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

**Set Up a Spring Project:**

* Create a Maven project named **LibraryManagement**.
* In pom.xml, add the Spring Core dependency:

xml

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version> <!-- Use recent stable version -->

</dependency>

**Define Spring Beans – applicationContext.xml**

* Create src/main/resources/applicationContext.xml.
* Define beans:

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="bookDAO" class="com.book.inventory.dao.BookDAO"/>

<bean id="bookManager" class="com.book.inventory.service.BookManager"/>

</beans>

**Create Java Classes**

// BookDAO.java

package com.book.inventory.dao;

public class BookDAO {

// Handles DB logic (simulated here)

}

// BookManager.java

package com.book.inventory.service;

public class BookManager {

// Business logic here

}

**Load Context and Run**

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.book.inventory.service.BookManager;

public class BookInventoryApp {

public static void main(String[] args) {

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

BookManager manager = (BookManager) context.getBean("bookManager");

// manager.doSomething();

}

}  
   
**Exercise 2: Implementing Dependency Injection**

**Modify applicationContext.xml for DI**

<bean id="bookDAO" class="com.book.inventory.dao.BookDAO" />

<bean id="bookManager" class="com.book.inventory.service.BookManager">

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

</bean>

**Update the BookService Class:**

public class BookManager {

private BookDAO bookDAO;

public void setBookDAO(BookDAO bookDAO) {

this.bookDAO = bookDAO;

}

}

**Test the Configuration:**

* Run BookInventoryApp to verify DI works (no exceptions, beans wired).

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

**Add Spring AOP Dependency:**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.30</version>

</dependency>

**Create an Aspect for Logging:**

package com.book.inventory.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class PerformanceLogger {

@Around("execution(\* com.book.inventory.service.\*.\*(..))")

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long begin = System.currentTimeMillis();

Object result = joinPoint.proceed();

long duration = System.currentTimeMillis() - begin;

System.out.println(joinPoint.getSignature() + " took " + duration + "ms");

return result;

}

}

<beans ...>

<bean id="performanceLogger" class="com.book.inventory.aspect.PerformanceLogger"/>

<aop:aspectj-autoproxy xmlns:aop="http://www.springframework.org/schema/aop"/>

</beans>

**Test the Aspect:**

* Run the main class and observe logs for execution times.

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

**Create a New Maven Project:**

* Project name: BookInventory

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

xml

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

**Configure Maven Plugins:**

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 9: Creating a Spring Boot Application**

**Create a Spring Boot Project:**

Project name: BookInventoryBoot

Dependencies: Spring Web, Spring Data JPA, H2

**Create Application Properties:**

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

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

**Define Entities and Repositories:**

@Entity

public class Book {

@Id @GeneratedValue

private Long id;

private String title;

// Getters and setters

}

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

**Create a REST Controller:**

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookRepo repo;

@GetMapping

public List<Book> fetchAll() {

return repo.findAll();

}

// Add more CRUD if needed

}

**Run the Application:**

* Run BookInventoryBootApplication.java and test endpoints

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

**Update Aspect with @Before and @After**

@Before("execution(\* com.book.inventory.service.\*.\*(..))")

public void beforeAdvice() {

System.out.println(">>> Entering method...");

}

@After("execution(\* com.book.inventory.service.\*.\*(..))")

public void afterAdvice() {

System.out.println("<<< Exiting method...");

}

**Register and Enable AOP**

Already done in applicationContext.xml during Exercise 3.

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

**Configure Constructor Injection:**

<bean id="bookManager" class="com.book.inventory.service.BookManager">

<constructor-arg ref="bookDAO"/>

</bean>

public BookManager(BookDAO dao) {

this.bookDAO = dao;

}   
   
**Exercise 6: Configuring Beans with Annotations**

**Enable Component Scanning:**

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

**Annotate Classes:**

@Service

public class BookManager { ... }

@Repository

public class BookDAO { ... }

**Test the Configuration:**

* Run the main class, ensure beans are auto-wired.