# Exercise 1: Configuring a Basic Spring Application

# Step 1: Create Maven Project in Eclipse

1. Open Eclipse IDE.  
2. Go to File → New → Maven Project.  
3. Enter:  
 - Group Id: com.library  
 - Artifact Id: Library  
4. Click Finish.

# Step 2: Add Spring Dependencies to 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-SNAPSHOT</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 </dependencies>  
</project>

# Step 3: Create Package and Classes

BookRepository.java:  
package com.library.repository;  
  
public class BookRepository {  
 public void displayBooks() {  
 System.out.println("Displaying list of books from the repository...");  
 }  
}

# Step 4: Create BookService.java

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 listBooks() {  
 System.out.println("BookService: Calling BookRepository to list books...");  
 bookRepository.displayBooks();  
 }  
}

# Step 5: Create 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>

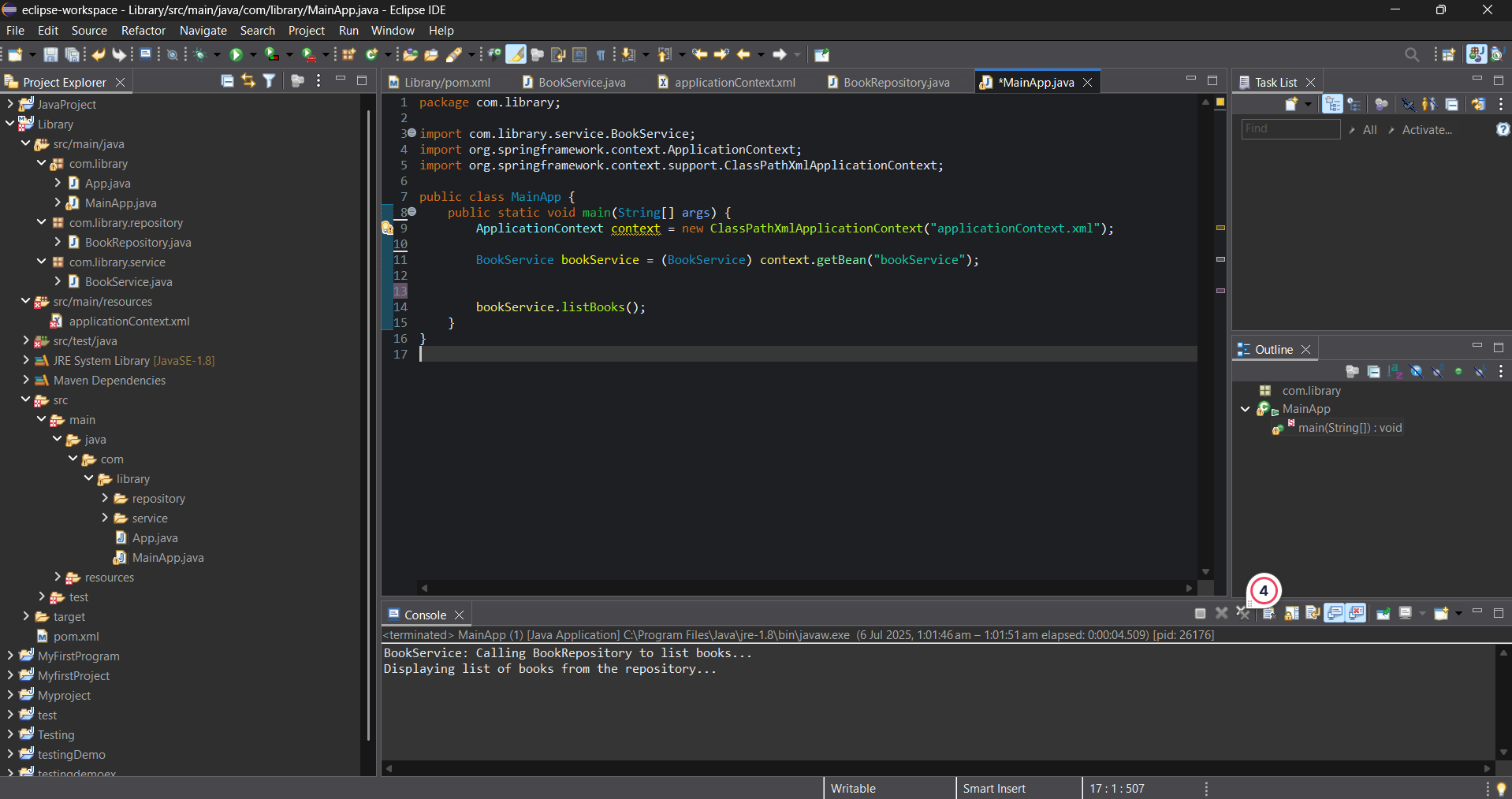
# Step 6: Create MainApp.java

MainApp.java:  
package com.library;  
  
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) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService service = (BookService) context.getBean("bookService");  
 service.listBooks();  
 }  
}

# Step 7: Run the Application

Right-click MainApp.java → Run As → Java Application  
Expected Output:  
  
BookService: Calling BookRepository to list books...  
Displaying list of books from the repository...

**Output:**



**Exercise 2: Implementing Dependency Injection**

# Step 1: Modify XML Configuration (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>

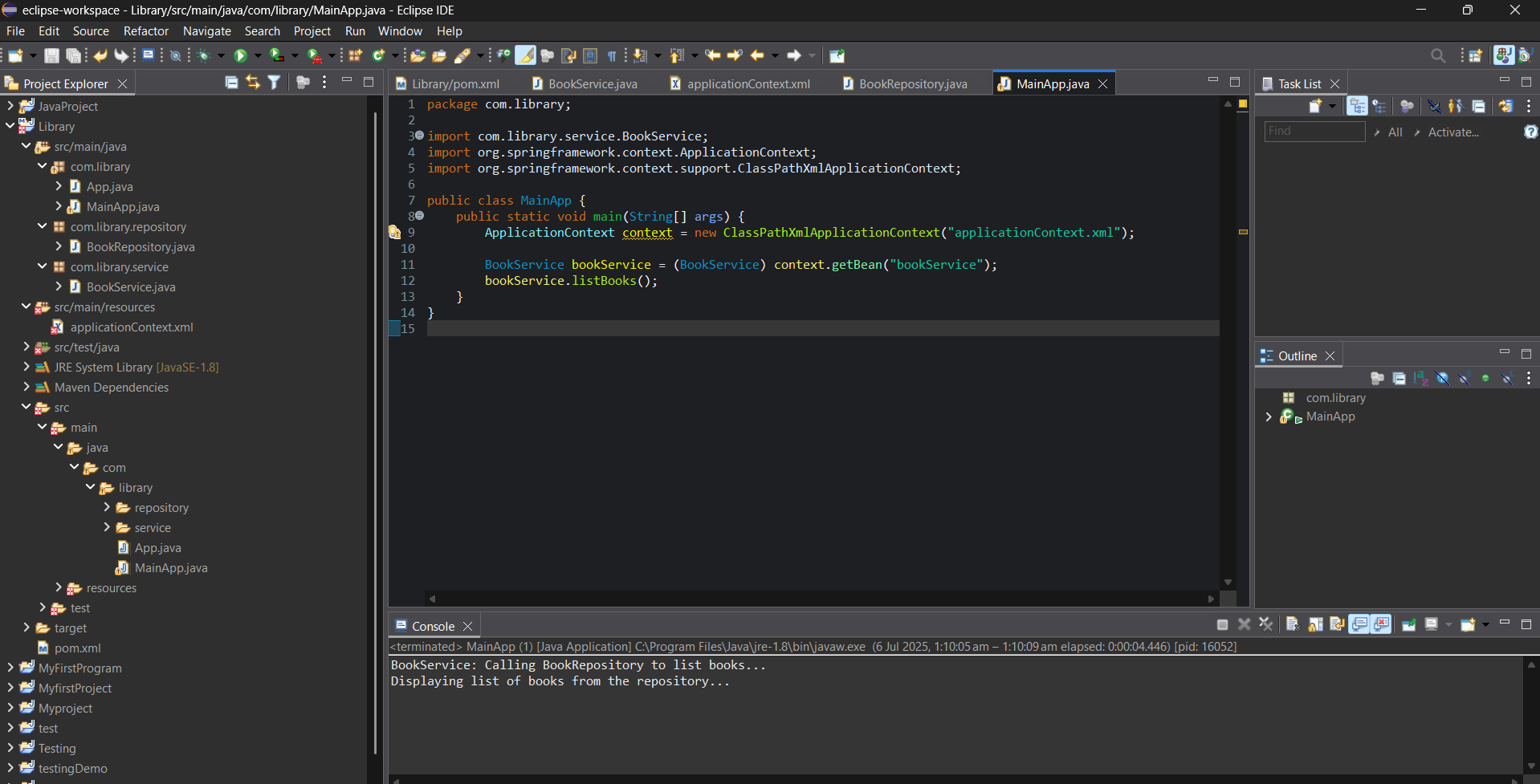
# Step 2: Update BookService.java with Setter Injection

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 listBooks() {  
 System.out.println("BookService: Calling BookRepository to list books...");  
 bookRepository.displayBooks();  
 }  
}

# Step 3: Test Dependency Injection

Right-click MainApp.java → Run As → Java Application

**Output:**



# Exercise 4: Creating and Configuring a Maven Project

# Step 1: Create a New Maven Project

1. Open Eclipse IDE.  
2. Go to File → New → Maven Project.  
3. Check Create a simple project.  
4. Click Next.  
5. Fill in the following:  
 - Group Id: com.library  
 - Artifact Id: Lib  
 - Version: 1.0-SNAPSHOT  
6. Click Finish.  
  
This will create the basic Maven project structure in Eclipse.

# Step 2: Add Spring Dependencies to 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>Lib</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 </dependencies>

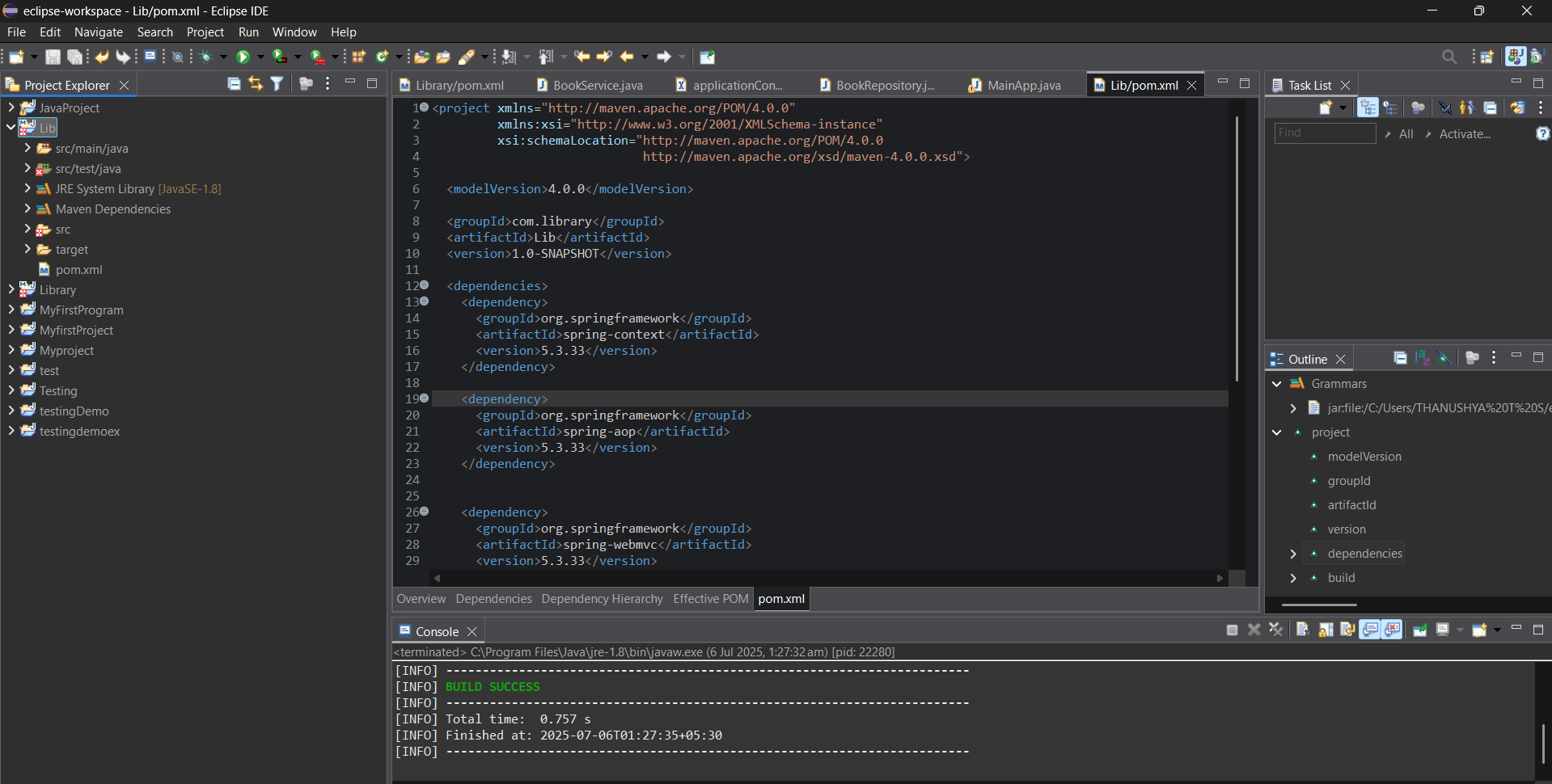
# Step 3: Configure Maven Compiler Plugin

<build>  
 <plugins>  
 <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>

# Step 4: Build and Validate

1. Right-click on the project → Maven → Update Project   
2. Then go to Run As → Maven clean.  
3. Then go to Run As → Maven install.  
4. If everything is configured properly, you should see:  
 BUILD SUCCESS

**Output:**

****

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

# Step 1: Create Spring Configuration File (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 for BookRepository -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository"/>  
  
 <!-- Bean for BookService with dependency injection -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

# Step 2: Create BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void displayBooks() {  
 System.out.println("Displaying list of books from the repository...");  
 }  
}

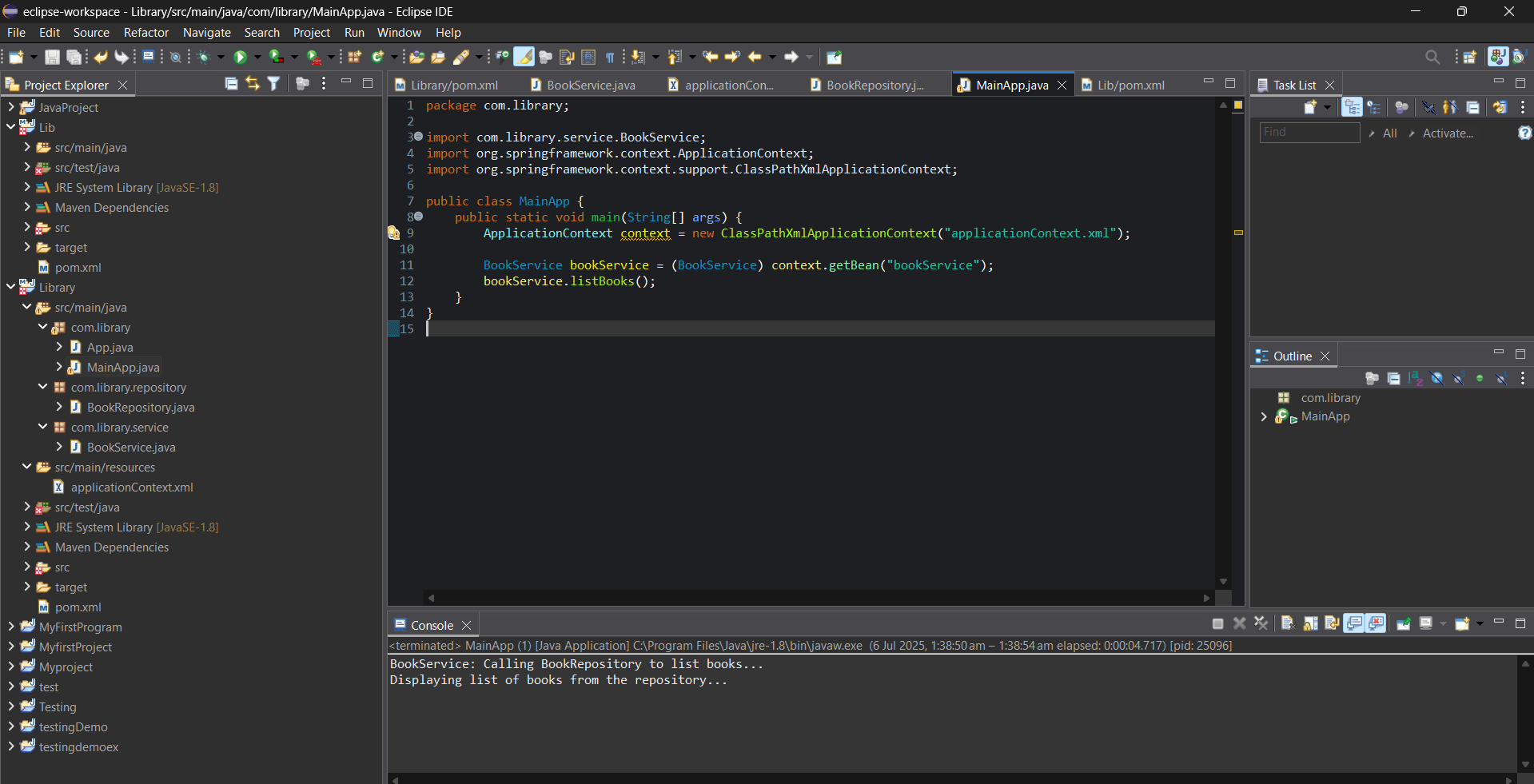
# Step 3: Create BookService.java

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 listBooks() {  
 System.out.println("BookService: Calling BookRepository to list books...");  
 bookRepository.displayBooks();  
 }  
}

# Step 4: Create MainApp.java

package com.library;  
  
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) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.listBooks();  
 }  
}

**Output:**

****

# Exercise 7: Implementing Constructor and Setter Injection

# Step 1: Configure Constructor Injection

<bean id="bookService" class="com.library.service.BookService">  
 <constructor-arg ref="bookRepository"/>  
</bean>

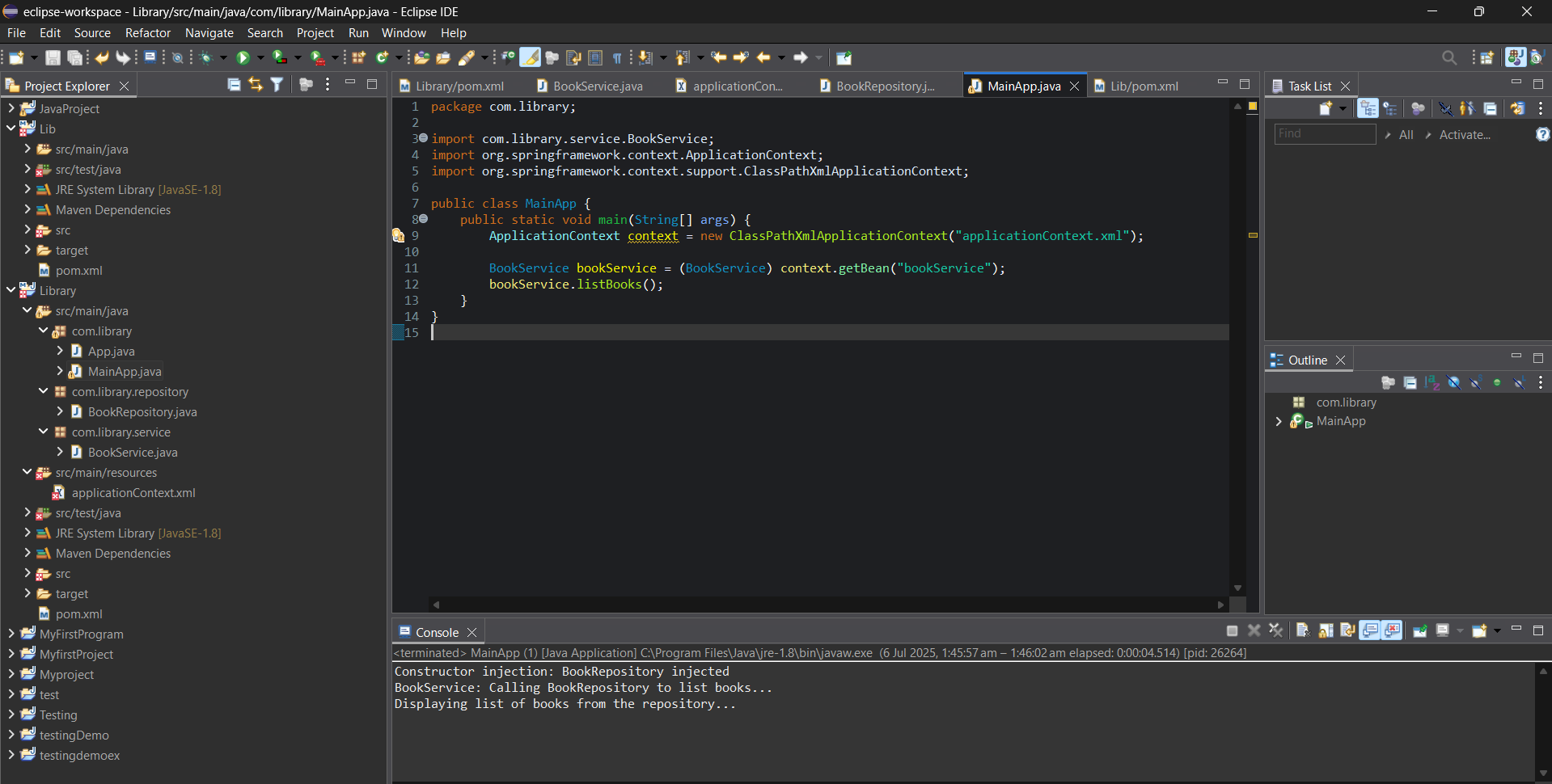
# Step 2: Configure Setter Injection

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public BookService(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 System.out.println("Constructor injection: BookRepository injected");  
 }  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 System.out.println("Setter injection: BookRepository injected");  
 }  
  
 public void listBooks() {  
 System.out.println("BookService: Calling BookRepository to list books...");  
 bookRepository.displayBooks();  
 }  
}

# Step 3: Test the Injection

package com.library;  
  
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) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.listBooks();  
 }  
}

**Output:**

****

**Exercise 9: Creating a Spring Boot Application**

**Step 1: Create a Spring Boot Project**

* Go to [https://start.spring.io](https://start.spring.io/)
* Project: Maven
* Language: Java
* Spring Boot: 3.5.3
* Project Name: LibraryManagement
* Package Name: com.library
* Dependencies:
  + Spring Web
  + Spring Data JPA
  + H2 Database

Click **Generate**, unzip the project, and open it in **Eclipse.**

**Step 2: Add Application Properties**

In src/main/resources/application.properties, add:

spring.datasource.url=jdbc:h2:mem:librarydb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

**Step 3: Define the Book Model**

In com.library.model.Book:

package com.library.model;

import jakarta.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getAuthor() { return author; }

public void setAuthor(String author) { this.author = author; }

}

**Step 4: Create BookRepository**

import com.library.repository.BookRepository:

package com.library.repository;

import com.library.model.Book;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {

}

**Step 5: Create BookController**

Import com.library.controller.BookController:

package com.library.controller;

import com.library.model.Book;

import com.library.repository.BookRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

@Autowired

private BookRepository bookRepository;

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@GetMapping

public List<Book> getBooks() {

return bookRepository.findAll();

}

}

**Step 6: Run the Application**

* Run LibraryManagementApplication.java

**Access APIs:**

* GET: http://localhost:8080/books
* POST: http://localhost:8080/books

**Sample POST JSON:**

{

"title": "Effective Java",

"author": "Joshua Bloch"

}

**Output:**

