**Spring Framework – Basic Application with Dependency Injection**

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

**Problem Statement**

You are tasked with developing the backend of a **Library Management System** using the **Spring Framework**. The goal is to set up a simple Spring project using Maven and configure it to manage service and repository layers.

**Setup Steps (in short)**

1. **Create Maven project**: LibraryManagement.
2. **Add Spring dependency** in pom.xml.
3. **Create BookService and BookRepository** classes.
4. **Configure beans in applicationContext.xml**.
5. **Create a main class to test the setup**.

**Code**

**🔹 pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0</version>

<properties>

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

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

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</project>

**🔹 applicationContext.xml (in src/main/resources)**

<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">

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

<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 saveBook(String title) {

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

}

}

**🔹 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 title) {

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

bookRepository.saveBook(title);

}

}

**🔹 LibraryManagementApplication.java**

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

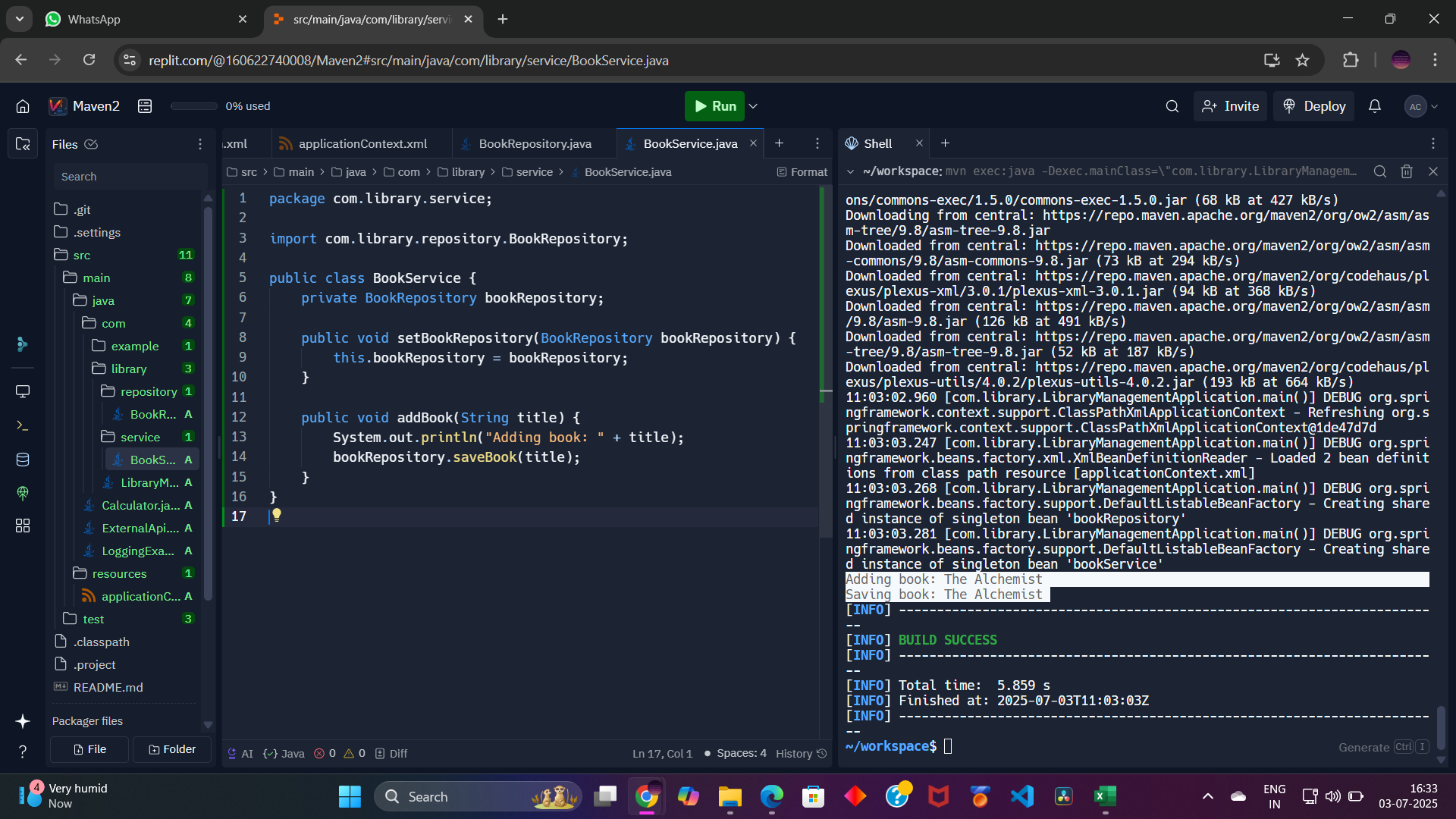
BookService bookService = context.getBean("bookService", BookService.class);

bookService.addBook("The Alchemist");

}

}

**Output:**

****

Adding book: The Alchemist

Saving book: The Alchemist

**Exercise 2: Implementing Dependency Injection**

**Problem Statement**

You must **wire BookRepository into BookService using Spring’s IoC container**. The injection should be done using **setter-based DI** defined in XML.

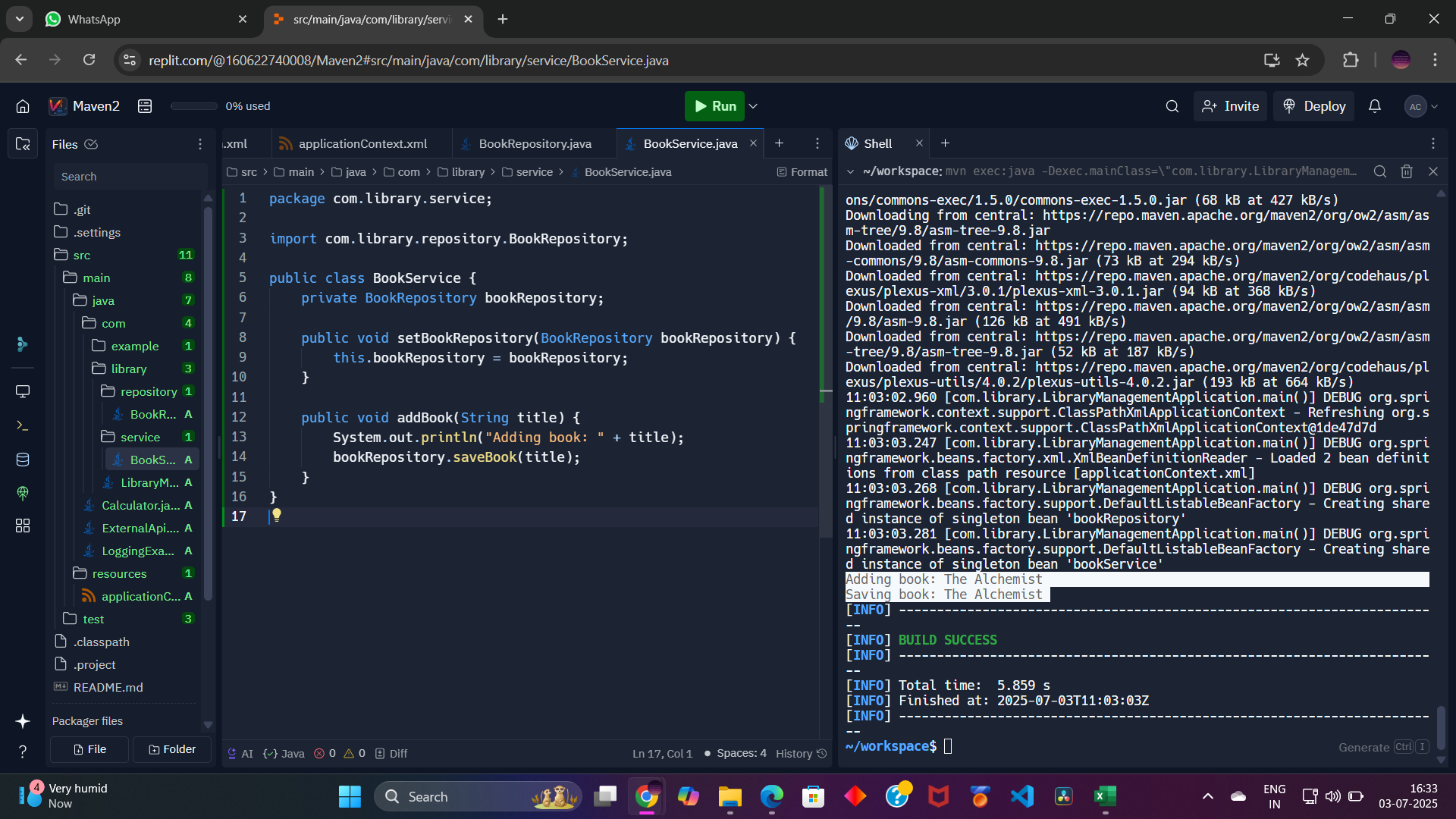
**Setup Steps (in short)**

1. **Update XML** to inject BookRepository into BookService.
2. **Ensure setter method exists** in BookService.
3. **Run the main class** to test the dependency injection.

**Code (Already covered in Exercise 1)**

* applicationContext.xml uses <property name="bookRepository" ref="bookRepository"/> for DI.
* BookService.java has the setBookRepository() method.
* LibraryManagementApplication.java runs the test.

**Output:**

****

Adding book: The Alchemist

Saving book: The Alchemist

✅ If we see the above output, Spring successfully injected the dependency using XML-based setter injection.

**Spring Framework – Exercise 4: Creating and Configuring a Maven Project**

**Problem Statement**

You need to set up a new **Maven project** for the Library Management application and configure it with the necessary **Spring dependencies** and **Maven plugins**, including support for Java 8.

**Setup Steps (in short)**

1. **Create a new Maven project** named LibraryManagement.
2. **Add Spring dependencies** for:
   * spring-context
   * spring-aop
   * spring-webmvc
3. **Configure the Maven Compiler Plugin** to use Java version 1.8.

**Code**

**🔹 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</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>5.3.34</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<plugin>

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

<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:**

Upon running:

mvn clean compile

