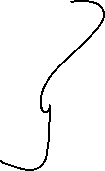
**WEEK-3**



**Exercise-1: Configuring a Basic Spring Application the Singleton Pattern**

**Exercise 2: Implementing Dependency Injection**



**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" (Both in same code)



         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</version>

    <dependencies>

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.34</version>

        </dependency>

    </dependencies>

</project>

**BookRepository.java**

package com.example;

public class BookRepository {

    public void saveBook(String name) {

        System.out.println("Saving book: " + name);

    }}

**BookService.java**

package com.example;

public class BookService {

    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String name) {

        bookRepository.saveBook(name);

    }

}

**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.example.BookRepository"/>

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

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

    </bean>

</beans>

**MainApp.java**

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

    public static void main(String[] args) {

        ApplicationContext context =

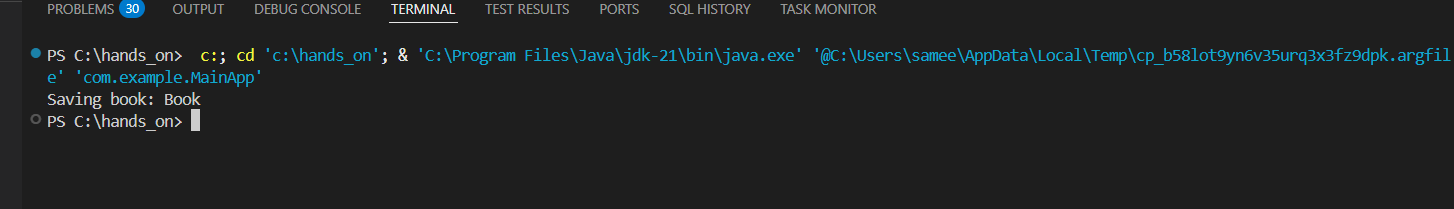
            new ClassPathXmlApplicationContext("applicationContext.xml");

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

        bookService.addBook("Book");

    }}

**OUTPUT:**



**Exercise 3:** **Creating and Configuring a Maven Project**

**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.library</groupId>

    <artifactId>LibraryManagement</artifactId>

    <version>1.0-SNAPSHOT</version>

    <properties>

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

    </properties>

    <dependencies>

        <!-- Spring Core + Context -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.34</version>

        </dependency>

        <!-- Spring AOP -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-aop</artifactId>

            <version>5.3.34</version>

        </dependency>

        <!-- Spring Web MVC -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-webmvc</artifactId>

            <version>5.3.34</version>

        </dependency>

        <dependency>

            <groupId>javax.servlet</groupId>

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

            <version>4.0.1</version>

            <scope>provided</scope>

        </dependency>

        <dependency>

    <groupId>org.slf4j</groupId>

    <artifactId>slf4j-api</artifactId>

    <version>1.7.36</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>

</project>

**LibraryManagementApplication.java**

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

    public static void main(String[] args) {

        ApplicationContext context =

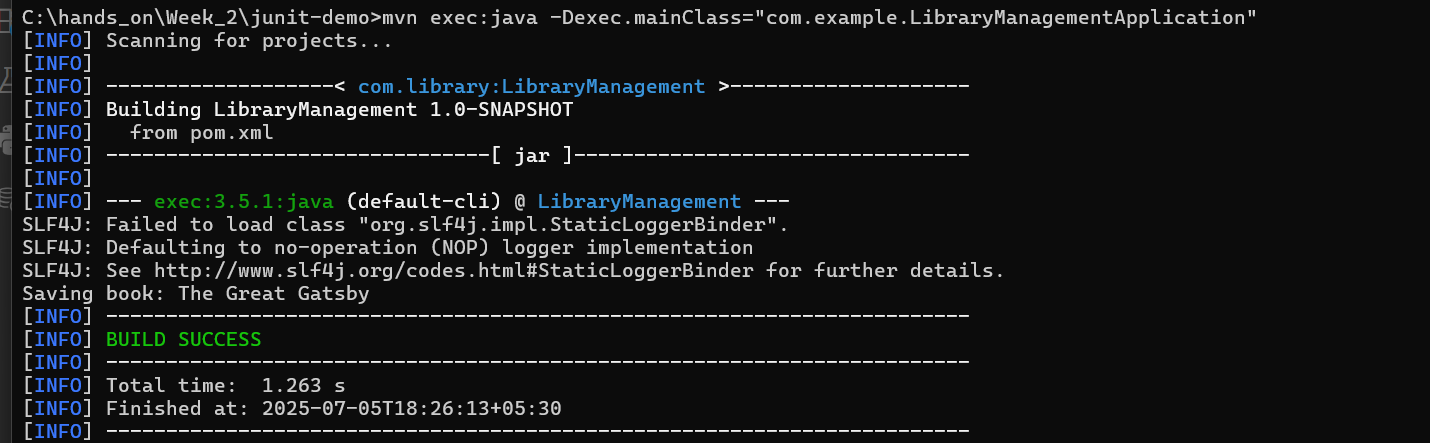
            new ClassPathXmlApplicationContext("applicationContext.xml");

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

        bookService.addBook("The Great Gatsby");

    }}

**OUTPUT:**

****

**Exercise 4:** **Spring Data JPA - Quick Example**

**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</groupId>

  <artifactId>spring-data-jpa-demo</artifactId>

  <version>1.0.0</version>

  <packaging>jar</packaging>

  <parent>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-parent</artifactId>

    <version>3.2.5</version>

  </parent>

  <dependencies>

    <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter-data-jpa</artifactId>

    </dependency>

    <dependency>

      <groupId>com.h2database</groupId>

      <artifactId>h2</artifactId>

      <scope>runtime</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <plugin>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-maven-plugin</artifactId>

      </plugin>

    </plugins>

  </build>

</project>

**Book.java**

package com.example.demo;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

public Book() {}

public Book(String title, String author) {

this.title = title;

this.author = author;

}}

**SpringDataJpaDemoApplication.java**

package com.example;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class SpringDataJpaDemoApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringDataJpaDemoApplication.class, args);

    }

    @Bean

    CommandLineRunner run(BookRepository repo) {

        return args -> {

            repo.save(new Book("The Alchemist", "Paulo Coelho"));

            repo.save(new Book("1984", "George Orwell"));

            repo.findAll().forEach(book ->

                System.out.println(book.getId() + ": " + book.getTitle()));

        };}}

**application.properties**

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

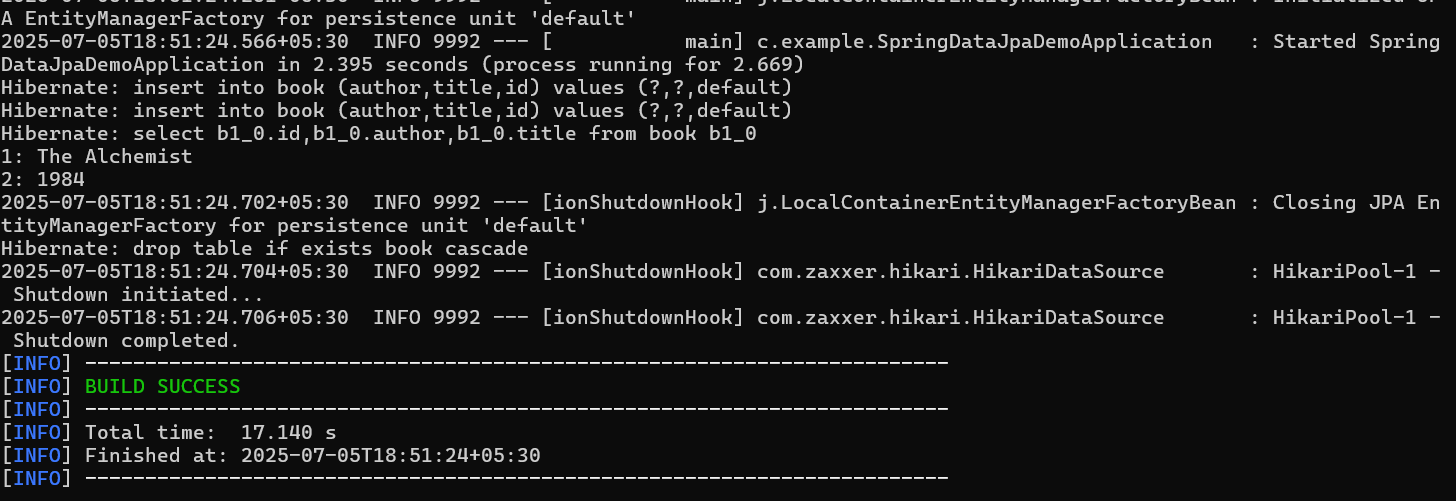
spring.datasource.password=

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

spring.h2.console.enabled=true

spring.jpa.show-sql=true

**OUTPUT:**



**Exercise 5: Difference between JPA, Hibernate and Spring Data JPA**

**1. JPA (Java Persistence API) – Specification / Interface**

* What it is: A Java specification for object-relational mapping (ORM).
* Think of it as: A set of rules or interfaces.
* Defined by: Java EE (Jakarta EE now).
* Key Interfaces: EntityManager, Query, EntityTransaction, etc.
* Does NOT provide implementation – needs a provider like Hibernate.

Analogy: JPA is like a driver's license test (the rules). It doesn’t give you a car to drive.

**2. Hibernate – JPA Implementation (and more)**

* What it is: A popular implementation of JPA.
* Also adds extra features beyond JPA (like @Filter, caching, custom strategies, etc.).
* Can be used with or without JPA.
* Manages database sessions, SQL generation, caching, lazy loading, etc.

Analogy: Hibernate is the car you use to follow the rules defined by JPA.

**3. Spring Data JPA – Spring’s Abstraction Layer on JPA**

* What it is: A Spring module that builds on JPA (usually using Hibernate underneath).
* Provides:
  + Ready-to-use Jpa Repository interfaces.
  + Auto-generated queries by method names.
  + Boilerplate-free code (no need to write DAOs).
  + Integrated with Spring Boot (easy configuration, transaction management, etc.).

Analogy: Spring Data JPA is like an automatic self-driving car built on top of Hibernate, which follows JPA rules.