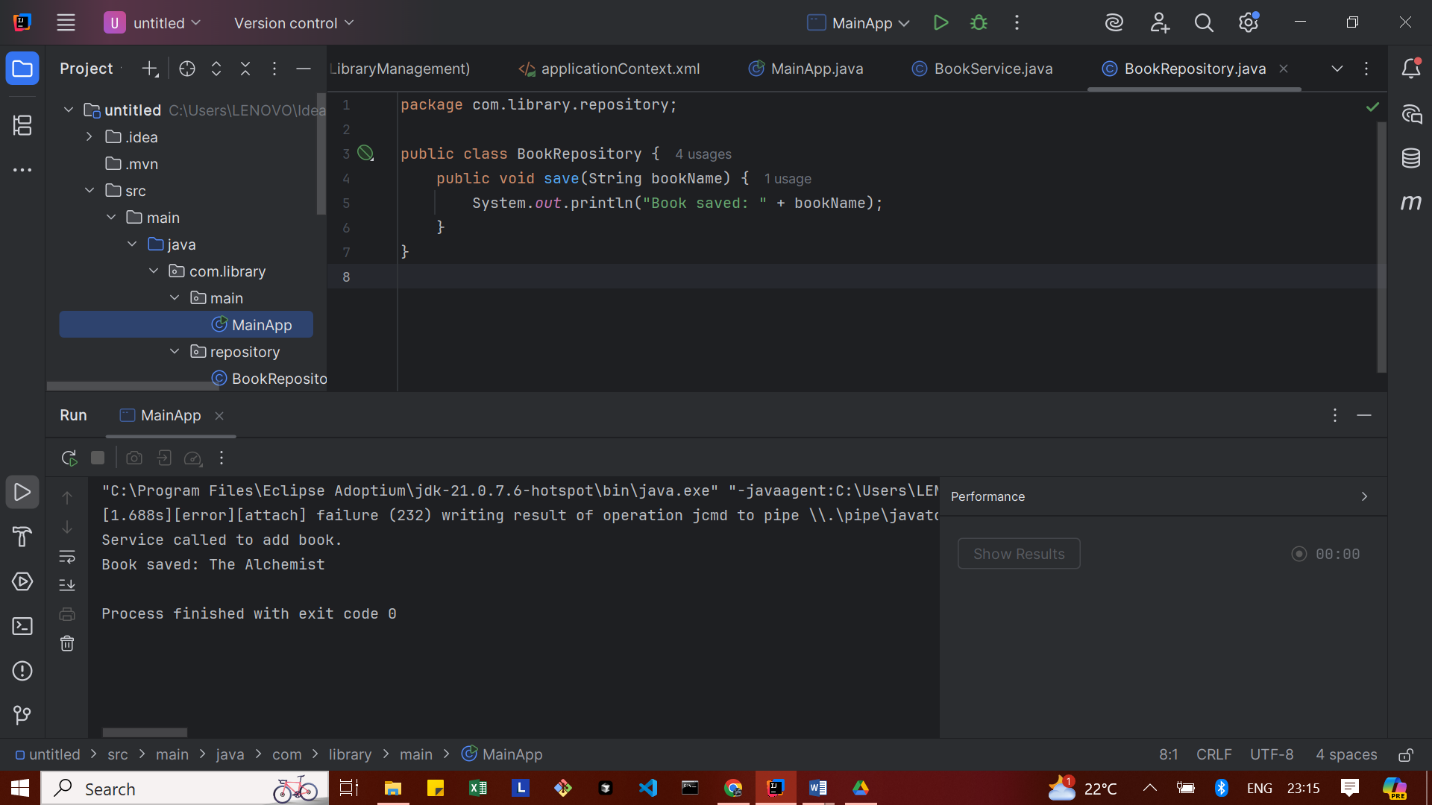
MainApp.java

package com.library.main;  
  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import com.library.service.BookService;  
  
public class MainApp {  
 public static void main(String[] args) {  
 // Load Spring context from XML  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // Get the BookService bean  
 BookService service = context.getBean("bookService", BookService.class);  
  
 // Test method  
 service.addBook("The Alchemist");  
 }  
}

BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void save(String bookName) {  
 System.*out*.println("Book saved: " + bookName);  
 }  
}

**OUTPUT**

****

**Exercise 3: Implementing Logging with Spring AOP**

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.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <name>LibraryManagement</name>  
 <url>http://www.example.com</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 </properties>  
  
 <repositories>  
 <repository>  
 <id>central</id>  
 <url>https://repo.maven.apache.org/maven2</url>  
 </repository>  
 </repositories>  
  
 <dependencies>  
 <!-- ✅ Spring Context Dependency -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.31</version> <!-- Updated to latest compatible version -->  
 </dependency>  
  
 <!-- ✅ JUnit -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
  
 <!-- AspectJ Weaver -->  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.9.19</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>  
 </build>  
  
</project>

LoggingAspect

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(); // execute the method  
  
 long end = System.*currentTimeMillis*();  
 System.*out*.println("Execution time of " + joinPoint.getSignature() + ": " + (end - start) + "ms");  
  
 return result;  
 }  
}

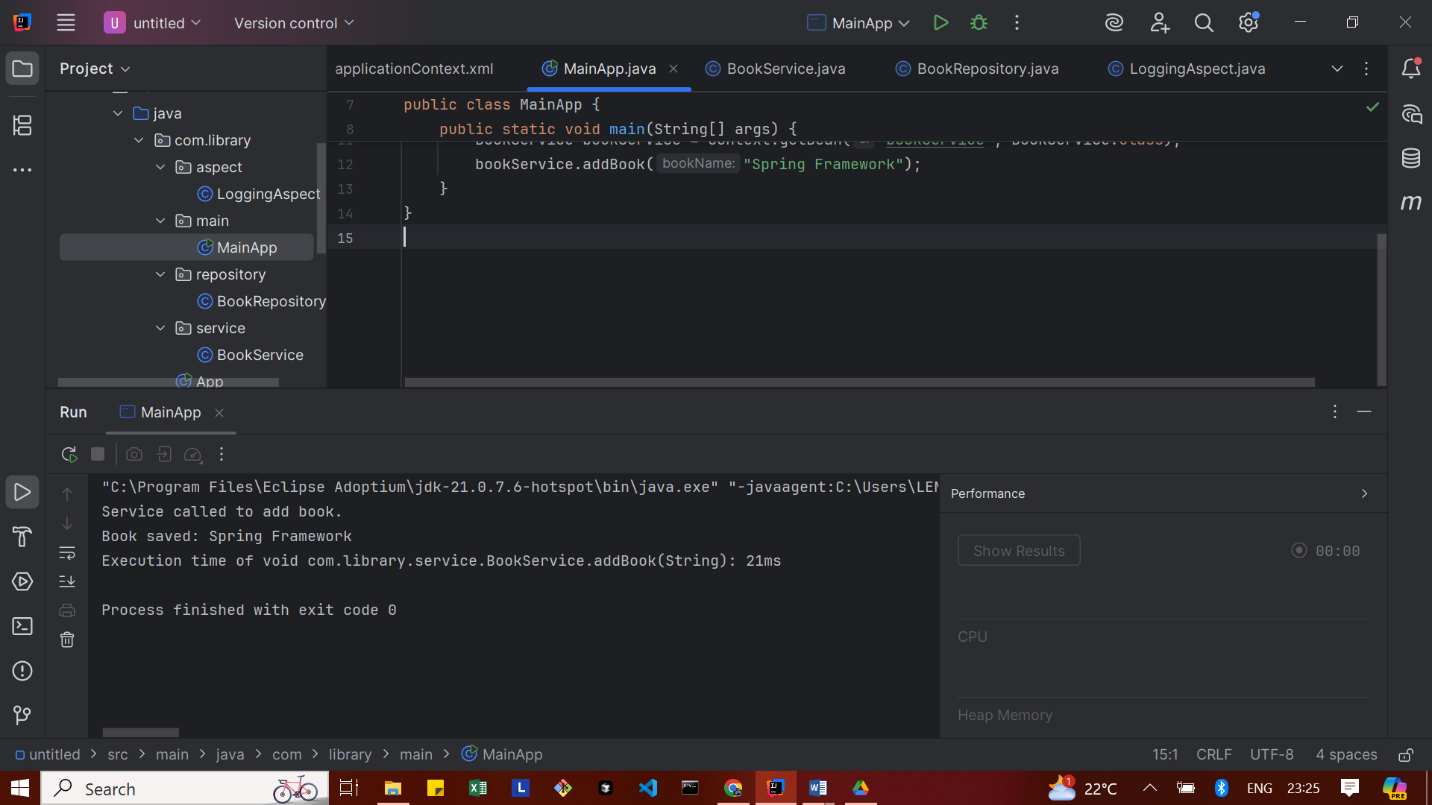
applicationContext.xml

<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">  
  
 <!-- Enable AOP support -->  
 <aop:aspectj-autoproxy />  
  
 <!-- Register Repository -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- Register Service and inject Repository -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
 <!-- Register Logging Aspect -->  
 <bean id="loggingAspect" class="com.library.aspect.LoggingAspect" />  
  
</beans>

MainApp.java

package com.library.main;  
  
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", BookService.class);  
 bookService.addBook("Spring Framework");  
 }  
}

**OUTPUT**

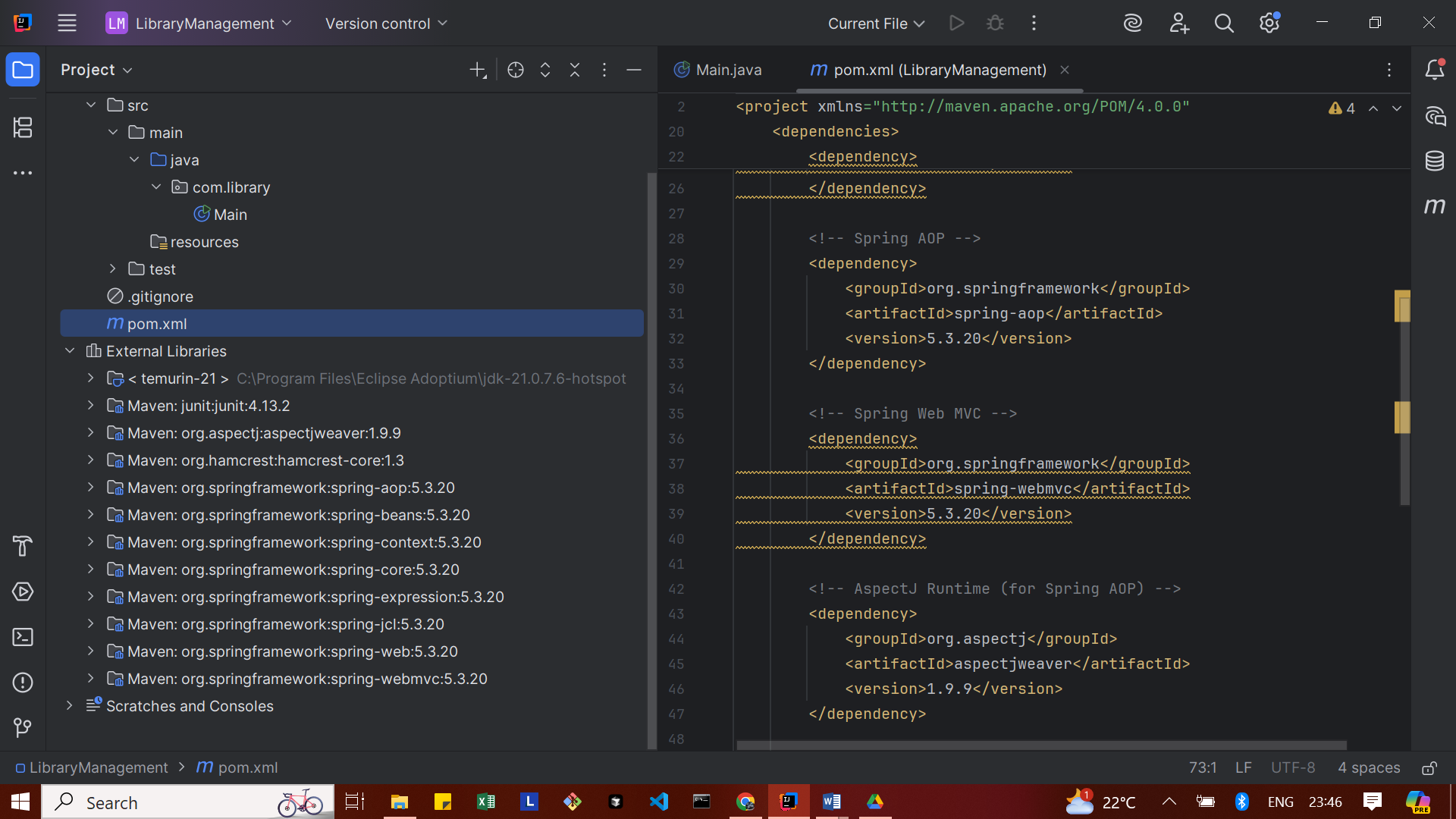
****

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

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.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <name>LibraryManagement</name>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
  
 <!-- Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
  
 <!-- Spring Web MVC -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
  
 <!-- AspectJ Runtime (for Spring AOP) -->  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.9.9</version>  
 </dependency>  
  
 <!-- JUnit for testing -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Maven Compiler Plugin -->  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**OUTPUT**

****

**Exercise 5: Configuring the Spring IoC Container**

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.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <name>LibraryManagement</name>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
  
 <!-- Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
  
 <!-- Spring Web MVC -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
  
 <!-- AspectJ Runtime (for Spring AOP) -->  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.9.9</version>  
 </dependency>  
  
 <!-- JUnit for testing -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Maven Compiler Plugin -->  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</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"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- BookRepository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- BookService Bean with DI -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

MainApp.java

package com.library.main;  
  
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", BookService.class);  
 bookService.addBook("Spring In Action");  
 }  
}

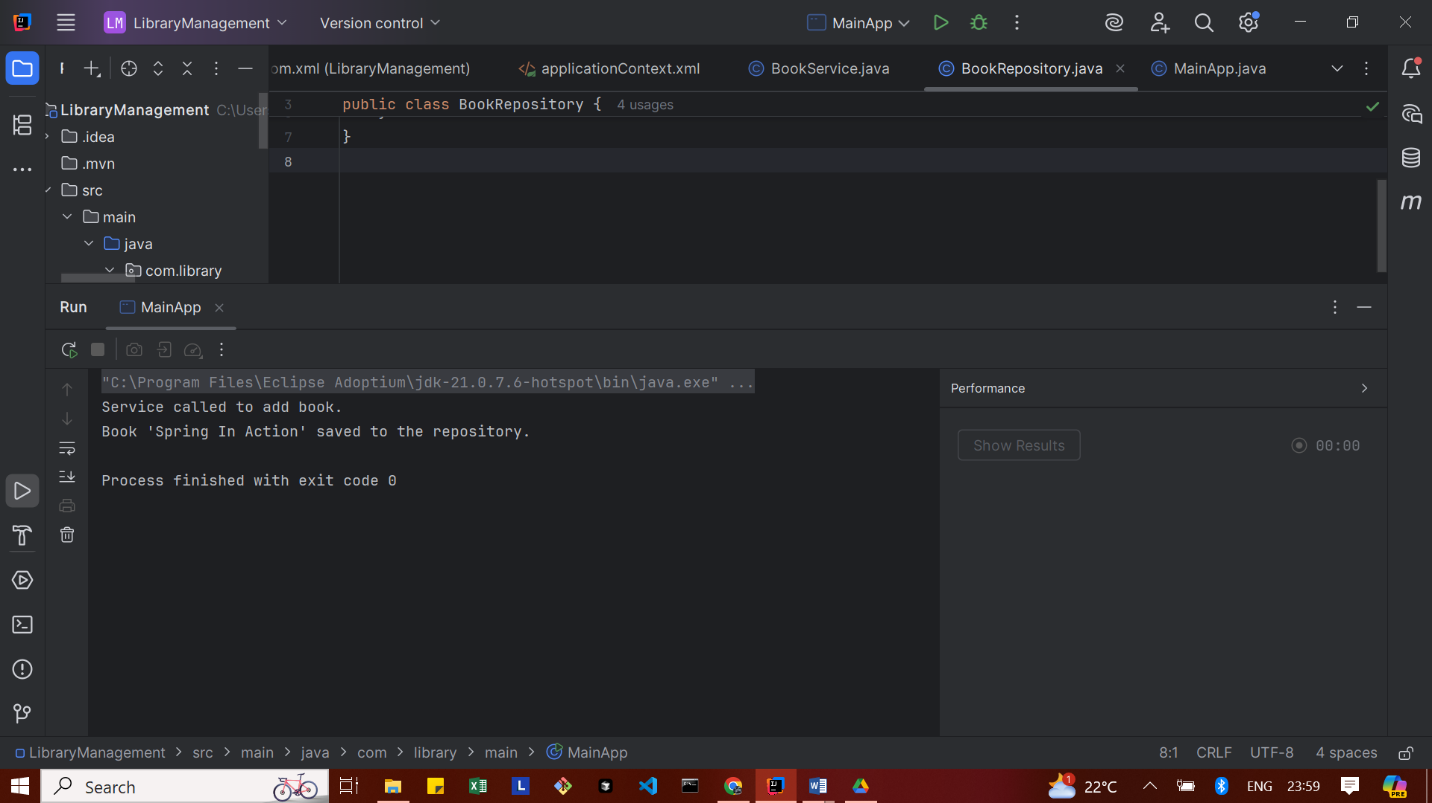
BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Service called to add book.");  
 bookRepository.save(bookName);  
 }  
}

BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void save(String bookName) {  
 System.*out*.println("Book '" + bookName + "' saved to the repository.");  
 }  
}

**OUTPUT**

****

**Exercise 6: Configuring Beans with Annotations**

applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context  
 http://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 bookName) {  
 System.*out*.println("Book '" + bookName + "' saved to the repository.");  
 }  
}

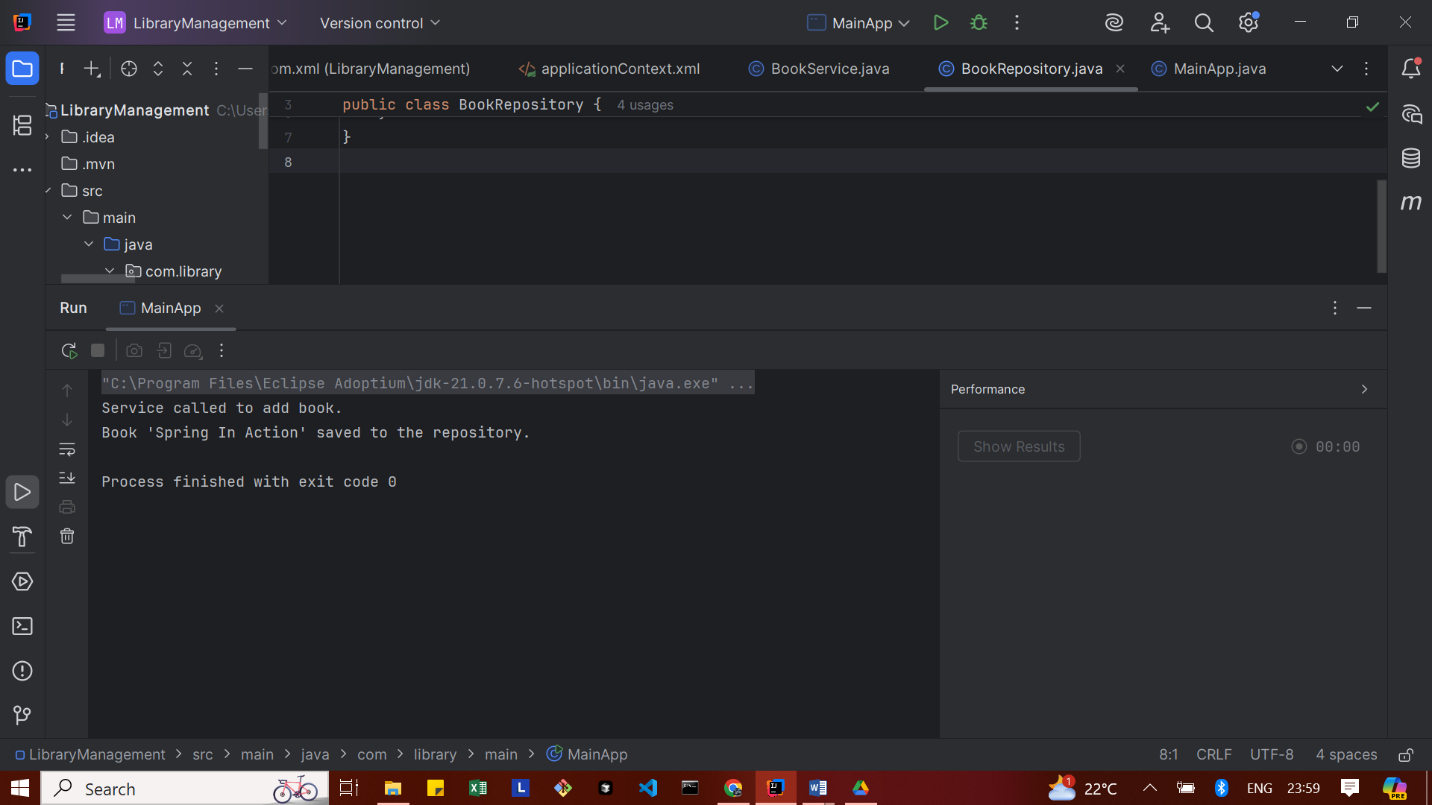
BookService.java

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 {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 public void addBook(String bookName) {  
 System.*out*.println("Service called to add book.");  
 bookRepository.save(bookName);  
 }  
}

MainApp.java

package com.library.main;  
  
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("Spring In Action");  
 }  
}

**OUTPUT**

****

**Exercise 7: Implementing Constructor and Setter Injection**

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  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- BookRepository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- BookService Bean with constructor + setter injection -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <!-- ✅ Constructor injection -->  
 <constructor-arg value="Welcome to Library Management!" />  
  
 <!-- ✅ Setter injection -->  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

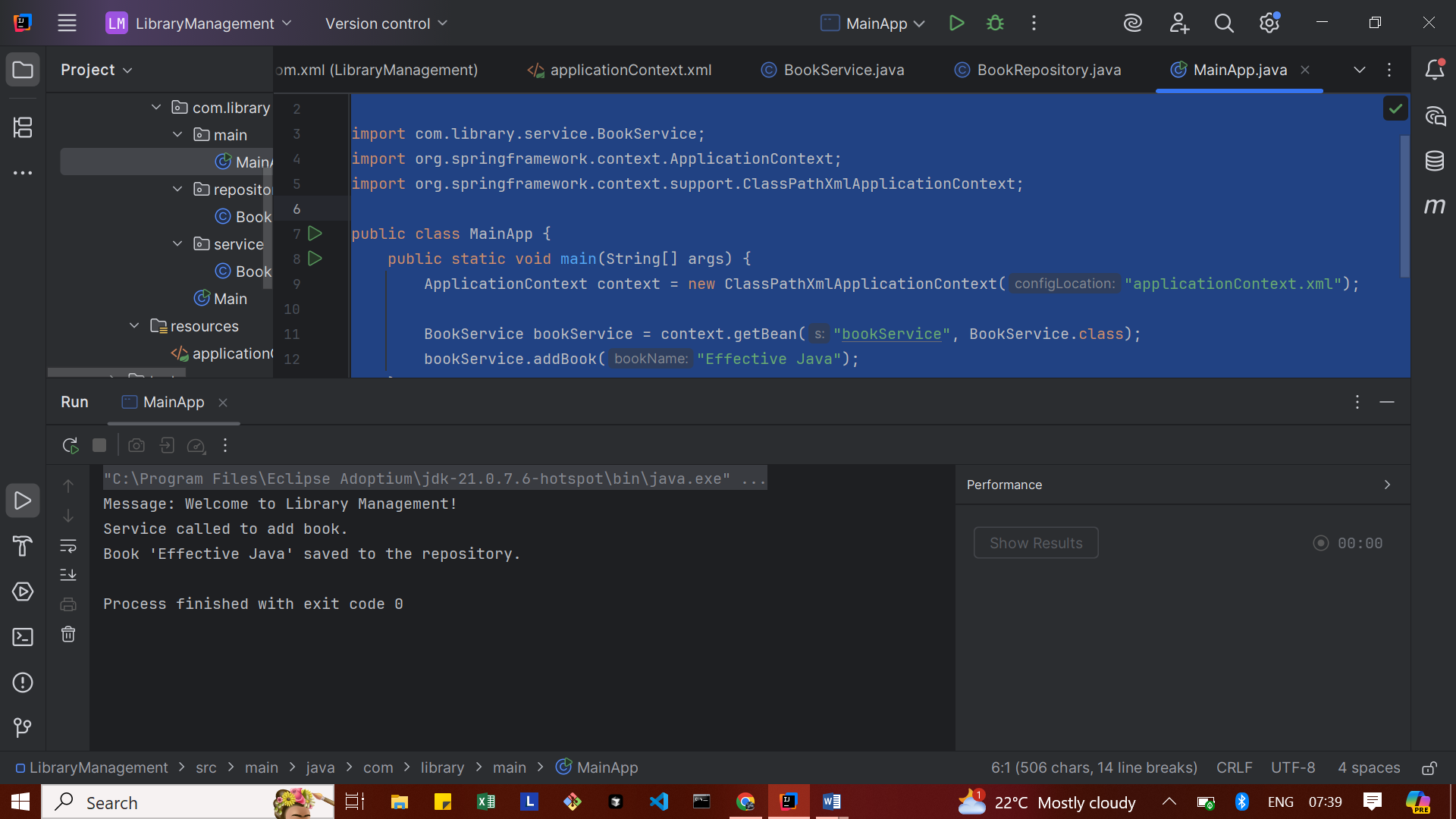
BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
 private String message;  
  
 // ✅ Constructor injection  
 public BookService(String message) {  
 this.message = message;  
 }  
  
 // ✅ Setter injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Message: " + message);  
 System.*out*.println("Service called to add book.");  
 bookRepository.save(bookName);  
 }  
}

MainApp.java

package com.library.main;  
  
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", BookService.class);  
 bookService.addBook("Effective Java");  
 }  
}

**OUTPUT**

****

**Exercise 8: Implementing Basic AOP with Spring**

applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd  
 http://www.springframework.org/schema/aop  
 http://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>

LoggingAspect.java

package com.library.aspect;  
  
import org.aspectj.lang.JoinPoint;  
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.\*(..))")  
 public void logBefore(JoinPoint joinPoint) {  
 System.*out*.println("[AOP] Before method: " + joinPoint.getSignature().getName());  
 }  
  
 @After("execution(\* com.library.service.BookService.\*(..))")  
 public void logAfter(JoinPoint joinPoint) {  
 System.*out*.println("[AOP] After method: " + joinPoint.getSignature().getName());  
 }  
}

MainApp.java

package com.library.main;  
  
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", BookService.class);  
 bookService.addBook("Effective Java");  
 }  
}

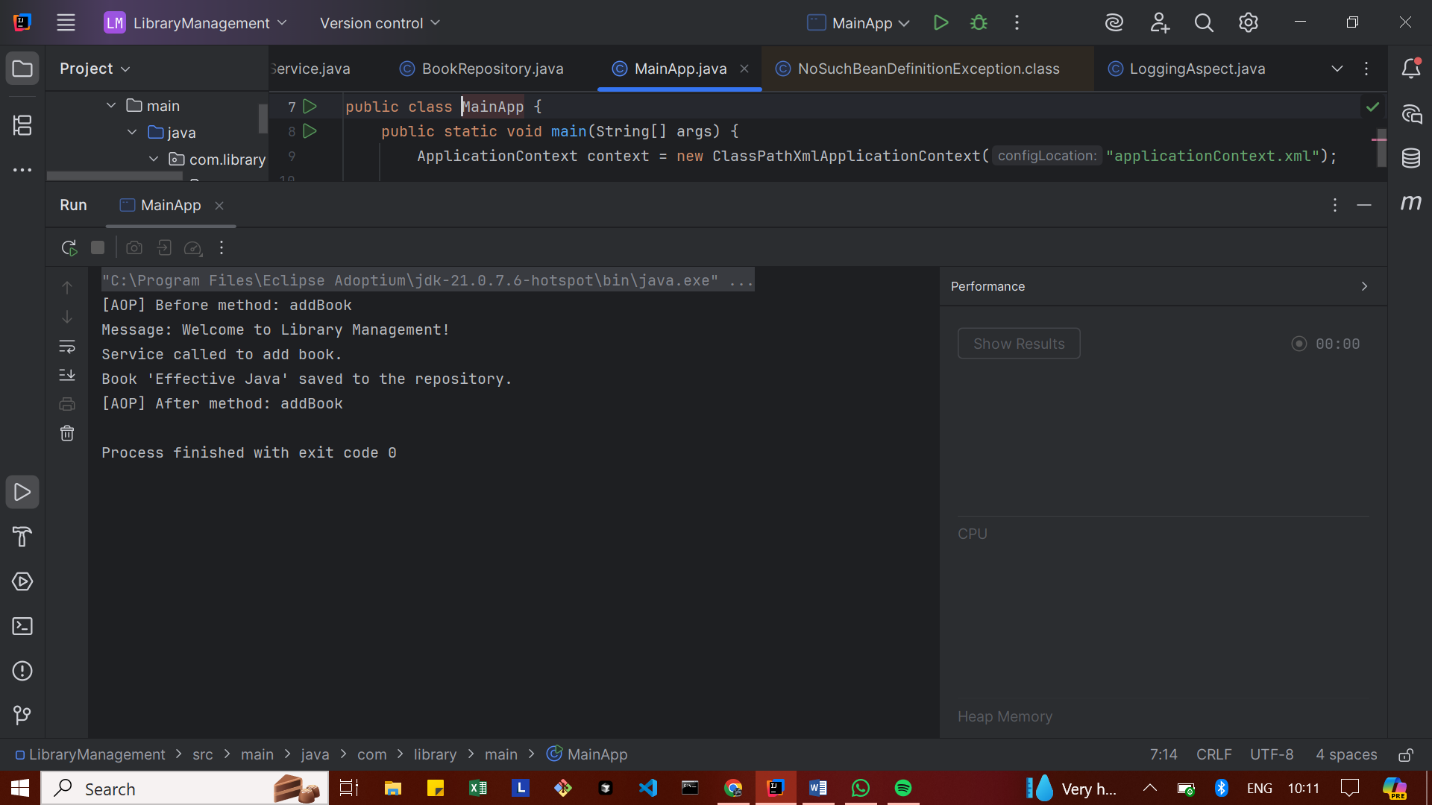
BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service // ✅ Marks this class as a Spring bean named "bookService"  
public class BookService {  
  
 private BookRepository bookRepository;  
 private String message;  
  
 // ✅ Constructor injection  
 @Autowired  
 public BookService() {  
 this.message = "Welcome to Library Management!";  
 }  
  
 // ✅ Setter injection (for BookRepository)  
 @Autowired  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Message: " + message);  
 System.*out*.println("Service called to add book.");  
 bookRepository.save(bookName);  
 }  
}

BookRepository.java

package com.library.repository;  
  
import org.springframework.stereotype.Repository;  
  
@Repository // ✅ Register as Spring bean  
public class BookRepository {  
 public void save(String bookName) {  
 System.*out*.println("Book '" + bookName + "' saved to the repository.");  
 }  
}

**OUTPUT**

****

**Exercise 9: Creating a Spring Boot Application**

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>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.5</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <groupId>com.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0.0</version>  
 <name>LibraryManagement</name>  
  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Boot Web Starter -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <!-- Spring Boot JPA Starter -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <!-- H2 In-Memory Database -->  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
  
 <!-- Spring Boot Test -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Spring Boot Maven Plugin -->  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

application.properties

spring.datasource.url=jdbc:h2:mem:librarydb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.h2.console.enabled=true  
spring.jpa.hibernate.ddl-auto=update

MainApp.java

package com.library;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class MainApp {  
 public static void main(String[] args) {  
 SpringApplication.*run*(MainApp.class, args);  
 }  
}

BookController.java

package com.library.controller;  
  
import com.library.model.Book;  
import com.library.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/books")  
public class BookController {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
  
 @PostMapping  
 public Book createBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
  
 @GetMapping("/{id}")  
 public Book getBookById(@PathVariable Long id) {  
 return bookRepository.findById(id).orElse(null);  
 }  
  
 @PutMapping("/{id}")  
 public Book updateBook(@PathVariable Long id, @RequestBody Book updatedBook) {  
 Book book = bookRepository.findById(id).orElse(null);  
 if (book != null) {  
 book.setTitle(updatedBook.getTitle());  
 book.setAuthor(updatedBook.getAuthor());  
 return bookRepository.save(book);  
 }  
 return null;  
 }  
  
 @DeleteMapping("/{id}")  
 public void deleteBook(@PathVariable Long id) {  
 bookRepository.deleteById(id);  
 }  
}

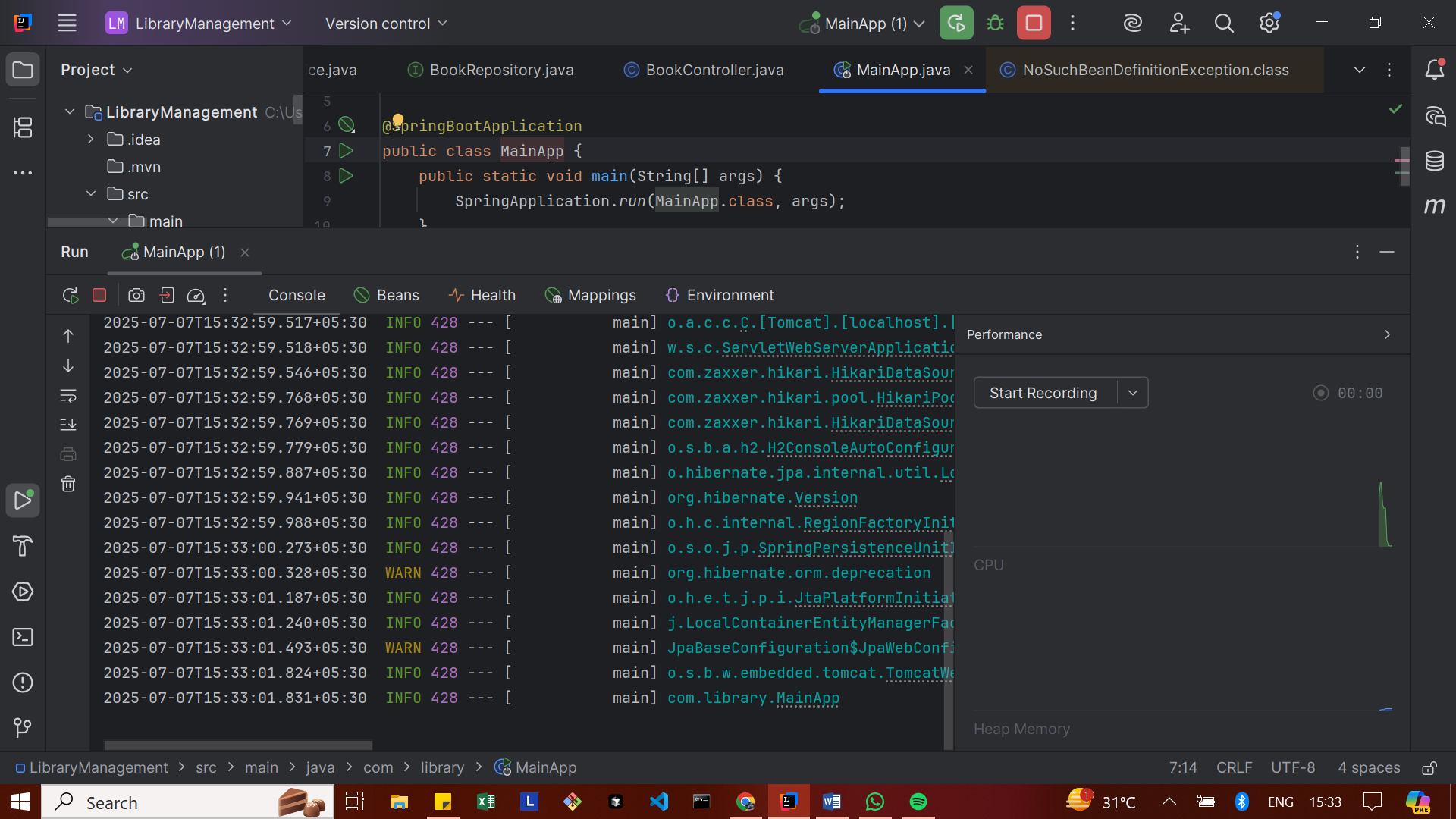
Book.java

package com.library.model;  
  
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; // ✅ NEW FIELD  
  
 public Book() {  
 }  
  
 public Book(String title, String author) {  
 this.title = title;  
 this.author = author;  
 }  
  
 public Book(Long id, String title, String author) {  
 this.id = id;  
 this.title = title;  
 this.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() { // ✅ Needed to fix error  
 return author;  
 }  
  
 public void setAuthor(String author) { // ✅ Needed to fix error  
 this.author = author;  
 }  
  
 @Override  
 public String toString() {  
 return "Book{" +  
 "id=" + id +  
 ", title='" + title + '\'' +  
 ", author='" + author + '\'' +  
 '}';  
 }  
}

BookRepository.java

package com.library.repository;  
  
import com.library.model.Book;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface BookRepository extends JpaRepository<Book, Long> {  
}

**OUTPUT**

****