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

**1. pom.xml** (Add Spring Core Dependency):

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.29</version>

</dependency>

</dependencies>

**2. 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. 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 performService() {

bookRepository.getBook();

}

}

**4. BookRepository.java**:

package com.library.repository;

public class BookRepository {

public void getBook() {

System.out.println("Fetching book from repository.");

}

}

**5. Main Class**:

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

service.performService();

}

}

**Exercise 2: Implementing Dependency Injection**

**Step 1: Modify the XML Configuration**

<!-- File: 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">

<!-- Repository Bean -->

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

<!-- Service Bean with DI -->

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

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

</bean>

</beans>

**Step 2: Update BookService Class**

// File: com/library/service/BookService.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for DI

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBook() {

bookRepository.getBook();

}

}

**Step 3: Create BookRepository Class**

// File: com/library/repository/BookRepository.java

package com.library.repository;

public class BookRepository {

public void getBook() {

System.out.println("Getting book from the repository...");

}

}

**Step 4: Test the Configuration (Main Class)**

// File: 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 service = (BookService) context.getBean("bookService");

service.displayBook(); // Should print: Getting book from the repository...

}

}

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

**1. Add Spring AOP Dependency**:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.29</version>

</dependency>

**2. LoggingAspect.java**:

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class LoggingAspect {

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

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object result = joinPoint.proceed();

long end = System.currentTimeMillis();

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

return result;

}

}

**3. applicationContext.xml** (enable AOP):

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

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

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

<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>

<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**

**Step 1: Create Spring Configuration File – applicationContext.xml**

<!-- File: 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">

<!-- Define Repository Bean -->

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

<!-- Define Service Bean and Inject Repository using Setter Injection -->

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

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

</bean>

</beans>

**Step 2: Update BookService Class for Setter Injection**

// File: com/library/service/BookService.java

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void issueBook() {

bookRepository.getBook();

}

}

**Step 3: Create BookRepository Class**

// File: com/library/repository/BookRepository.java

package com.library.repository;

public class BookRepository {

public void getBook() {

System.out.println("Fetching book data from repository...");

}

}

**Step 4: Create Main Class to Load Spring Context**

// File: 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 service = (BookService) context.getBean("bookService");

service.issueBook(); // Expected Output: Fetching book data from repository...

}

}

**Exercise 6: Configuring Beans with Annotations**

**1. Enable Component Scanning**:

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

<context:annotation-config />

**2. Annotate Classes**:

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

public void performService() {

bookRepository.getBook();

}

}

@Repository

public class BookRepository {

public void getBook() {

System.out.println("Book retrieved.");

}

}

**Exercise 7: Constructor and Setter Injection**

**Constructor Injection in XML**:

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

<constructor-arg ref="bookRepository"/>

<property name="anotherDependency" ref="someOtherBean"/>

</bean>

**BookService.java**:

public class BookService {

private BookRepository bookRepository;

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void setAnotherDependency(OtherType other) {

// setter injection example

}

}

**Exercise 8: Basic AOP with Spring**

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

public void beforeMethod() {

System.out.println("Before method call...");

}

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

public void afterMethod() {

System.out.println("After method call...");

}

**Exercise 9: Spring Boot Application**

**Book.java**:

@Entity

public class Book {

@Id

@GeneratedValue

private Long id;

private String title;

}

**BookRepository.java**:

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

**BookController.java**:

@RestController

@RequestMapping("/books")

public class BookController {

@Autowired

private BookRepository repo;

@GetMapping

public List<Book> getAll() {

return repo.findAll();

}

@PostMapping

public Book save(@RequestBody Book book) {

return repo.save(book);

}

}

**application.properties**:

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

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update