# Spring Core and Maven

# Mandatory Hands-On

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

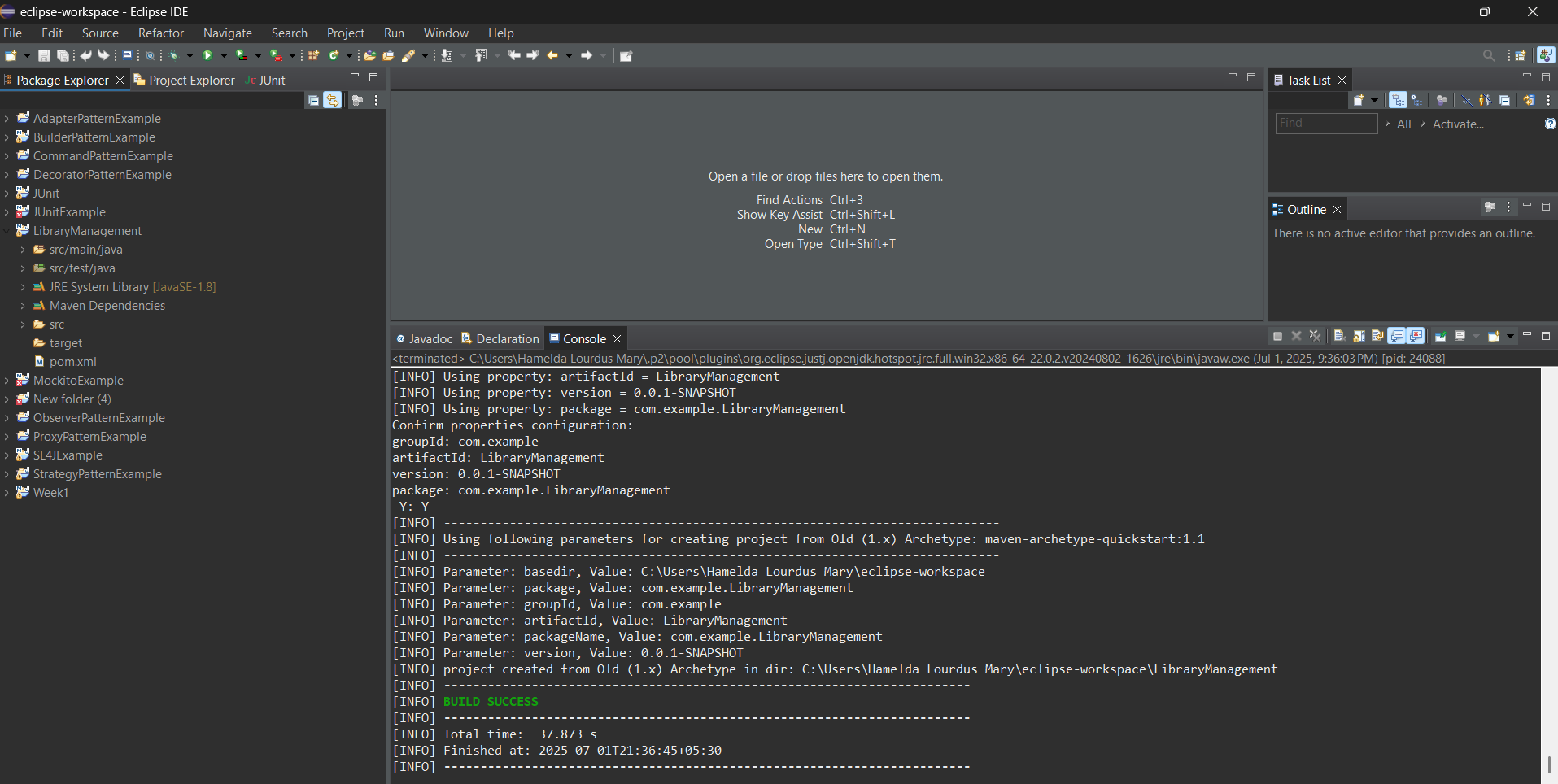
**Scenario:**

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.**

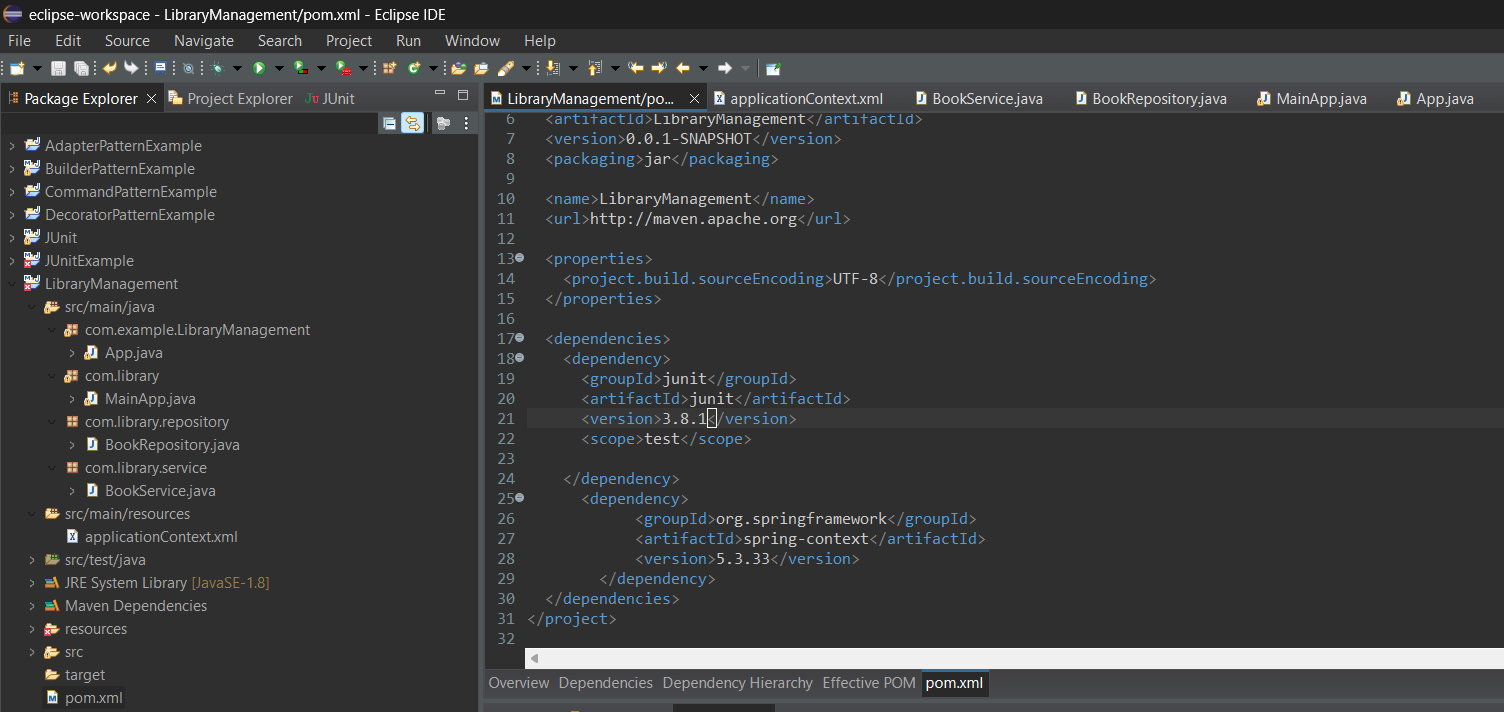
<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>



1. **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.

<?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" />

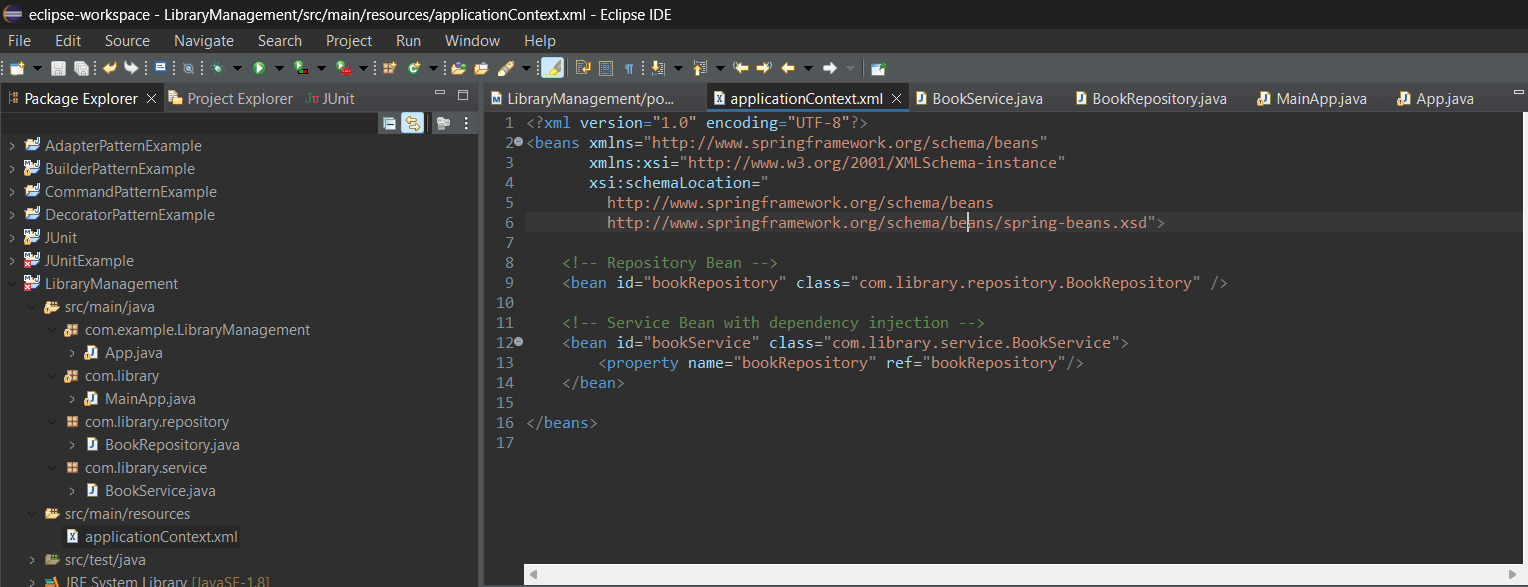
<!-- Service Bean with dependency injection -->

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

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

</bean>

</beans>



1. **Create a service class:**

Create a package **com.library.service** and add a class **BookService**.

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.out.println("[Service] Request received to add book: " + bookName);

bookRepository.save(bookName);

}

}

**create a repository class:**

Create a package **com.library.repository** and add a class **BookRepository**.

package com.library.repository;

public class BookRepository {

public void save(String bookName) {

System.out.println("[Repository] Book '" + bookName + "' has been saved to the database.");

}

}

1. **Run the Application:**

Create a main class to load the Spring context and test the configuration.

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");

System.out.println("[Main] Spring context loaded successfully.");

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

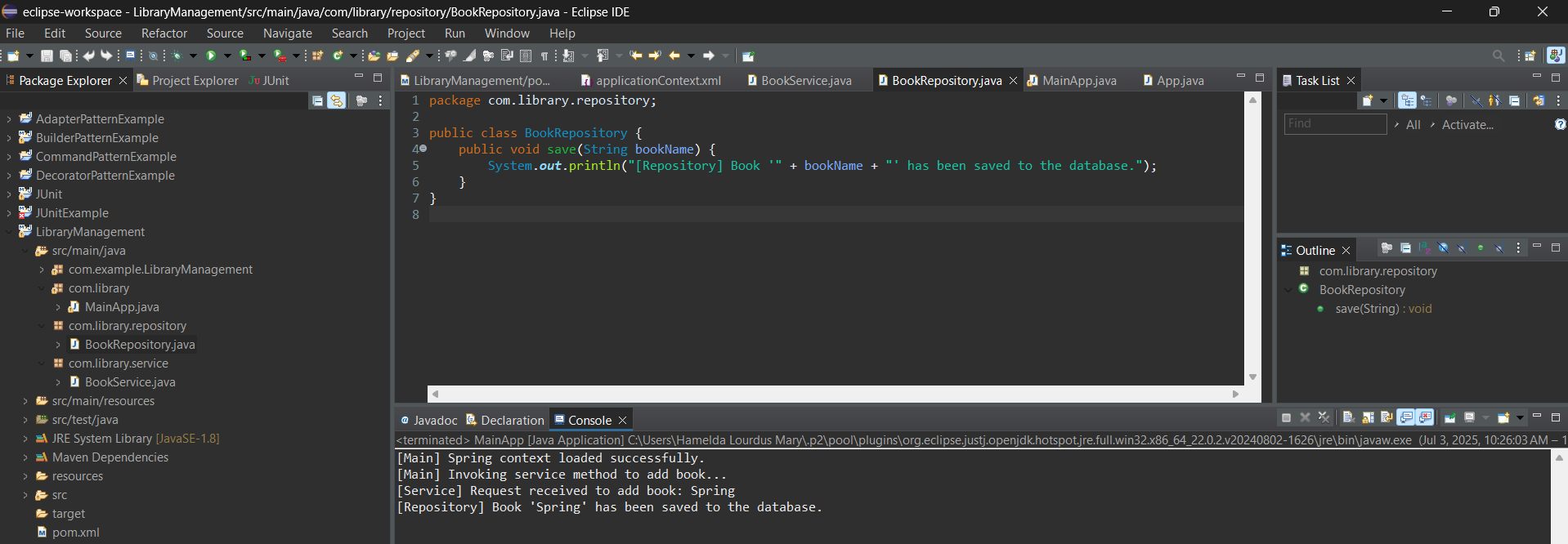
System.out.println("[Main] Invoking service method to add book...");

bookService.addBook("Clean Code");

}

}

**Output:**



**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.

**1.Modify the XML Configuration:**

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

<?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 Repository Bean -->

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

<!-- Define Service Bean and Inject Repository -->

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

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

</bean>

</beans>

1. **Update the BookService Class:**

Ensure that **BookService** class has a setter method for **BookRepository**.

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) {

System.out.println("[Service] Setter-based injection of BookRepository successful.");

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.out.println("[Service] Processing request to add book: " + bookName);

bookRepository.save(bookName);

}

}

**create a repository class:**

package com.library.repository;

public class BookRepository {

public void save(String bookName) {

System.out.println("[Repository] Book '" + bookName + "' saved successfully.");

}

}

1. **Test the Configuration:**

Run the **LibraryManagementApplication** main class to verify the dependency injection.

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

System.out.println("[Main] Spring context initialized.");

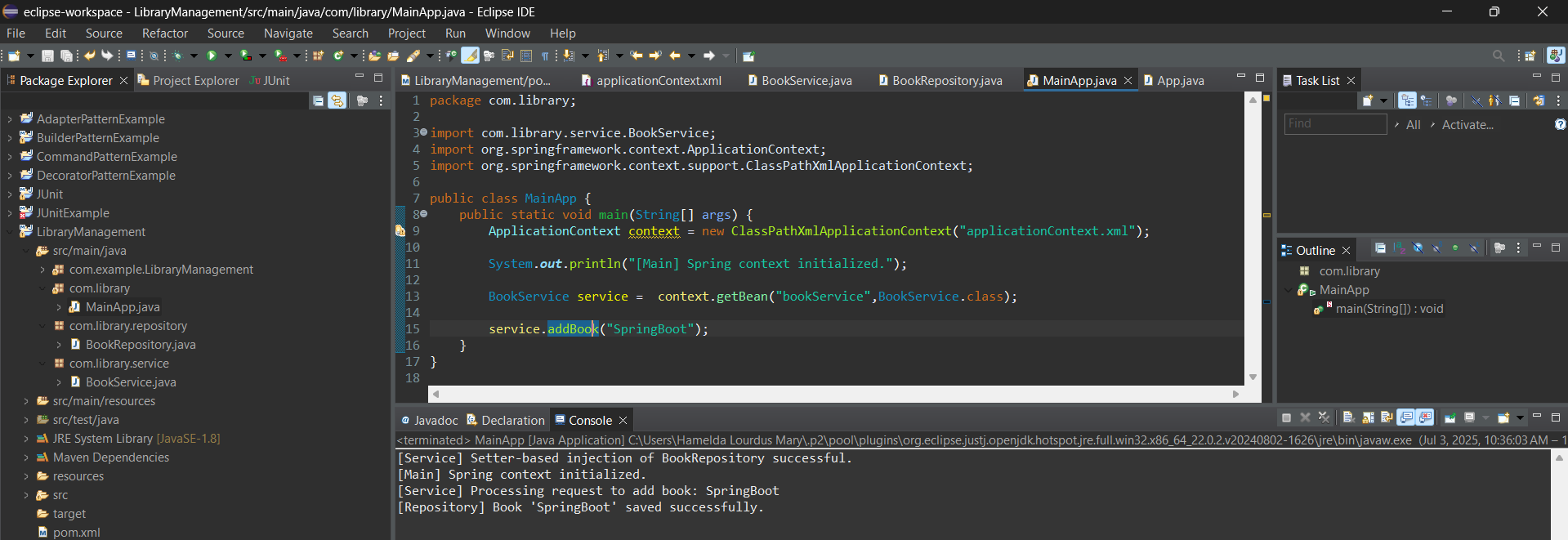
BookService service = (BookService) context.getBean("bookService");

service.addBook("Design Patterns");

}

}

**Output:**



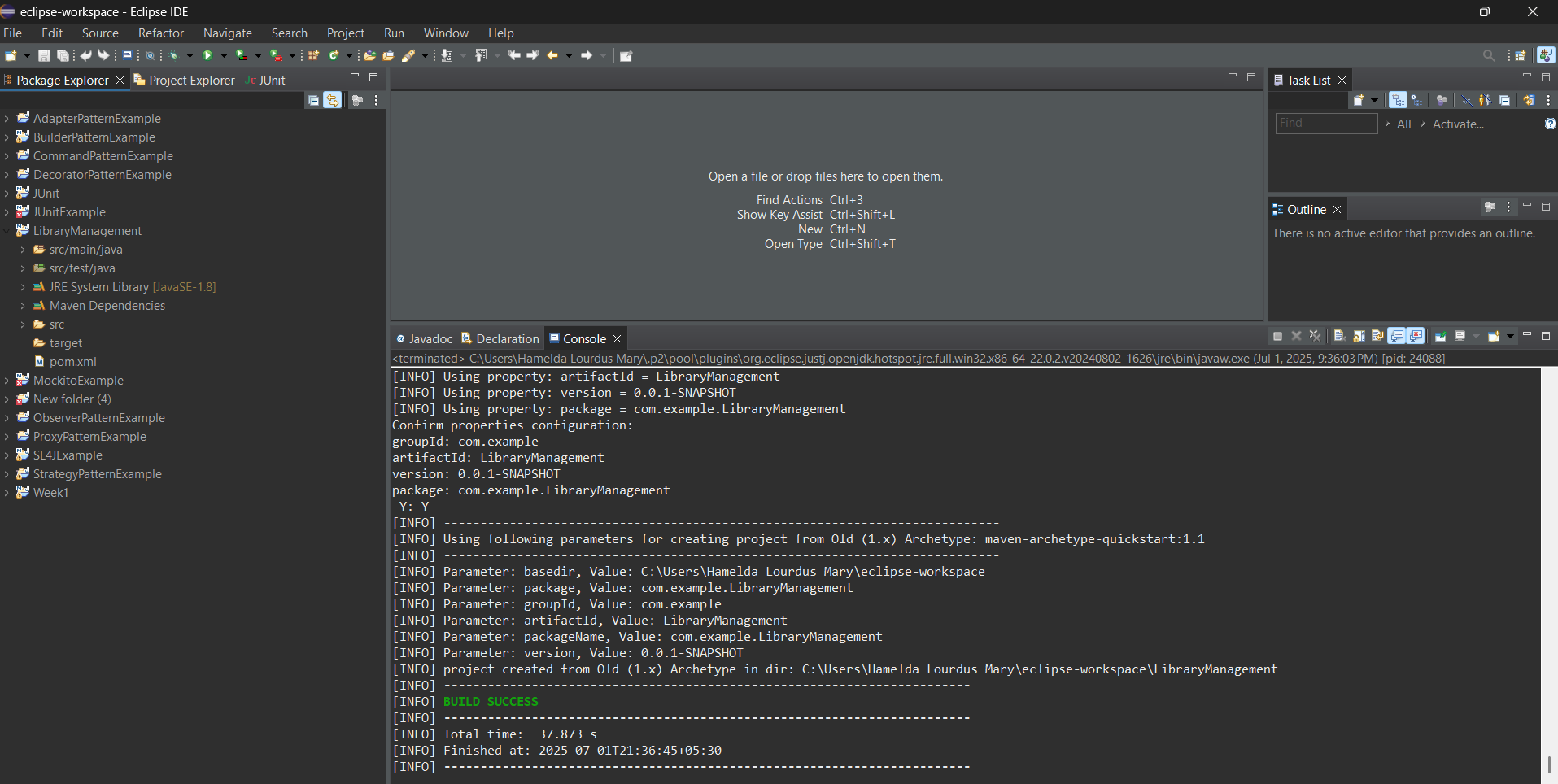
**Exercise 4: Creating and Configuring a Maven Project**

**Scenario:**

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

1. **Create a New Maven Project:**

Create a new Maven project named **LibraryManagement**.



1. **Add Spring Dependencies in pom.xml:**

Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.

<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>

***<!-- Spring Core & Context -->***

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

***<!-- Spring AOP -->***

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

***<!-- Spring Web MVC -->***

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

***<!-- AspectJ Weaver (for AOP) -->***

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21</version>

</dependency>

***<!-- Servlet API (for Spring MVC) -->***

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

**<!-- Step 3: Configure Maven Compiler Plugin -->**

<build>

<plugins>

<plugin>

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

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

<version>3.10.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

1. **Run the Application:**

**MainApp.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

System.*out*.println("Library Management Application Started");

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

} }

**Output:**

