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

**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

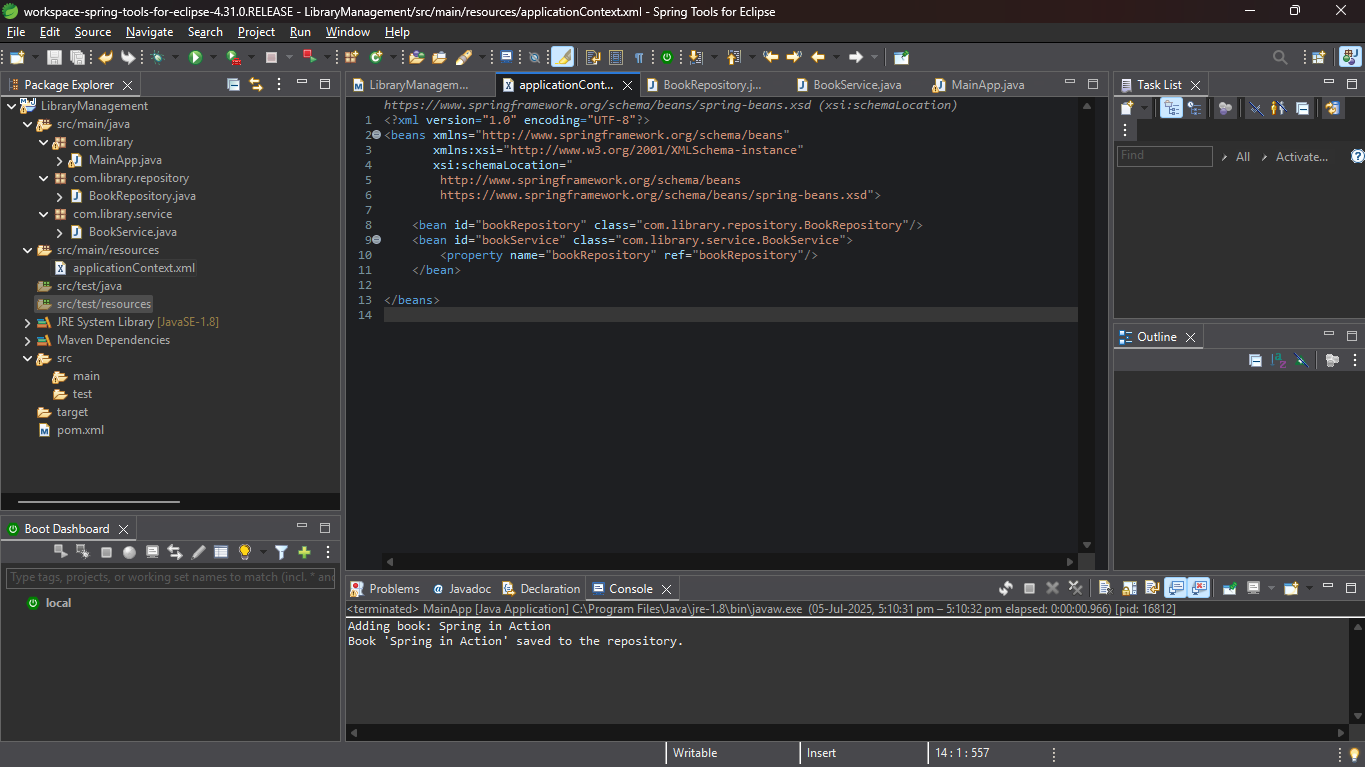
https://www.springframework.org/schema/beans/spring-beans.xsd">

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

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

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

</bean>

</beans>

2.

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.***out***.println("Book '" + bookName + "' saved to the repository.");

}

}

3.

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("Adding book: " + bookName);

bookRepository.saveBook(bookName);

}

}

4.Main

package com.library;

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) {

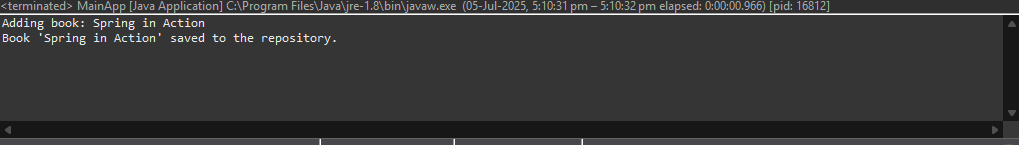
ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.addBook("Spring in Action");

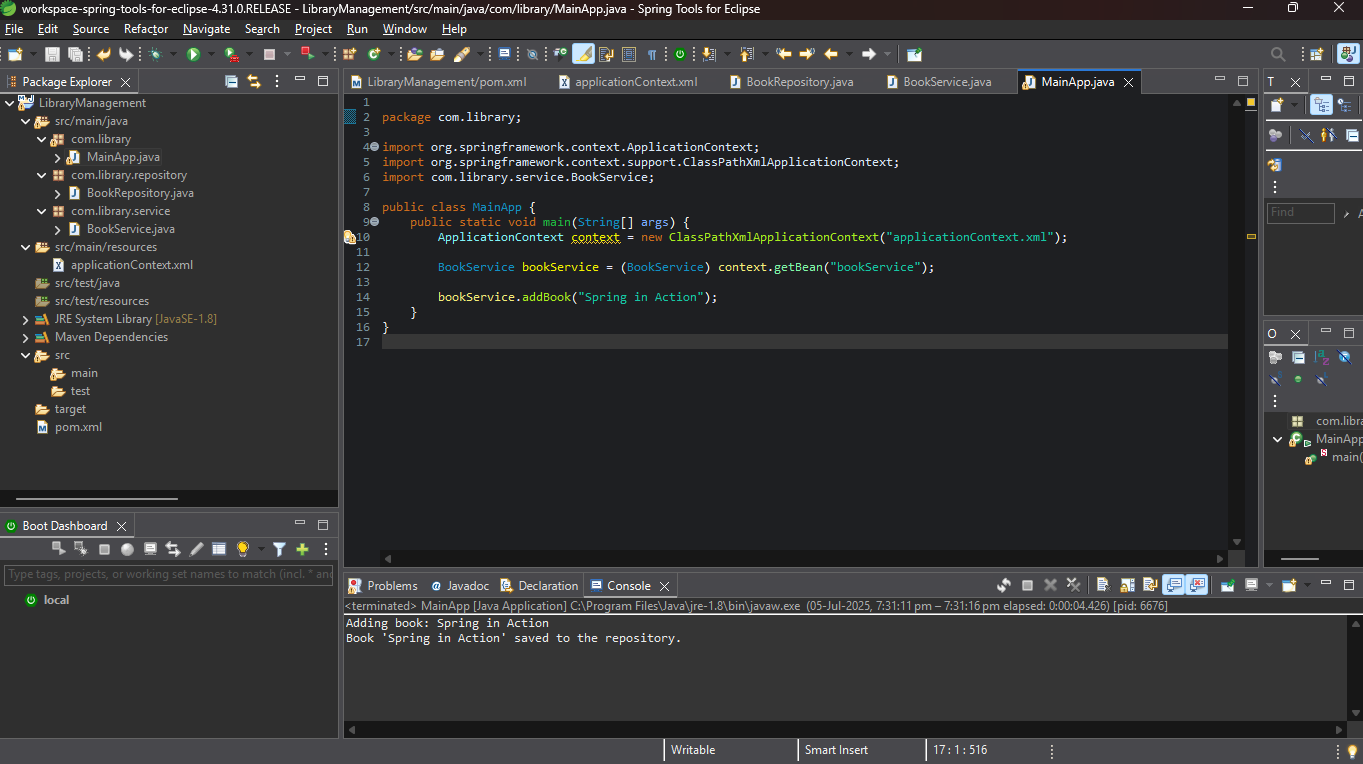
}

}



**Exercise 2: Implementing Dependency Injection**

* This exercise is the continuation of 1st one so in 1st exercise all are done.

****

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

1.AOP DEPENDENCY

<groupId>com.library</groupId>

<artifactId>LibraryManagement-AOP</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>5.3.30</version>

</dependency>

2.

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(); // Proceed with method execution

long duration = System.*currentTimeMillis*() - start;

System.***out***.println("Executed method: " + joinPoint.getSignature() +

" in " + duration + "ms");

return result;

}

}

3.

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

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

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

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

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

<aop:aspectj-autoproxy/>

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

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

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

</bean>

<bean id="loggingAspect" class="com.library.aspect.LoggingAspect"/>

</beans>

4.

package com.library;

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) {

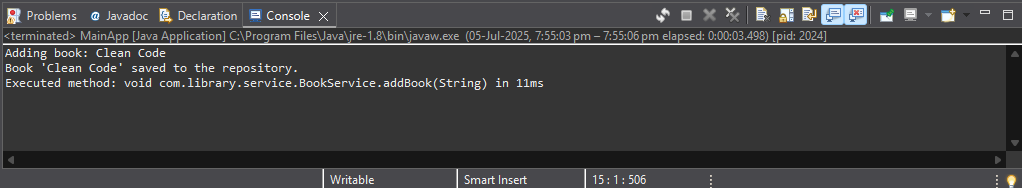
ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.addBook("Clean Code");

}

}



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

**Updated pom xml file**

<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>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Context (Core) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>5.3.30</version>

</dependency>

<!-- AspectJ runtime (needed for AOP) -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

<!-- Spring Web MVC (for future web integration) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

<!-- Servlet API (optional but required for Spring WebMVC if you plan to use it) -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- ✅ Compiler plugin for Java 1.8 -->

<plugin>

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

<version>3.11.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

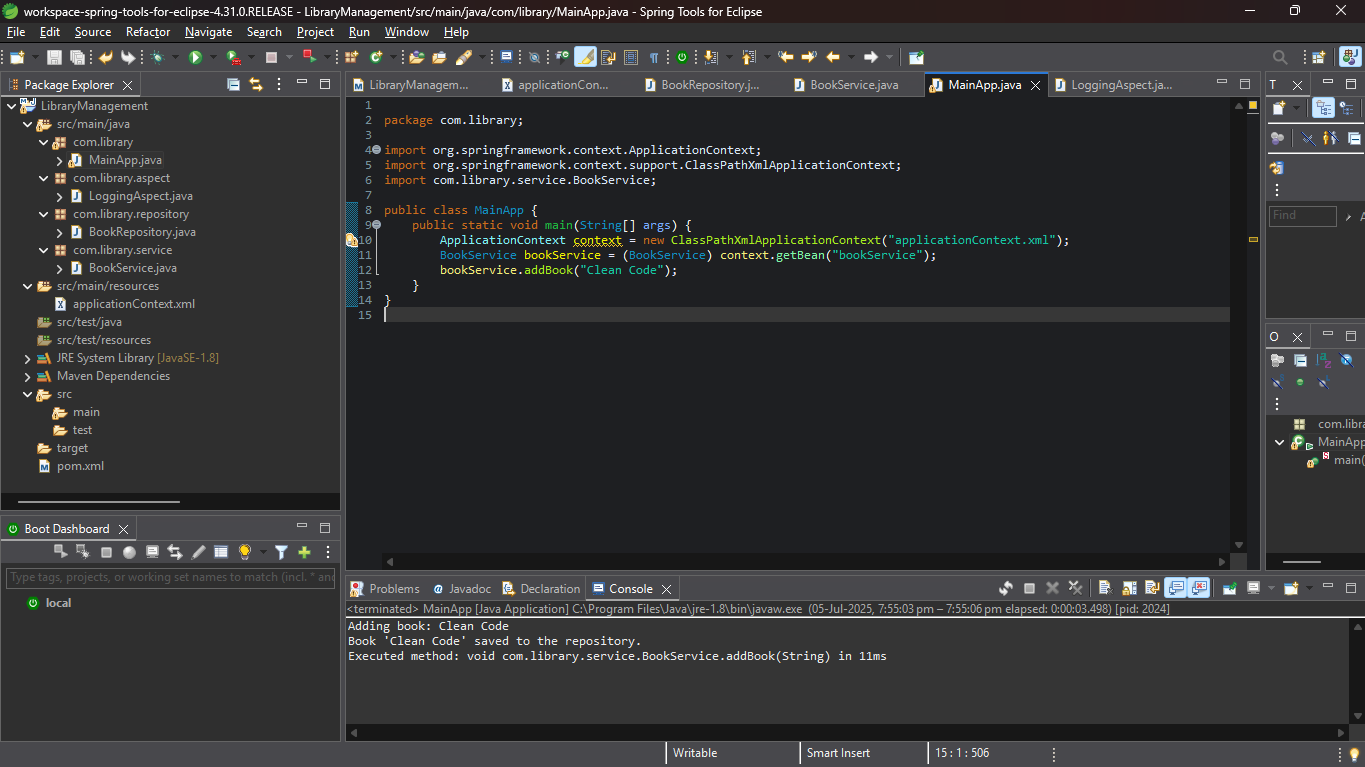
</plugin>

</plugins>

</build>

</project>

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

All implementation are done with proper outputs and ss.

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

1.

<?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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- Book Repository Bean -->

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

<!-- Book Service Bean with constructor and setter injection -->

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

<!-- Constructor Injection -->

<constructor-arg value="Library Book Service"/>

<!-- Setter Injection -->

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

</bean>

</beans>

4.main

package com.library;

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) {

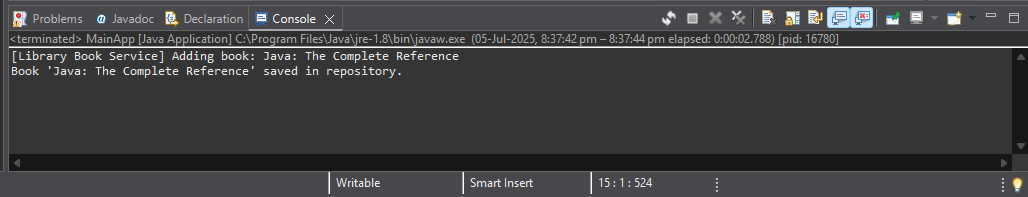
ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

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

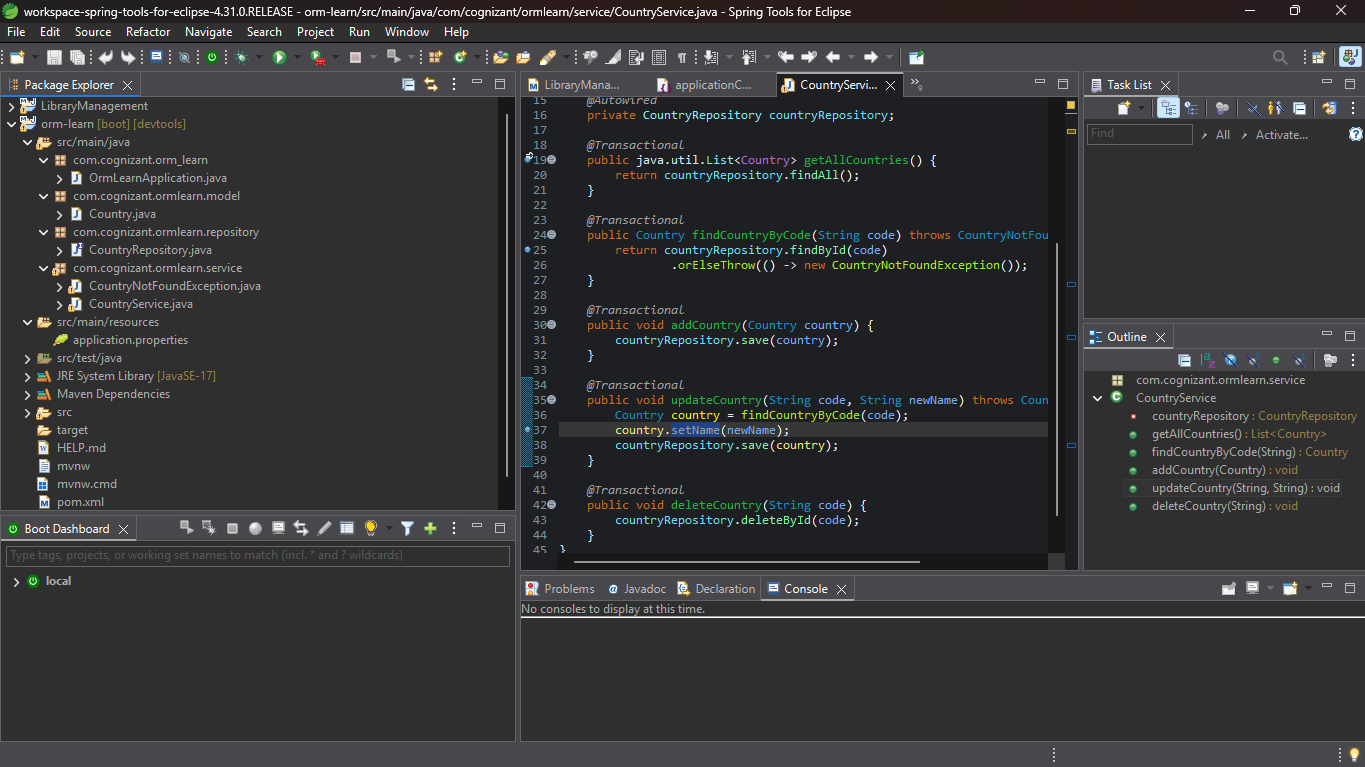
bookService.addBook("Java: The Complete Reference");

}

}



**Spring Data JPA with Spring Boot, Hibernate**

1. As per given instruction

**2.** **Difference between JPA, Hibernate and Spring Data JPA**

**1. Java Persistence API (JPA)**

| **Aspect** | **Description** |
| --- | --- |
| **What it is** | A **specification** (interface) for ORM in Java. |
| **Who defines it** | Defined by Java EE (now Jakarta EE), currently JSR 338. |
| **Purpose** | Provides a **standard API** to persist data to relational databases. |
| **Implementation** | JPA itself has **no implementation** — needs a provider like Hibernate. |
| **Key Interfaces** | EntityManager, PersistenceContext, Query |
| **Annotation Examples** | @Entity, @Id, @GeneratedValue, @ManyToOne, @OneToMany |

**2. Hibernate**

| **Aspect** | **Description** |
| --- | --- |
| **What it is** | A **concrete ORM framework** that **implements JPA**. |
| **Relation to JPA** | Hibernate is the **most popular JPA provider**. |
| **Extra Features** | Provides features **beyond JPA** like caching, filters, interceptors. |
| **Configuration** | Can use XML (hibernate.cfg.xml) or annotations. |
| **API Usage** | Hibernate can be used directly (via Session, Transaction) or via JPA. |
| **Pros** | Powerful, flexible, mature ecosystem. |

**3. Spring Data JPA**

| **Aspect** | **Description** |
| --- | --- |
| **What it is** | A **Spring module** that provides **abstraction over JPA**. |
| **Purpose** | Simplifies DAO layer with **automatic repository implementations**. |
| **Base Interface** | JpaRepository<T, ID> — prebuilt CRUD methods. |
| **Query Methods** | Method naming convention for queries (e.g., findByName() auto-implemented) |
| **No Boilerplate** | Removes need to write EntityManager or JPQL manually. |
| **Implementation** | Uses a JPA provider (like Hibernate) under the hood. |