**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.

**CODE:**

LibraryManagementApp

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class LibraryManagementApp {

@Bean

public BookRepository bookRepository() {

return new BookRepository();

}

@Bean

public BookService bookService() {

return new BookService(bookRepository());

}

public static void main(String[] args) {

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

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

System.out.println("BookService bean loaded: " + (bookService != null));

bookService.performService();

}

}

@Service

class BookService {

private final BookRepository bookRepository;

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performService() {

System.out.println("Service performed.");

bookRepository.performRepositoryAction();

}

}

@Repository

class BookRepository {

public void performRepositoryAction() {

System.out.println("Repository action performed.");

}

}

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.library</groupId> <artifactId>exercise1</artifactId>

<version>1.0-SNAPSHOT</version> <properties> <maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target> <spring.version>5.2.8.RELEASE</spring.version>

</properties> <dependencies> <dependency> <groupId>org.springframework</groupId> <artifactId>spring-

core</artifactId> <version>6.1.11</version> </dependency> <dependency>

<groupId>org.springframework</groupId> <artifactId>spring-context</artifactId> <version>6.1.11</version>

</dependency> </dependencies> <build> <sourceDirectory>src</sourceDirectory> <plugins> <plugin>

<artifactId>maven-compiler-plugin</artifactId> <version>3.8.1</version> <configuration>

<source>1.8</source> <target>1.8</target> </configuration> </plugin> </plugins> </build> </project>

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">

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

<bean id="bookService" class="main.java.com.library.service.BookService"/></beans>

**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.

**CODE:**

LibraryManagementApp

package main.java.com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.Random;

public class LibraryManagementApp {

public static void main(String[] args) {

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

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

bookService.askBookRepo();

}

public static class BookService {

private BookRepository bookRepo;

public void setBookRepository(BookRepository bookRepo) {

this.bookRepo = bookRepo;

}

public void askBookRepo() {

if (bookRepo.hasBooks()) {

System.out.println("Books are available in the repository.");

} else {

System.out.println("No books found in the repository.");

}

}

}

public static class BookRepository {

public Boolean hasBooks() {

Random random = new Random();

return random.nextBoolean();

}

}

}

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.library</groupId>

<artifactId>exercise2</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

<spring.version>5.2.8.RELEASE</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.11</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

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">

<bean id="bookRepository" class="main.java.com.library.Repository.BookRepository"/>

<!-- here we wired the bookrepository class and bookservice class-->

<bean id="bookService" class="main.java.com.library.Service.BookService">

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

</bean>

</beans>

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

**Scenario:**

The library management application requires logging capabilities to track method execution times.

**Steps:**

1. **Add Spring AOP Dependency:**
   * Update **pom.xml** to include Spring AOP dependency.
2. **Create an Aspect for Logging:**
   * Create a package **com.library.aspect** and add a class **LoggingAspect** with a method to log execution times.
3. **Enable AspectJ Support:**
   * Update **applicationContext.xml** to enable **AspectJ** support and register the aspect.
4. **Test the Aspect:**
   * Run the **LibraryManagementApplication** main class and observe the console for log messages indicating method execution times.

**CODE:**

LibraryManagement.java

package main.java.com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.springframework.stereotype.Component;

import java.util.Random;

public class LibraryManagementApp {

public static void main(String[] args) {

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

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

System.out.println("BookService bean loaded: " + (bookService != null));

bookService.askBookRepo();

}

public static class BookService {

private BookRepository bookRepo;

public void setBookRepository(BookRepository bookRepo) {

this.bookRepo = bookRepo;

}

public void askBookRepo() {

if (bookRepo.hasBooks()) {

System.out.println("Books are available in the repository.");

} else {

System.out.println("No books found in the repository.");

}

}

}

public static class BookRepository {

public Boolean hasBooks() {

Random random = new Random();

return random.nextBoolean();

}

}

@Aspect

@Component

public static class LoggingAspect {

@Around("execution(\* main.java.com.library..\*(..))")

public Object logExecutionTimes(ProceedingJoinPoint pjp) throws Throwable {

long startTime = System.currentTimeMillis();

Object proceed = pjp.proceed();

long endTime = System.currentTimeMillis();

System.out.println(pjp.getSignature().toShortString() + " executed in " + (endTime - startTime) + "ms");

return proceed;

}

}

}

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.library</groupId>

<artifactId>exercise3</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

<spring.version>5.2.8.RELEASE</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.5.4</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.2</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjtools</artifactId>

<version>1.9.2</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

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"

xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="

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

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

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

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

<!-- Define beans -->

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

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

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

</bean>

<!-- Enable AspectJ auto-proxying -->

<aop:aspectj-autoproxy />

<!-- Register the LoggingAspect -->

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

</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.

**CODE:**

LibraryManagementApp.java

@SpringBootApplication

public class LibraryManagementApp {

public static void main(String[] args) {

System.out.println("Welcome to Library Management Application!");

}

}

@SpringBootTest

class LibraryManagementAppTests {

@Test

void contextLoads() {

}

}

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.lms</groupId>

<artifactId>exercise4</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>6.1.11</version>

</dependency>

</dependencies>

<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>

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

**Scenario:**

The library management application requires a central configuration for beans and dependencies.

**Steps:**

1. **Create Spring Configuration File:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.
2. **Update the BookService Class:**
   * Ensure that the **BookService** class has a setter method for **BookRepository**.
3. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration.

**CODE:**

JavaFile

package main.java.com.exercise5;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

@SuppressWarnings("unused")

public static void main(String[] args) {

@SuppressWarnings("resource")

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

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

System.out.println("BookService bean loaded, Configuration Success");

}

public static class BookService {

public BookRepository bookRepo;

public void setBookRepository(BookRepository bookRepo) {

this.bookRepo = bookRepo;

}

}

public static class BookRepository {

}

}

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.exercise5</groupId>

<artifactId>exercise5</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.5.4</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.2</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

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">

<bean id="bookRepository" class="main.java.com.exercise5.repository.BookRepository"/>

<bean id="bookService" class="main.java.com.exercise5.service.BookService">

</bean>

</beans>

**Exercise 6: Configuring Beans with Annotations**

**Scenario:**

You need to simplify the configuration of beans in the library management application using annotations.

**Steps:**

1. **Enable Component Scanning:**
   * Update **applicationContext.xml** to include component scanning for the **com.library** package.
2. **Annotate Classes:**
   * Use **@Service** annotation for the **BookService** class.
   * Use **@Repository** annotation for the **BookRepository** class.
3. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the annotation-based configuration.

**CODE:**

Java File

package main.java.com.exercise6;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import org.springframework.stereotype.Repository;

public class Main {

@SuppressWarnings({ "unused", "resource" })

public static void main(String[] args) {

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

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

System.out.println("Successfully verified annotation-based configuration.");

}

@Service

public static class BookService {

private final BookRepository bookRepo;

public BookService(BookRepository bookRepo) {

this.bookRepo = bookRepo;

}

}

@Repository

public static class BookRepository {

// BookRepository implementation

}

}

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.exercise6</groupId>

<artifactId>exercise6</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

<spring.version>5.2.8.RELEASE</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.11</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

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"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

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

http://www.springframework.org/schema/context/spring-context.xsd">

<!-- Enable component scanning for the com.library package -->

<context:component-scan base-package="main.java.com.exercise6"/>

</beans>

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

**Scenario:**

The library management application requires both constructor and setter injection for better control over bean initialization.

**Steps:**

1. **Configure Constructor Injection:**
   * Update applicationContext.**xml** to configure constructor injection for **BookService**.
2. **Configure Setter Injection:**
   * Ensure that the **BookService** class has a setter method for **BookRepository** and configure it in **applicationContext.xml**.
3. **Test the Injection:**
   * Run the **LibraryManagementApplication** main class to verify both constructor and setter injection.

**CODE:**

Java File

package main.java.com.exercise7;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import java.util.Random;

public class Main {

public static void main(String[] args) {

@SuppressWarnings("resource")

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

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

System.out.println(bookService.getAns());

int bookCount = bookService.getBookCount();

System.out.println("Number of books: " + bookCount);

}

@Repository

public static class BookRepository {

public int getBookCount() {

Random r = new Random();

return r.nextInt(1000);

}

public String gotAns() {

return "BookService bean retrieved successfully through constructor injection.";

}

}

@Service

public static class BookService {

private final BookRepository bookRepo;

private BookRepository br;

public BookService(BookRepository bookRepo) {

this.br = bookRepo;

}

public void setBookRepository(BookRepository bookRepo) {

this.bookRepo = bookRepo;

}

public int getBookCount() {

return bookRepo.getBookCount();

}

public String getAns() {

return br.gotAns();

}

}

}

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.exercise7</groupId>

<artifactId>exercise7</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.11</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

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"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

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

http://www.springframework.org/schema/context/spring-context.xsd">

<!-- Enable component scanning for the com.library package -->

<context:component-scan base-package="com.exercise7"/>

<bean id="bookRepository" class="main.java.com.exercise7.repository.BookRepository"/>

<bean id="bookService" class="main.java.com.exercise7.service.BookService">

<constructor-arg ref="bookRepository"/>

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

</bean>

</beans>

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

**Scenario:**

The library management application requires basic AOP functionality to separate cross-cutting concerns like logging and transaction management.

**Steps:**

1. **Define an Aspect:**
   * Create a package **com.library.aspect** and add a class **LoggingAspect**.
2. **Create Advice Methods:**
   * Define advice methods in **LoggingAspect** for logging before and after method execution.
3. **Configure the Aspect:**
   * Update **applicationContext.xml** to register the aspect and enable **AspectJ** auto-proxying.
4. **Test the Aspect:**
   * Run the **LibraryManagementApplication** main class to verify the AOP functionality.

**CODE:**

Java File

package main.java.com.exercise8;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Component;

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import java.util.Random;

public class Main {

public static void main(String[] args) {

@SuppressWarnings("resource")

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

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

System.out.println(bookService.getAns());

int bookCount = bookService.getBookCount();

System.out.println("Number of books: " + bookCount);

}

@Aspect

@Component

public static class LoggingAspect {

@Before("execution(\* main.java.com.exercise8.service.BookService.\*(..))")

public void logBefore() {

System.out.println("LoggingAspect: Before method execution.");

}

@After("execution(\* main.java.com.exercise8.service.BookService.\*(..))")

public void logAfter() {

System.out.println("LoggingAspect: After method execution.");

}

}

@Repository

public static class BookRepository {

public int getBookCount() {

Random r = new Random();

return r.nextInt(1000);

}

public String gotAns() {

return "BookService bean retrieved successfully through constructor injection.";

}

}

@Service

public static class BookService {

private final BookRepository bookRepo;

private BookRepository br;

public BookService(BookRepository bookRepo) {

this.br = bookRepo;

}

public void setBookRepository(BookRepository bookRepo) {

this.bookRepo = bookRepo;

}

public int getBookCount() {

return bookRepo.getBookCount();

}

public String getAns() {

return br.gotAns();

}

}

}

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.exercise8</groupId>

<artifactId>exercise8</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.5.4</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.2</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

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"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

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

http://www.springframework.org/schema/context/spring-context.xsd

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

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

<!-- Enable component scanning for the com.library package -->

<context:component-scan base-package="com.exercise8"/>

<bean id="bookRepository" class="main.java.com.exercise8.repository.BookRepository"/>

<bean id="bookService" class="main.java.com.exercise8.service.BookService">

<constructor-arg ref="bookRepository"/>

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

</bean>

<aop:aspectj-autoproxy/>

<!-- Register LoggingAspect as a bean -->

<bean id="loggingAspect" class="main.java.com.exercise8.aspect.LoggingAspect"/>

</beans>

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

**Scenario:**

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

**Steps:**

1. **Create a Spring Boot Project:**
   * Use **Spring Initializr** to create a new Spring Boot project named **LibraryManagement**.
2. **Add Dependencies:**
   * Include dependencies for **Spring Web, Spring Data JPA, and H2 Database**.
3. **Create Application Properties:**
   * Configure database connection properties in **application.properties**.
4. **Define Entities and Repositories:**
   * Create **Book** entity and **BookRepository** interface.
5. **Create a REST Controller:**
   * Create a **BookController** class to handle CRUD operations.
6. **Run the Application:**
   * Run the Spring Boot application and test the REST endpoints.

**CODE:**

Java File

package com.ex9;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

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

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

import org.springframework.http.ResponseEntity;

import java.util.List;

import java.util.Optional;

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

@SpringBootApplication

public class Main {

public static void main(String[] args) {

SpringApplication.run(Main.class, args);

}

}

@RestController

@RequestMapping("/books")

class BookController {

@Autowired

private BookRepository bookRepository;

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

@GetMapping("/{id}")

public ResponseEntity<Book> getBookById(@PathVariable Long id) {

Optional<Book> book = bookRepository.findById(id);

return book.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@PutMapping("/{id}")

public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book bookDetails) {

Optional<Book> book = bookRepository.findById(id);

if (book.isPresent()) {

Book updatedBook = book.get();

updatedBook.setTitle(bookDetails.getTitle());

updatedBook.setAuthor(bookDetails.getAuthor());

return ResponseEntity.ok(bookRepository.save(updatedBook));

} else {

return ResponseEntity.notFound().build();

}

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteBook(@PathVariable Long id) {

if (bookRepository.existsById(id)) {

bookRepository.deleteById(id);

return ResponseEntity.ok().build();

} else {

return ResponseEntity.notFound().build();

}

}

}

@Entity

class Book {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

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;

}

}

@Repository

interface BookRepository extends JpaRepository<Book, Long> {

}

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">

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.3.2</version>

<relativePath/>

<!-- lookup parent from repository -->

</parent>

<modelVersion>4.0.0</modelVersion>

<groupId>com.ex9</groupId>

<artifactId>exercise9</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>jakarta.persistence</groupId>

<artifactId>jakarta.persistence-api</artifactId>

</dependency>

</dependencies>

</project>