**Week2 Module1**

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

• Adding Dependencies

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>6.1.11</version>

</dependency>

</dependencies>

• Creating 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">

<!-- Define the BookRepository bean -->

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

<!-- Define the BookService bean -->

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

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

</bean>

</beans>

• Create Service Class

package com.library.service;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

}

• Create Repository Classes

package com.library.repository;

public class BookRepository {

}

• Running The Application:

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class Main {

public static void main(String[] args) {

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

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

}

}

**Exercise 2: Implementing Dependency Injection**

Exercise 2:

• Updating 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="com.librarymanagement.repository.BookRepository"/>

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

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

</bean>

</beans>

• Update BookService.java

package com.librarymanagement.service;

import com.librarymanagement.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performService() {

System.out.println("Performing service using BookRepository");

}

}

• Run The Application

package com.librarymanagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.librarymanagement.service.BookService;

public class Main {

public static void main(String[] args) {

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

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

bookService.performService();

}

}

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

Exercise 3:

• Add Spring AOP Dependency

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>6.1.11</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.1.11</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.9.22.1</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.22.1</version> <!-- Check for the latest version -->

</dependency>

</dependencies>

• Create an Aspect for Logging

package com.librarymanagement.aspect;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.After;

import org.springframework.stereotype.Component;

@Aspect

@Component

public class LoggingAspect {

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

public void logBefore(JoinPoint joinPoint) {

System.out.println("Starting method: " + joinPoint.getSignature().getName());

}

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

public void logAfter(JoinPoint joinPoint) {

System.out.println("Finished method: " + joinPoint.getSignature().getName());

}

}

• Enable AspectJ Support

<?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">

<aop:aspectj-autoproxy/>

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

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

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

</bean>

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

</beans>

• Run The Application

package com.librarymanagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.librarymanagement.service.BookService;

public class Main {

public static void main(String[] args) {

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

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

bookService.performService();

}

}

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

Exercise 4:

• Add Spring AOP Dependency

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>6.1.11</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>6.1.11</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>6.1.11</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>6.1.11</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.9.22.1</version> <!-- Check for the latest version -->

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.22.1</version> <!-- Check for the latest version -->

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<plugin>

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

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

<version>3.13.0. </version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

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

Exercise 5:

• Create Spring Configuration 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">

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

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

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

</bean>

</beans>

• Update the BookService Class

package com.librarymanagement.service;

import com.librarymanagement.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performService() {

System.out.println("Performing service using BookRepository");

}

}

• Run Application:

package com.librarymanagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.librarymanagement.service.BookService;

public class Main {

public static void main(String[] args) {

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

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

bookService.performService();

}

}

**Exercise 6: Configuring Beans with Annotations**

Exercise 6:

• Enable Component Scanning

<?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 specified package -->

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

</beans>

• Annotate Classes

o Service class:

package com.librarymanagement.service;

import org.springframework.stereotype.Service;

import com.librarymanagement.repository.BookRepository;

@Service

public class BookService {

private final BookRepository bookRepository;

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performService() {

System.out.println("Performing service using BookRepository");

}

}

o Repository Class

package com.librarymanagement.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

}

• Run Application:

package com.librarymanagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.librarymanagement.service.BookService;

public class Main {

public static void main(String[] args) {

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

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

bookService.performService();

}

}

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

Exercise 7:

• Configure Constructor Injection

<?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">

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

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

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

<constructor-arg ref="bookRepository"/>

</bean>

</beans>

• Configure Setter Injection

package com.librarymanagement.service;

import com.librarymanagement.repository.BookRepository;

import org.springframework.stereotype.Service;

@Service

public class BookService {

private BookRepository bookRepository;

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public BookService() {}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performService() {

System.out.println("Performing service using BookRepository");

}

}

• Run The Application

package com.librarymanagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.librarymanagement.service.BookService;

public class Main {

public static void main(String[] args) {

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

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

bookService.performService();

}

}

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

Exercise 8:

• Define an Aspect

import org.aspectj.lang.JoinPoint;

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 {

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

public void logBefore(JoinPoint joinPoint) {

System.out.println("Before executing method: " + joinPoint.getSignature().getName());

}

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

public void logAfter(JoinPoint joinPoint) {

System.out.println("After executing method: " + joinPoint.getSignature().getName());

}

}

• Create Advice Methods

• Configure the Aspect

<?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">

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

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

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

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

<constructor-arg ref="bookRepository"/>

</bean>

<aop:aspectj-autoproxy/>

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

</beans>

• Run The Application

package com.librarymanagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.librarymanagement.service.BookService;

public class Main {

public static void main(String[] args) {

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

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

bookService.performService();

}

}

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

Exercise 9:

• Add Dependencies

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

• Create Aplication Propeties

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

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

• Define Entities and Repositories:

package com.librarymanagement.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;

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

package com.librarymanagement.repository;

import com.librarymanagement.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

• Create Rest Controller

package com.librarymanagement.controller;

import com.librarymanagement.entity.Book;

import com.librarymanagement.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();

}

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

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

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@PutMapping("/{id}")

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

book.setId(id);

return bookRepository.save(book);

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

}

}

• Run The Application

package com.librarymanagement;

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);

}

}