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

**1. Set Up a Spring Project**

* **pom.xml:**

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>5.3.29</version>

</dependency>

</dependencies>

</project>

**2. Configure the Application Context**

* **applicationContext.xml:**

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 id="bookService" class="com.library.service.BookService"/>

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

</beans>

**3. Define Service and Repository Classes**

* **BookService.java:**

java

package com.library.service;

public class BookService {

// Service methods

}

* **BookRepository.java:**

java

package com.library.repository;

public class BookRepository {

// Repository methods

}

**4. Run the Application**

* **LibraryManagementApplication.java:**

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) {

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

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

System.out.println("BookService Bean: " + bookService);

}

}

**Exercise 2: Implementing Dependency Injection**

**1. Modify the XML Configuration**

* **Updated applicationContext.xml:**

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

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

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

</bean>

</beans>

**2. Update the BookService Class**

* **BookService.java:**

java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

// Service methods

}

**3. Test the Configuration**

* **Run the Main Class:**
  + The LibraryManagementApplication remains the same as in Exercise 1.

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

**1. Add Spring AOP Dependency**

* **Updated pom.xml:**

xml

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.29</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.29</version>

</dependency>

</dependencies>

**2. Create an Aspect for Logging**

* **LoggingAspect.java:**

java

package com.library.aspect;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

@Aspect

public class LoggingAspect {

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

public void logBefore() {

System.out.println("Method execution start time: " + System.currentTimeMillis());

}

}

**3. Enable AspectJ Support**

* **Updated applicationContext.xml:**

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">

<aop:aspectj-autoproxy/>

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

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

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

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

</bean>

</beans>

**4. Test the Aspect**

* **Run the Main Class:**
  + The LibraryManagementApplication remains the same as in Exercise 1.

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

**1. Create a New Maven Project**

* **pom.xml:**
  + Similar to the previous pom.xml files.

**2. Add Spring Dependencies in pom.xml**

* **Updated pom.xml:**

xml

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.29</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.29</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.29</version>

</dependency>

</dependencies>

**3. Configure Maven Plugins**

* **Maven Compiler Plugin in pom.xml:**

xml

<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>

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

* **applicationContext.xml:**
  + Same as in Exercise 2.
* **BookService.java:**
  + Same as in Exercise 2.
* **LibraryManagementApplication.java:**
  + Same as in Exercise 1.

**Exercise 6: Configuring Beans with Annotations**

**1. Enable Component Scanning**

* **Updated applicationContext.xml:**

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">

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

<aop:aspectj-autoproxy/>

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

</beans>

**2. Annotate Classes**

* **BookService.java:**

java

package com.library.service;

import com.library.repository.BookRepository;

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

import org.springframework.stereotype.Service;

@Service

public class BookService {

private BookRepository bookRepository;

@Autowired

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

// Service methods

}

* **BookRepository.java:**

java

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

// Repository methods

}

**3. Test the Configuration**

* **Run the Main Class:**
  + The LibraryManagementApplication remains the same as in Exercise 1.

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

**1. Configure Constructor Injection**

* **Updated applicationContext.xml:**

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">

<aop:aspectj-autoproxy/>

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

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

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

<constructor-arg ref="bookRepository"/>

</bean>

</beans>

**2. Configure Setter Injection**

* **BookService.java:**
  + If needed, the setter method from previous examples will still be in use.

**3. Test the Injection**

* **Run the Main Class:**
  + The LibraryManagementApplication remains the same as in Exercise 1.

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

* **LoggingAspect.java:**
  + Same as in Exercise 3.
* **applicationContext.xml:**
  + Same as in Exercise 3.
* **Run the Main Class:**
  + The LibraryManagementApplication remains the same as in Exercise 1.

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

**1. Create a Spring Boot Project**

* Use **Spring Initializr** to create a new Spring Boot project named LibraryManagement with the following dependencies:
  + Spring Web
  + Spring Data JPA
  + H2 Database

**2. Add Dependencies**

* **pom.xml:**
  + This will be generated automatically by Spring Initializr.

**3. Create Application Properties**

* **application.properties:**

properties

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

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

spring.h2.console.enabled=true

**4. Define Entities and Repositories**

* **Book.java:**

java

package com.library.entity;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

// Getters and Setters

}

* **BookRepository.java:**

java

package com.library.repository;

import com.library.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**5. Create a REST Controller**

* **BookController.java:**

java

package com.library.controller;

import com.library.entity.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;

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookRepository.findById(id).orElse(null);

}

@PutMapping("/{id}")

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

Book book = bookRepository.findById(id).orElse(null);

if (book != null) {

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

return bookRepository.save(book);

}

return null;

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

}

}

**6. Run the Application**

* **LibraryManagementApplication.java:**

java

package com.library;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryManagementApplication {

public static void main(String[] args) {

SpringApplication.run(LibraryManagementApplication.class, args);

}

}