Exercise 1: Configuring a Basic Spring Application

Create Maven Project:

Project Name: LibraryManagement

Use your IDE or mvn archetype:generate command.

Add Spring Core Dependency in pom.xml:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

Create applicationContext.xml in src/main/resources:

<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>

Create Classes:

- BookService in com.library.service

- BookRepository in com.library.repository

Main Class:

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

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

service.someMethod(); // test method

Exercise 2: Implementing Dependency Injection

Modify applicationContext.xml:

Ensure bookRepository is injected into bookService.

Add Setter in BookService:

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

Run the main class to test the injection.

Exercise 3: Implementing Logging with Spring AOP

Add AOP Dependency in pom.xml:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>5.3.30</version>

</dependency>

Create LoggingAspect in com.library.aspect:

@Aspect

public class LoggingAspect {

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

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.currentTimeMillis();

Object proceed = joinPoint.proceed();

long duration = System.currentTimeMillis() - start;

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

return proceed;

}

}

Enable AspectJ in applicationContext.xml:

<aop:aspectj-autoproxy/>

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

Run application and observe console logs.

Exercise 4: Creating and Configuring a Maven Project

Create Maven Project LibraryManagement.

Dependencies in pom.xml:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>5.3.30</version>

</dependency>

Maven Compiler Plugin:

<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

Create the Spring Configuration File

Create an XML configuration file named applicationContext.xml under the src/main/resources directory. This file will define the beans and how they are wired together

<?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 BookRepository using setter -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository" />
    </bean>
</beans>
```

Create the BookRepository Class

Located in com.library.repository package:

```
package com.library.repository;

public class BookRepository {
    public void saveBook() {
        System.out.println("BookRepository: Book saved.");
    }
}
```

Create the BookService Class with Setter Injection

Located in com.library.service package:

```
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 performAction() {
        System.out.println("BookService: Performing service operation.");
        bookRepository.saveBook(); // calling repository method
    }
}
```

Create the Main Class to Load Spring Context

This class will load the Spring IoC container and retrieve the 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) {

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

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



        service.performAction();
    }
}
```

Exercise 6: Configuring Beans with Annotations

Update applicationContext.xml:

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

<context:annotation-config/>

Add Annotations:

@Service on BookService

@Repository on BookRepository

Inject Dependency Using @Autowired:

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

}

Run main class to verify.

Exercise 7: Constructor and Setter Injection

Constructor Injection in applicationContext.xml:

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

<constructor-arg ref="bookRepository"/>

</bean>

Setter Injection (alternative):

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

Modify BookService:

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

Test the injection in the main class.

Exercise 8: Basic AOP with Spring

Aspect Class (LoggingAspect):

@Aspect

public class LoggingAspect {

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

public void beforeAdvice() {

System.out.println("Method execution started...");

}

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

public void afterAdvice() {

System.out.println("Method execution completed.");

}

}

Enable AOP in XML:

<aop:aspectj-autoproxy/>

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

Test by calling service methods.

Exercise 9: Spring Boot Application

Create Spring Boot Project:

Use Spring Initializr

Name: LibraryManagement

Add Dependencies:

- Spring Web

- Spring Data JPA

- H2 Database

application.properties:

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

spring.datasource.driverClassName=org.h2.Driver

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

spring.h2.console.enabled=true

Create Entity Book:

@Entity

public class Book {

@Id @GeneratedValue

private Long id;

private String title;

private String author;

}

Repository:

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

Controller:

@RestController

@RequestMapping("/books")

```
public class BookController {
@Autowired
private BookRepository repository;
@PostMapping
public Book addBook(@RequestBody Book book) {
return repository.save(book);
}
@GetMapping
public List<Book> getBooks() {
return repository.findAll();
}
}
```

Run Spring Boot App and test /books endpoints.