Exercise 1: Configuring a Basic Spring Application

<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>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.22</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.22</version>

</dependency>

<!-- Spring Beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.3.22</version>

</dependency>

</dependencies>

</project>

<?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 definitions -->

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

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

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

</bean>

</beans>

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Getter and Setter for bookRepository

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performSomeService() {

System.out.println("Performing service with book repository");

bookRepository.doSomething();

}

}

package com.library.repository;

public class BookRepository {

public void doSomething() {

System.out.println("BookRepository is doing something");

}

}

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

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

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

bookService.performSomeService();

}

}

Exercise 2: Implementing Dependency 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"

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

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

<!-- Bean definitions -->

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

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

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

</bean>

</beans>

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for bookRepository

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performSomeService() {

System.out.println("Performing service with book repository");

bookRepository.doSomething();

}

}

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 XML configuration

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

// Retrieve the BookService bean from the context

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

// Call the service method to test dependency injection

bookService.performSomeService();

}

}

Exercise 3: Implementing Logging with Spring AOP

<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>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.22</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.22</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.22</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.9</version>

</dependency>

</dependencies>

</project>

package com.library.aspect;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Pointcut;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Component;

@Aspect

@Component

public class LoggingAspect {

private static final Logger logger = LoggerFactory.getLogger(LoggingAspect.class);

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

public void serviceMethods() {

// Pointcut for service methods

}

@Around("serviceMethods()")

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long startTime = System.currentTimeMillis();

Object proceed = joinPoint.proceed();

long executionTime = System.currentTimeMillis() - startTime;

logger.info(joinPoint.getSignature() + " executed in " + executionTime + "ms");

return proceed;

}

}

<?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">

<!-- Bean definitions -->

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

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

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

</bean>

<!-- Enable AspectJ auto proxy -->

<aop:aspectj-autoproxy />

<!-- Register the LoggingAspect -->

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

</beans>

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 XML configuration

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

// Retrieve the BookService bean from the context

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

// Call the service method to test logging aspect

bookService.performSomeService();

}

}

Exercise 4: Creating and Configuring a Maven Project

mvn archetype:generate -DgroupId=com.library -DartifactId=LibraryManagement -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

<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>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.22</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.22</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.22</version>

</dependency>

<!-- SLF4J API for logging (required by Spring AOP) -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.36</version>

</dependency>

<!-- AspectJ Weaver for AOP support -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.9</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>

</project>

<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

<?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 and inject the BookRepository -->

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

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

</bean>

</beans>

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for bookRepository

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performSomeService() {

System.out.println("Performing service with book repository");

bookRepository.doSomething();

}

}

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 XML configuration

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

// Retrieve the BookService bean from the context

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

// Call the service method to test the configuration

bookService.performSomeService();

}

}

Exercise 6: Configuring Beans with Annotations

<?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 -->

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

</beans>

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 final BookRepository bookRepository;

@Autowired

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performSomeService() {

System.out.println("Performing service with book repository");

bookRepository.doSomething();

}

}

package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

public void doSomething() {

System.out.println("Repository method called");

}

}

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 XML configuration

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

// Retrieve the BookService bean from the context

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

// Call the service method to test the configuration

bookService.performSomeService();

}

}

Exercise 7: Implementing Constructor and Setter 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">

<!-- Enable component scanning -->

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

<!-- Define the BookRepository bean -->

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

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

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

<constructor-arg ref="bookRepository" />

</bean>

</beans>

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;

// Constructor injection

@Autowired

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

// Setter injection

@Autowired

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void performSomeService() {

System.out.println("Performing service with book repository");

bookRepository.doSomething();

}

}

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 XML configuration

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

// Retrieve the BookService bean from the context

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

// Call the service method to test the configuration

bookService.performSomeService();

}

}

Exercise 8: Implementing Basic AOP with Spring

package com.library.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.library.service.BookService.\*(..))")

public void logBeforeMethod(JoinPoint joinPoint) {

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

}

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

public void logAfterMethod(JoinPoint joinPoint) {

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

}

}

<?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 -->

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

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

<aop:aspectj-autoproxy />

<!-- Define the BookRepository bean -->

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

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

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

<constructor-arg ref="bookRepository" />

</bean>

<!-- Define the LoggingAspect bean -->

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

</beans>

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 XML configuration

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

// Retrieve the BookService bean from the context

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

// Call the service method to test AOP functionality

bookService.performSomeService();

}

}

Exercise 9: Creating a Spring Boot Application

<dependencies>

<!-- Spring Boot Starter Web for REST controllers -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Starter Data JPA for ORM and repositories -->

<dependency>

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

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

</dependency>

<!-- H2 Database for in-memory database during development -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter Test for unit and integration tests -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

# H2 Database configuration

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

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

spring.datasource.username=sa

spring.datasource.password=

# JPA Configuration

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

# H2 Console configuration (optional for development)

spring.h2.console.enabled=true

package com.library.model;

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

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;

}

}

package com.library.repository;

import com.library.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

package com.library.controller;

import com.library.model.Book;

import com.library.repository.BookRepository;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/books")

public 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 book) {

if (!bookRepository.existsById(id)) {

return ResponseEntity.notFound().build();

}

book.setId(id);

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

}

@DeleteMapping("/{id}")

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

if (!bookRepository.existsById(id)) {

return ResponseEntity.notFound().build();

}

bookRepository.deleteById(id);

return ResponseEntity.noContent().build();

}

}

mvn spring-boot:run