**Assignment Documentation: Spring Core + Maven (Library Management System)**

**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**.
2. **Update the BookService Class:**
   * Ensure that **BookService** class has a setter method for **BookRepository**.
3. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the dependency injection.

**Step 1: Modify the XML Configuration**

**applicationContext.xml:**

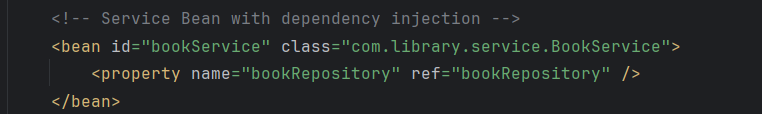
<bean id="bookRepository" class="com.library.repository.BookRepository" />

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

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

</bean>

**Screenshot: Updated XML Configuration**



**Step 2: Update BookService Class**

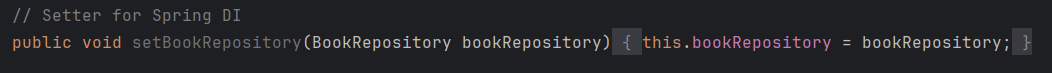
Ensure that the setter for BookRepository exists for setter injection.

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

**Screenshot: BookService with Setter**



**Step 3: Test the Configuration**

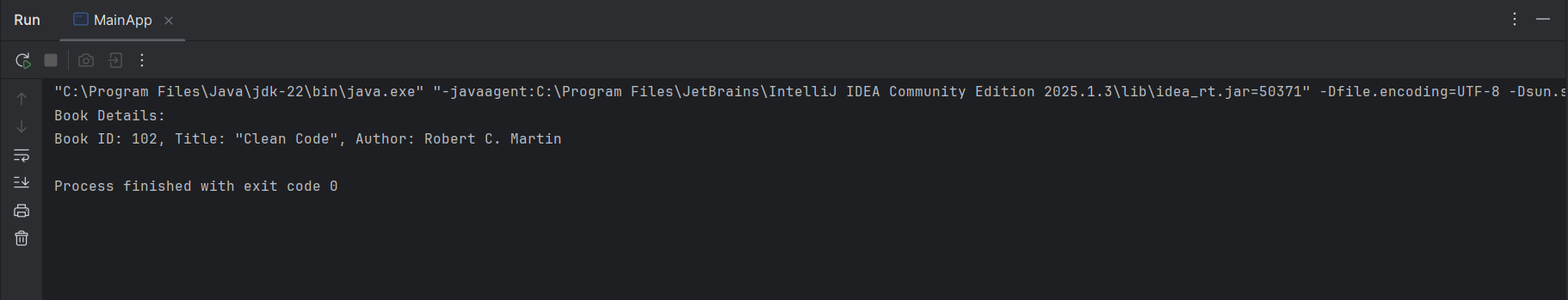
**MainApp.java:**

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

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

bookService.displayBookDetails(101);

**Screenshot: Output with Dependency Injection**



**Conclusion**

All requirements for Exercise 2 have been successfully implemented:

* Spring XML-based Configuration
* Proper use of IoC and Setter-based Dependency Injection
* Verified working output from main application class