**Exercise 8: Implementing Basic AOP with Spring**

**Step 1: Define an Aspect**

Create a package com.library.aspect and add a class LoggingAspect:

package com.library.aspect;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.After;

@Aspect

public class LoggingAspect {

// Advice methods will be defined here

}

**Step 2: Create Advice Methods**

Define advice methods in LoggingAspect for logging before and after method execution:

@Before("execution(\* com.library.service.BookService.\*(..))")

public void logBeforeMethodExecution() {

System.out.println("Logging before method execution...");

}

@After("execution(\* com.library.service.BookService.\*(..))")

public void logAfterMethodExecution() {

System.out.println("Logging after method execution...");

}

**Step 3: Configure the Aspect**

Update applicationContext.xml to register the aspect and enable AspectJ auto-proxying:

<beans xmlns="(link unavailable)"

xmlns:xsi="(link unavailable)"

xmlns:aop="(link unavailable)"

xsi:schemaLocation="(link unavailable)

(link unavailable)

(link unavailable)">

<aop:aspectj-autoproxy/>

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

</beans>

**Step 4: Test the Aspect**

Run the LibraryManagementApplication main class to verify the AOP functionality:

package com.library;

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");

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

bookService.someMethod(); // This will trigger the logging aspect

}

}

**Output:**

Logging before method execution...

Logging after method execution...