**Week 3**

**Mandatory Hands-On**

**Exercise 1**

**Code**

**Library App**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryApp {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.listAllBooks();  
 }  
}

**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   
 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>

**Book Repository**

package com.library.repository;  
  
import java.util.Arrays;  
import java.util.List;  
  
public class BookRepository {  
 public List<String> getBooks() {  
 return Arrays.*asList*("Harry Potter", "The Hobbit", "1984");  
 }  
}

**Output**

A screen shot of a computer

AI-generated content may be incorrect.

**Exercise 2**

**Code**

**BookRepository**

package com.library.repository;  
  
import java.util.Arrays;  
import java.util.List;  
  
public class BookRepository {  
 public List<String> getBooks() {  
 return Arrays.*asList*("Harry Potter", "The Hobbit", "1984");  
 }  
}

**Setter class is already present in the previous code. So no changes here.**

**Output**

A screen shot of a computer

AI-generated content may be incorrect.

**Exercise 4**

**Code**

**Updated pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Library Management</name>

<properties>

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

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler plugin configured for Java 1.8 -->

<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>

</project>

**Spring Data jpa**

**Quick Example**

Spring Data JPA is a part of the Spring ecosystem that simplifies database access using Java Persistence API (JPA).  
It eliminates boilerplate code like EntityManager and DAO implementations by using interfaces and annotations.

**Code Example**

import javax.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

// getters and setters

}

**Difference between JPA, Hibernate and Spring Data JPA**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification (Interface/API) | Implementation of JPA | Abstraction layer over JPA (built on top of JPA & Spring) |
| **Provided By** | Jakarta EE (formerly Java EE) | Red Hat | Spring Framework (Pivotal/VMware) |
| **Role** | Defines standard annotations & interfaces | Provides actual implementation + extra features | Simplifies data access using repositories |
| **Examples** | @Entity, @Id, EntityManager | Session, Criteria, HQL, caching, lazy loading | JpaRepository, @Query, method-based queries |
| **Boilerplate Code** | High | Medium | Very Low |
| **Custom Queries** | Requires JPQL or native SQL | HQL or Criteria API | Supports JPQL, native SQL, and derived query methods |
| **Ease of Use** | Moderate | Easier than JPA | Easiest (auto CRUD methods) |
| **Integration Needed** | Must be manually integrated with persistence | Integrates with JPA providers | Seamlessly integrates with Spring Boot |
| **Use Case** | Standard ORM layer for Java EE apps | When you need more control over JPA behavior | When building Spring apps that require fast and clean DB access |