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:**
   - **Update applicationContext.xml to wire BookRepository into BookService.**

   Output:

```
http://www.springframework.org/schema/beans/spring-beans.xsd (xsi:schemaLocation)
<?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">

    <!-- Repository Bean -->
    <bean id="bookRepository" class="com.library.repository.BookRepository" />

    <!-- Service Bean -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>
```

2. **Update the BookService Class:**
   - **Ensure that BookService class has a setter method for BookRepository.**

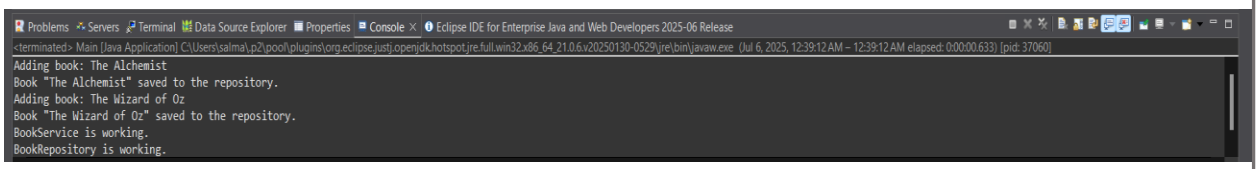
   Output:

LibraryManagement/pom.xml | applicationContext.xml | applicationContext.xml | 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("Adding book: " + bookName);
            bookRepository.saveBook(bookName);
        }

        public void performService() {
            System.out.println("BookService is working.");
            bookRepository.performRepositoryAction();
        }
}
```

3. **Test the Configuration:**
    - **Run the LibraryManagementApplication main class to verify the dependency injection.**

    Output:



- In this exercise, **Spring's Inversion of Control (IoC) container** manages object creation and dependency injection. The IoC container is configured **applicationContext.xml** where beans are defined and their relationships are established.
- The **BookService** class depends on **BookRepository** to perform backend data operations. Instead of manually instantiating the repository within the service class, Spring injects the dependency using **setter-based Dependency Injection**. This is achieved using the **<property>** tag in the **applicationContext.xml**