**EXERCISE - 1:**

Pom.xml

<dependencies>

<!-- Spring Core Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.27</version>

</dependency>

<!-- Spring Beans Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.3.27</version>

</dependency>

</dependencies>

applicationContext.xml:

<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 definition for BookRepository -->

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

<!-- Bean definition for BookService -->

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

<!-- Inject BookRepository into BookService -->

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

</bean>

</beans>

BookService.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void manageBooks() {

System.out.println("Managing books in the library...");

// Call a method on bookRepository to demonstrate dependency injection

bookRepository.saveBook();

}

}

BookRepository.java:

package com.library.repository;

public class BookRepository {

public void saveBook() {

System.out.println("Saving a book...");

}

}

LibraryManagementApplication.java

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) {

// Load Spring context from the XML configuration file

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

// Retrieve the BookService bean

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

// Call the method to manage books

bookService.manageBooks();

}

}

**Exercise 2: Implementing Dependency Injection**

applicationContext.xml

<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 definition for BookRepository -->

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

<!-- Bean definition for BookService -->

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

<!-- Inject BookRepository into BookService -->

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

</bean>

</beans>

BookService.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void manageBooks() {

System.out.println("Managing books in the library...");

// Call a method on bookRepository to demonstrate dependency injection

bookRepository.saveBook();

}

}

LibraryManagementApplication.java

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) {

// Load Spring context from the XML configuration file

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

// Retrieve the BookService bean

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

// Call the method to manage books

bookService.manageBooks();

}

}

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

Pom.xml

<dependencies>

<!-- Spring Core Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.27</version>

</dependency>

<!-- Spring Beans Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.3.27</version>

</dependency>

<!-- Spring AOP Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.27</version>

</dependency>

<!-- AspectJ Weaver Dependency -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.19</version>

</dependency>

</dependencies>

LoggingAspect.java

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class LoggingAspect {

@Around("execution(\* com.library.service.\*.\*(..))")

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object proceed = joinPoint.proceed();

long executionTime = System.currentTimeMillis() - start;

System.out.println(joinPoint.getSignature() + " executed in " + executionTime + "ms");

return proceed;

}

}

applicationContext.xml

<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">

<!-- Enable AspectJ support -->

<aop:aspectj-autoproxy/>

<!-- Register the LoggingAspect bean -->

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

<!-- Other beans configuration (BookService, BookRepository) -->

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

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

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

</bean>

</beans>

LibraryManagementApplication.java

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) {

// Load Spring context from the XML configuration file

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

// Retrieve the BookService bean

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

// Call the method to manage books

bookService.manageBooks();

}

}

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

Pom.xml

<dependencies>

<!-- Spring Context Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.27</version>

</dependency>

<!-- Spring AOP Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.27</version>

</dependency>

<!-- Spring WebMVC Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.27</version>

</dependency>

<!-- AspectJ Weaver Dependency -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.19</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<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>

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

applicationContext.xml

<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 definition for BookRepository -->

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

<!-- Bean definition for BookService -->

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

<!-- Inject BookRepository into BookService -->

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

</bean>

</beans>

BookRepository.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void manageBooks() {

System.out.println("Managing books in the library...");

// Example usage of BookRepository

bookRepository.saveBook();

}

}

LibraryManagementApplication.java

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) {

// Load Spring context from the XML configuration file

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

// Retrieve the BookService bean

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

// Call the method to manage books

bookService.manageBooks();

}

}

**Exercise 6: Configuring Beans with Annotations**

applicationContext.xml

<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.library"/>

</beans>

BookService.java

package com.library.service;

import com.library.repository.BookRepository;

import org.springframework.stereotype.Service;

@Service

public class BookService {

private BookRepository bookRepository;

// Setter method for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void manageBooks() {

System.out.println("Managing books in the library...");

// Example usage of BookRepository

bookRepository.saveBook();

}

}

BookRepository.java

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

public void saveBook() {

System.out.println("Saving a book...");

}

}

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

applicationContext.xml

<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 -->

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

<!-- Define BookRepository bean (if not using annotations) -->

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

<!-- Define BookService bean with constructor injection -->

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

<constructor-arg ref="bookRepository"/>

</bean>

</beans>

BookService.java

package com.library.service;

import com.library.repository.BookRepository;

import org.springframework.stereotype.Service;

@Service

public class BookService {

private BookRepository bookRepository;

// Constructor for constructor injection

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

// Setter method for setter injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void manageBooks() {

System.out.println("Managing books in the library...");

bookRepository.saveBook();

}

}

LibraryManagementApplication

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) {

// Load Spring context from the XML configuration file

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

// Retrieve the BookService bean

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

// Call the method to manage books

bookService.manageBooks();

}

}

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

LoggingAspect

package com.library.aspect;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.springframework.stereotype.Component;

@Aspect

@Component

public class LoggingAspect {

// Advice method to log before method execution

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

public void logBefore() {

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

}

// Advice method to log after method execution

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

public void logAfter() {

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

}

}

applicationContext.xml

<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 -->

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

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

<aop:aspectj-autoproxy/>

<!-- Register the LoggingAspect bean (optional, since it's annotated with @Component) -->

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

</beans>

LibraryManagementApplication

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) {

// Load Spring context from the XML configuration file

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

// Retrieve the BookService bean

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

// Call the method to manage books

bookService.manageBooks();

}

}

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

Pom.xml

<dependencies>

<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>

</dependencies>