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

**Scenario:**

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Steps:**

1. **Set Up a Spring Project:**
   * Create a Maven project named **LibraryManagement**.
   * Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   * Create a package **com.library.service** and add a class **BookService**.
   * Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration.

Answer:

BookRepository.java

**package** com.library.repository;

**public** **class** BookRepository {

**public** BookRepository() {

System.***out***.println("BookRepository initialized.");

}

**public** **void** save(String bookName) {

System.***out***.println("Saving book: " + bookName + " to the database (simulated).");

}

}

BookServices.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** **final** BookRepository bookRepository;

**public** BookService(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

System.***out***.println("BookService initialized with BookRepository.");

}

**public** **void** addBook(String bookName) {

System.***out***.println("BookService: Attempting to add book: " + bookName);

bookRepository.save(bookName);

System.***out***.println("BookService: Book '" + bookName + "' added successfully.");

}

}

MainApplication.java

**package** com.lib;

**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) {

System.***out***.println("Starting Spring Application...");

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

System.***out***.println("Spring ApplicationContext loaded successfully.");

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

System.***out***.println("BookService bean retrieved from context.");

bookService.addBook("The Great Gatsby");

bookService.addBook("1984");

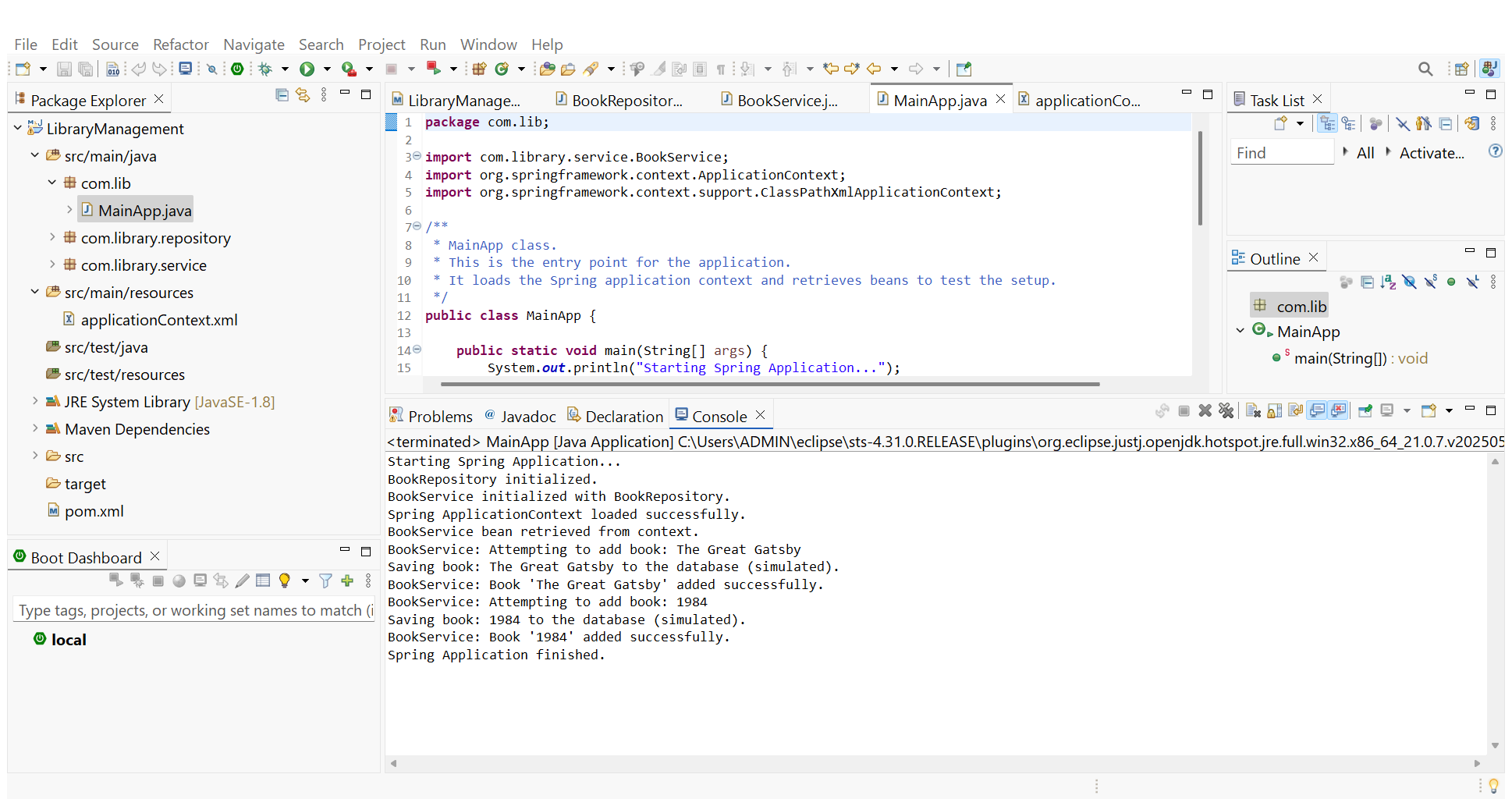
((ClassPathXmlApplicationContext) context).close();

System.***out***.println("Spring Application finished.");

}

}

Output Screenshot:



**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.

Answer:

BookRepository.java

**package** com.library.repository;

**public** **class** BookRepository {

**public** BookRepository() {

System.***out***.println("BookRepository initialized.");

}

**public** **void** save(String bookName) {

System.***out***.println("Saving book: " + bookName + " to the database (simulated).");

}

}

BookServices.java

**package** com.library.service;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

**public** BookService() {

System.***out***.println("BookService: Default constructor called.");

}

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

System.***out***.println("BookService: BookRepository dependency set via setter.");

}

**public** **void** addBook(String bookName) {

**if** (bookRepository == **null**) {

System.***err***.println("BookService: Error - BookRepository is not set!");

**return**;

}

System.***out***.println("BookService: Attempting to add book: " + bookName);

bookRepository.save(bookName);

System.***out***.println("BookService: Book '" + bookName + "' added successfully.");

}

}

MainApp.java

**package** com.lib;

**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) {

System.***out***.println("Starting Spring Application...");

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

System.***out***.println("Spring ApplicationContext loaded successfully.");

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

System.***out***.println("BookService bean retrieved from context.");

bookService.addBook("The Great Gatsby");

bookService.addBook("1984");

((ClassPathXmlApplicationContext) context).close();

System.***out***.println("Spring Application finished.");

}

}

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 -->

<!-- This bean represents our data access layer for books. -->

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

<!-- No properties to set for now, as it's a simple mock repository -->

</bean>

<!-- Define BookService Bean -->

<!-- This bean represents our business logic layer for books. -->

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

<!-- Inject the bookRepository into BookService using setter injection -->

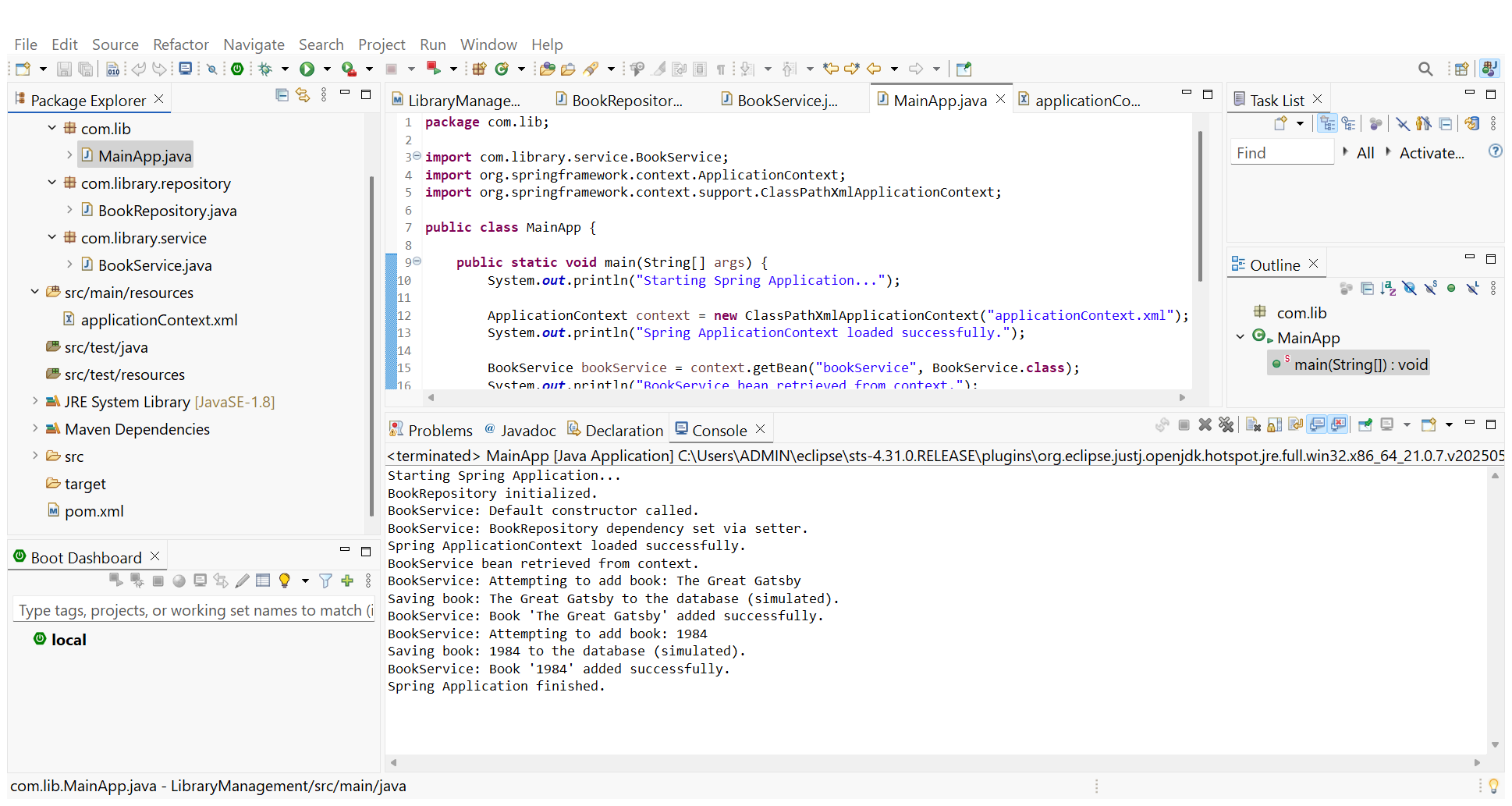
<!-- The 'name' attribute corresponds to the setter method name (setBookRepository) -->

<!-- The 'ref' attribute points to the 'bookRepository' bean defined above. -->

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

</bean>

</beans>



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

**Scenario:**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**Steps:**

1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

Answer:

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.librarymanagement</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

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

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

<spring.version>5.3.37</spring.version> </properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.36</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.13</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.11.0</version> <scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.13.0</version> <configuration>

<source>${maven.compiler.source}</source>

<target>${maven.compiler.target}</target>

</configuration>

</plugin>

</plugins>

</build>

</project>