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

**1. Create Maven Project**

**Project Name**: LibraryManagement

pom.xml:

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.22</version>

</dependency>

</dependencies>

</project>

**2. Create applicationContext.xml**

**Path**: src/main/resources/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 id="bookRepository" class="com.library.repository.BookRepository"/>

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

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

</bean>

</beans>

**3. Create Java Classes**

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void save() {

System.out.println("Saving book to the database...");

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook() {

System.out.println("Adding book...");

bookRepository.save();

}

}

**4. Create Main Class**

**LibraryManagementApplication.java**

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

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

bookService.addBook();

}

}

**Exercise 2: Implementing Dependency Injection**

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

**Exercise 3: Logging with Spring AOP**

**1. Add Spring AOP to pom.xml**

<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>

**2. LoggingAspect.java**

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

public class LoggingAspect {

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object obj = joinPoint.proceed();

long end = System.currentTimeMillis();

System.out.println("Execution time: " + (end - start) + "ms");

return obj;

}

}

**3. Update applicationContext.xml**

<aop:aspectj-autoproxy/>

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

<aop:config>

<aop:aspect ref="loggingAspect">

<aop:around method="logExecutionTime" pointcut="execution(\* com.library.service.\*.\*(..))"/>

</aop:aspect>

</aop:config>

Also include AOP namespace:

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

xsi:schemaLocation="... http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd"

**Exercise 4: Maven Project Setup**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.22</version>

</dependency>

<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 6: Configuring Beans with Annotations**

**1. Enable Component Scanning in applicationContext.xml**

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

Include context namespace:

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

xsi:schemaLocation="... http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"

**2. Add Annotations**

**BookRepository.java**

@Repository

public class BookRepository {

public void save() {

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

}

}

**BookService.java**

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

public void addBook() {

bookRepository.save();

}

}

**Exercise 7: Constructor & Setter Injection**

**In BookService.java:**

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

Update XML:

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

<constructor-arg ref="bookRepository"/>

</bean>

Also include setter if needed:

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

**Exercise 8: Basic AOP**

Covered in Exercise 3. You can expand it with:

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

public void logBefore(JoinPoint joinPoint) {

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

}

**Exercise 9: Spring Boot Version**

**1. Use Spring Initializr**

**Project Name**: LibraryManagement

**Dependencies**: Spring Web, Spring Data JPA, H2 Database

**2. application.properties**

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

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

spring.datasource.username=sa

spring.datasource.password=

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

spring.h2.console.enabled=true

**3. Book Entity and Repository**

@Entity

public class Book {

@Id

@GeneratedValue

private Long id;

private String title;

}

@Repository

public interface BookRepository extends JpaRepository<Book, Long> {}

**4. REST Controller**

@RestController

@RequestMapping("/books")

public class BookController {

@Autowired

private BookRepository bookRepository;

@PostMapping

public Book create(@RequestBody Book book) {

return bookRepository.save(book);

}

@GetMapping

public List<Book> getAll() {

return bookRepository.findAll();

}

}

**5. Main Class**

@SpringBootApplication

public class LibraryManagementApplication {

public static void main(String[] args) {

SpringApplication.run(LibraryManagementApplication.class, args);

}

}