**WEEK 3: SPRING CORE AND MAVEN**

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

**CODE:**

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 BookRepository bean -->

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

<!-- Define BookService bean -->

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

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

</bean>

</beans>

com.library.repository.BookRepository:

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for bookRepository (dependency injection)

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

}

com.library.repository.BookRepository:

package com.library.repository;

public class BookRepository {

public void displayBooks() {

System.out.println("Books displayed from the repository.");

}

}

com.library.Main:

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApp {

public static void main(String[] args) {

// Load Spring context from applicationContext.xml

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

// Retrieve the BookService bean from the Spring context

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

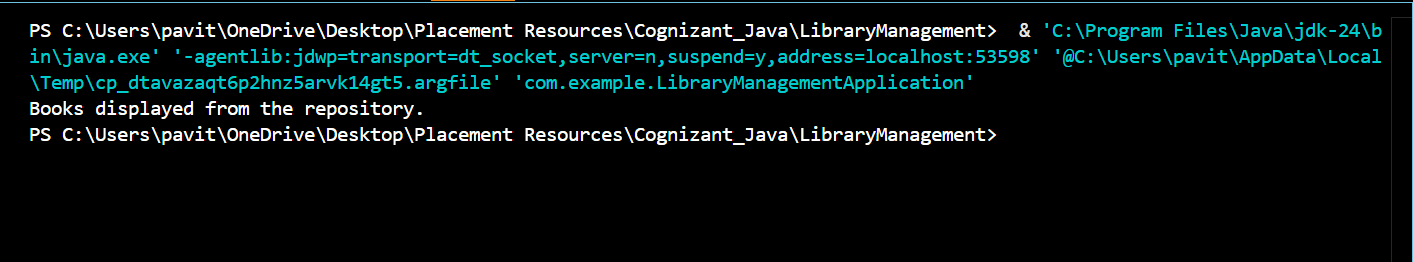
// Test your configuration

System.out.println("Library Management Application started successfully.");

}

}

**OUTPUT:**

**

**Exercise 2: Implementing Dependency Injection**

**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

http://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>

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;

}

// Example method to use the repository

public void displayBooks() {

System.out.println("Calling repository to list books:");

bookRepository.listBooks(); // Calls the BookRepository method

}

}

LibraryManagementApp.java:

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApp {

public static void main(String[] args) {

// Load the Spring configuration

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

// Get the BookService bean from the context

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

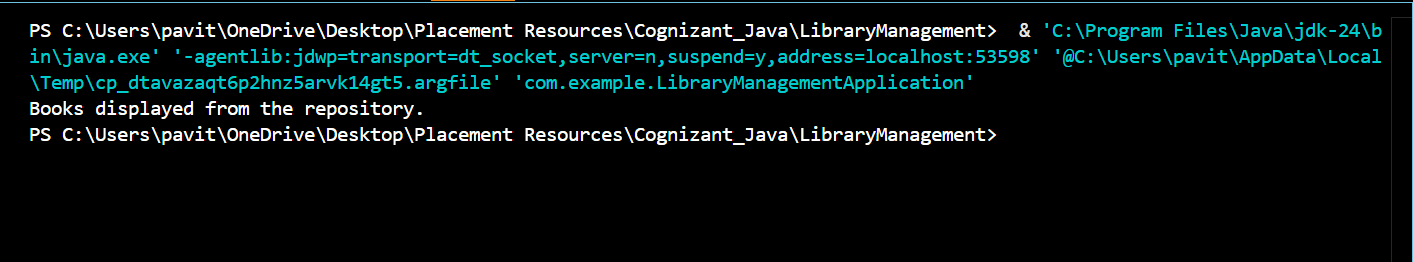
// Test the method

bookService.displayBooks();

}

}

**OUTPUT:**

**

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

**CODE:**

AppConfig.java:

package com.example.library.config;

import com.example.library.controller.LibraryController;

import com.example.library.repository.BookRepository;

import com.example.library.service.LibraryService;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

public BookRepository bookRepository() {

return new BookRepository();

}

@Bean

public LibraryService libraryService() {

return new LibraryService(bookRepository());

}

@Bean

public LibraryController libraryController() {

return new LibraryController(libraryService());

}

}

BookRepository.java:

package com.example.library.repository;

import java.util.Arrays;

import java.util.List;

public class BookRepository {

public List<String> getAllBooks() {

return Arrays.asList("The Great Gatsby", "1984", "To Kill a Mockingbird");

}

}

LibraryService.java:

package com.example.library.service;

import com.example.library.repository.BookRepository;

import java.util.List;

public class LibraryService {

private final BookRepository bookRepository;

public LibraryService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public List<String> fetchBooks() {

return bookRepository.getAllBooks();

}

}

LibraryController.java:

package com.example.library.controller;

import com.example.library.service.LibraryService;

import java.util.List;

public class LibraryController {

private final LibraryService libraryService;

public LibraryController(LibraryService libraryService) {

this.libraryService = libraryService;

}

public void displayBooks() {

List<String> books = libraryService.fetchBooks();

System.out.println("Available Books in the Library:");

books.forEach(System.out::println);

}

}

Main.java:

package com.example.library;

import com.example.library.config.AppConfig;

import com.example.library.controller.LibraryController;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {

public static void main(String[] args) {

AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

LibraryController controller = context.getBean(LibraryController.class);

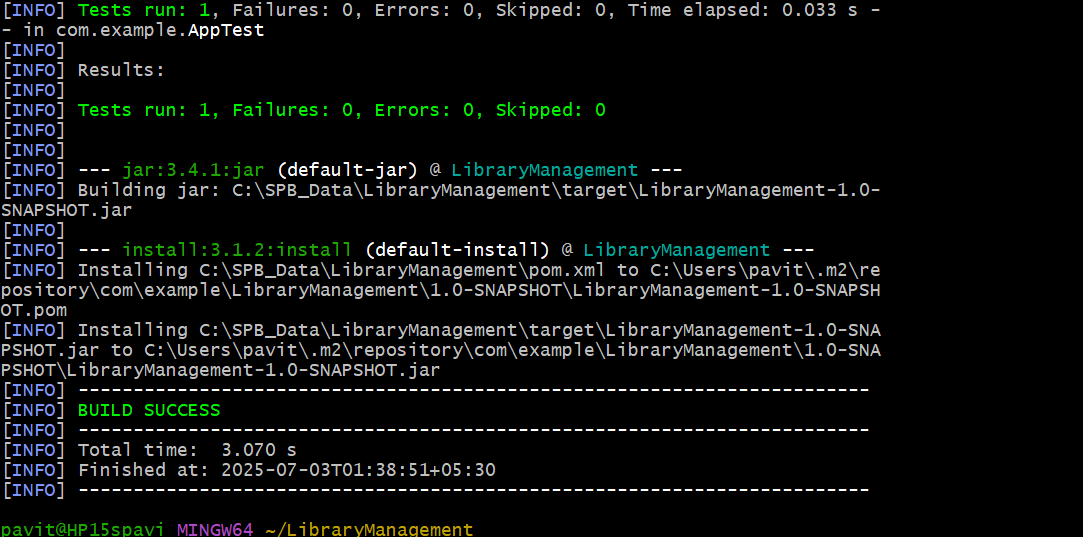
controller.displayBooks();

context.close();

}

}

**OUTPUT:**

****

**Handson 1: Spring Data JPA - Quick Example**

**CODE:**

application.properties:

# Logging Config

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# DB Configuration

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

pom.xml:

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

</dependencies>

model.Country:

package com.cognizant.ormlearn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="country")

public class Country {

@Id

@Column(name="code")

private String code;

@Column(name="name")

private String name;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country{" +

"code='" + code + '\'' +

", name='" + name + '\'' +

'}';

}

}

CountryRepository:

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

CountryService:

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

Main Class OrmLearnApplication.java:

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static CountryService countryService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

List<Country> countries = countryService.getAllCountries();

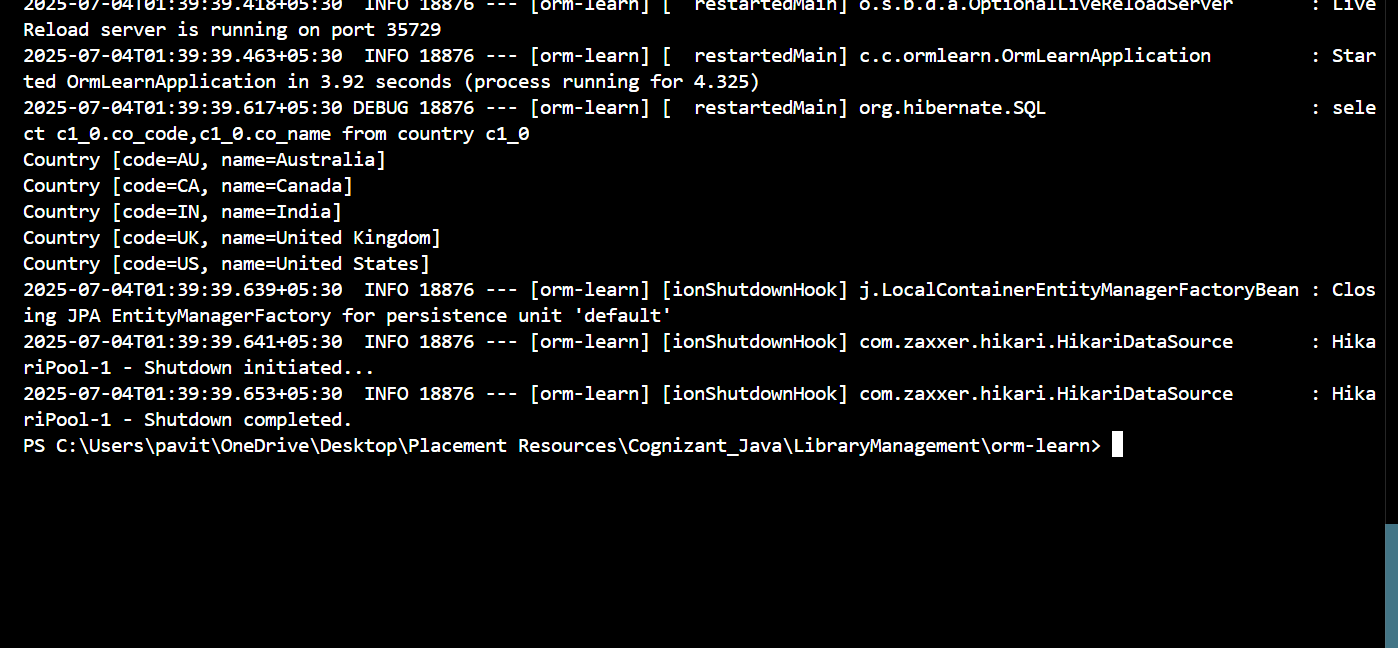
LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

}

**OUTPUT:**



**Handson 4: Difference between JPA, Hibernate and Spring Data JPA**

**CODE:**

**Java Persistence API (JPA):**

A Java specification (JSR 338) for managing relational data in Java applications.  
Defines a standard set of annotations and interfaces for object-relational mapping (ORM).

**Hibernate:**

A **popular ORM framework** and **the most widely used implementation of JPA**.  
Provides its own native APIs (**Session, Transaction, Query**) in addition to supporting the JPA interfaces.

**Spring Data JPA:**

A **Spring Framework project** that provides an **abstraction layer on top of JPA/Hibernate.**It **does not implement JPA itself** but uses **JPA providers like Hibernate** under the hood.

**Hibernate example:**

public Integer addEmployee(Employee employee){

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**Spring Data JPA Example:**

Repository:

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

Service Layer:

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}