**Mandatory hands-on**

**Spring Core and Maven**

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

Scenario: Your company is developing a web application for managing a library.

You need to use the Spring Framework to handle the backend operations.

**Code:**

**applicationContext.xml**

<?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

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

<!-- Define BookRepository Bean -->

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

<!-- Define BookService Bean and inject BookRepository -->

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

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

</bean>

</beans>

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void saveBook(String title) {

System.out.println("Book '" + title + "' saved to the repository.");

}

}

**BookService.java**

package com.library.service;s

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 addBook(String title) {

System.out.println("Adding book: " + title);

bookRepository.saveBook(title);

}

}

**Main.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class Main {

public static void main(String[] args) {

ApplicationContext context =

new ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.addBook("Spring in Action");

}

}

**Output:**  
Adding book: Spring in Action

Book 'Spring in Action' saved to the repository.

**Exercise 2: Implementing Dependency Injection**

Scenario: In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**Code:**

**applicationContext.xml**

<?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

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

<!-- Define BookRepository Bean -->

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

<!-- Bean for BookService with dependency injection -->

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

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

</bean>

</beans>

**BookRepository.java**

package com.example.repository;

import java.util.Arrays;

import java.util.List;

public class BookRepository {

public List<String> findAll() {

return Arrays.asList(

"Book ID: 101 - The Alchemist by Paulo Coelho",

"Book ID: 102 - Clean Code by Robert C. Martin",

"Book ID: 103 - Effective Java by Joshua Bloch"

);

}

}

**BookService.java**

package com.example.service;

import com.example.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void listBooks() {

bookRepository.findAll().forEach(System.out::println);

}

}

**LibraryManagementApplication.java**

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.service.BookService;

public class LibraryManagementApplication {

public static void main(String[] args) {

// Load Spring configuration

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

// Retrieve the BookService bean

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

// Call method to test DI

bookService.listBooks();

}

}

**Output:**

Book ID: 101 - The Alchemist by Paulo Coelho

Book ID: 102 - Clean Code by Robert C. Martin

Book ID: 103 - Effective Java by Joshua Bloch

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

**Scenario:** You need to set up a new Maven project for the library management application and add Spring dependencies.

**Code:**

**pom.xml**

<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.example.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!-- Spring Core and Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

<!-- For Servlet API -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler Plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>