**WEEK-3**

**SPRING CORE\_MAVEN**

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

**MY CODE:**

**MainApp.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

public static void main(String[] args) {

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

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

bookService.addBook("The Art of Spring");

((ClassPathXmlApplicationContext) context).close();

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

publiic void addBook(String title) {

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

bookRepository.saveBook(title);

}}

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void saveBook(String title) {

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

}

}

**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>

**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>

<packaging>jar</packaging>

<version>1.0-SNAPSHOT</version>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<dependencies>

<!-- Spring Core Context Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.library.MainApp</mainClass>

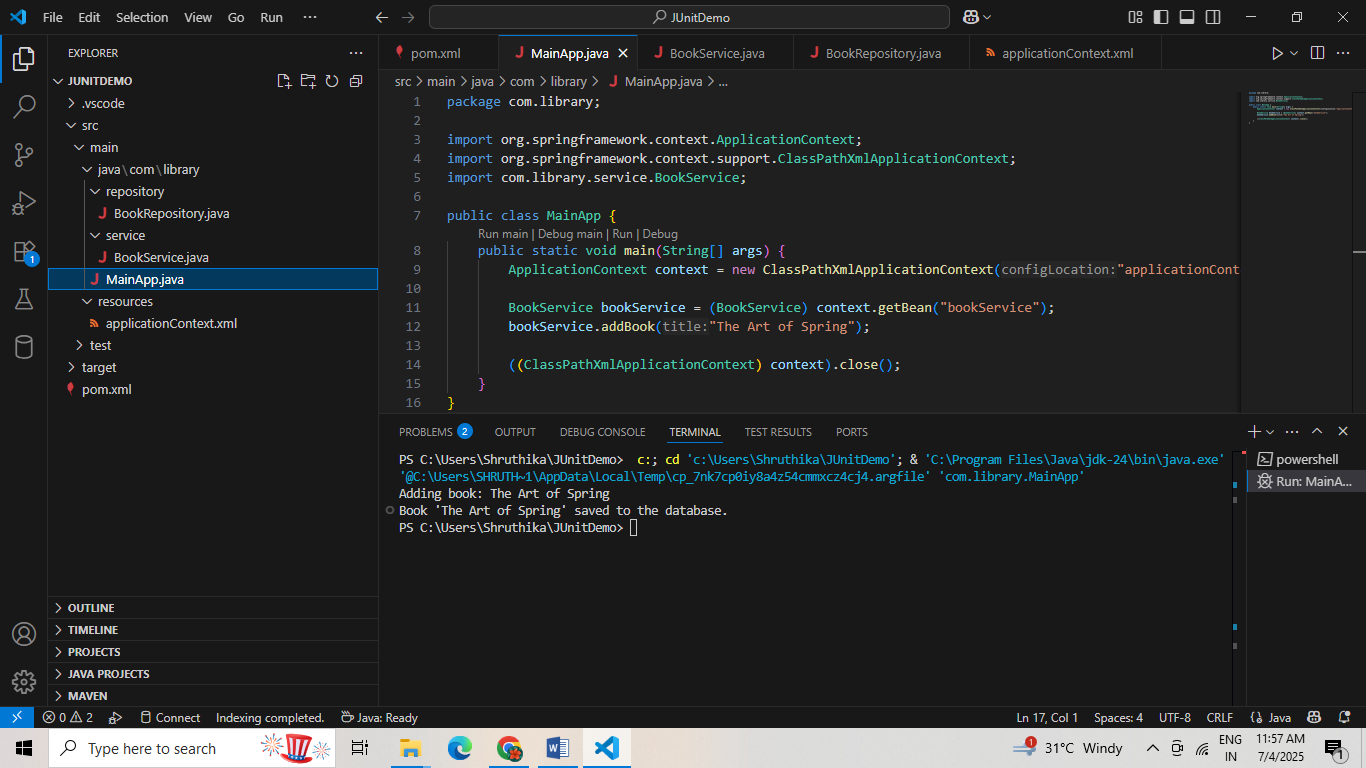
</configuration>

</plugin>

</plugins>

</build>

</project>

**OUTPUT: **

**Exercise 2: Implementing Dependency Injection**

**MY 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">

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

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

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

</bean>

</beans>

**LibraryManagementApplication.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

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

bookService.addBook("The Art of Spring");

((ClassPathXmlApplicationContext) context).close();

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

System.out.println("[DI] BookRepository has been injected into BookService.");

this.bookRepository = bookRepository;

}

public void addBook(String title) {

System.out.println("[DI ACTIVE] Adding book: " + title);

bookRepository.saveBook(title);

}

}

**BookRepository.java**

package com.library.repository;

public class BookRepository {

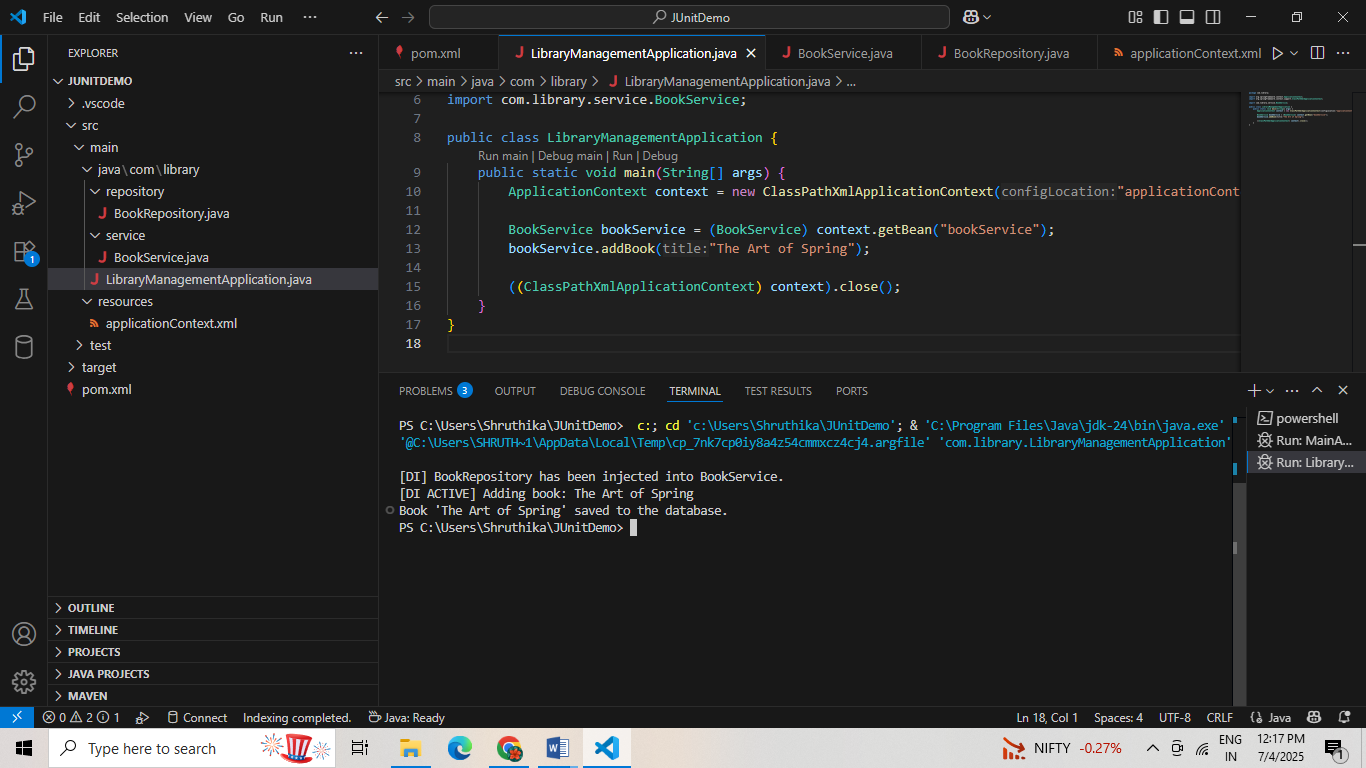
public void saveBook(String title) {

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

}

}

**OUTPUT:**

****

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

**MY CODE:**

<!-- FULL Spring-enabled 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>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Core & Beans -->

<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>

<!-- Java Compiler for 1.8 -->

<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>

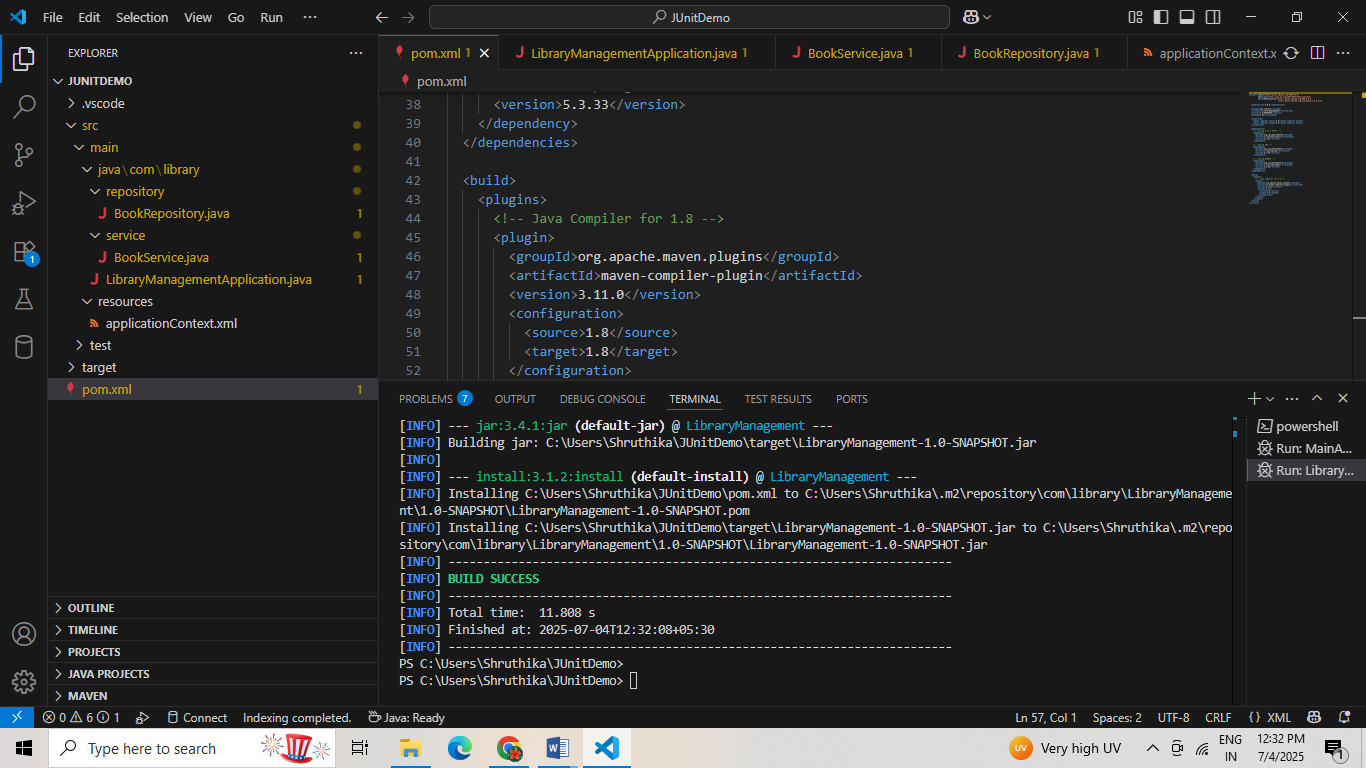
</plugin>

</plugins>

</build>

</project>

**OUTPUT:**

****

**ADDITIONAL HANDSON SPRING\_CORE MAVEN**

**Exercise 5: Configuring the Spring IoC Container**

**MY 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.example.library.BookRepository" />

<!-- Define BookService bean with setter injection -->

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

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

</bean>

</beans>

**MainApp.java**

package com.example.library;

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("Effective Java");

}

}

**Pom.xml**

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

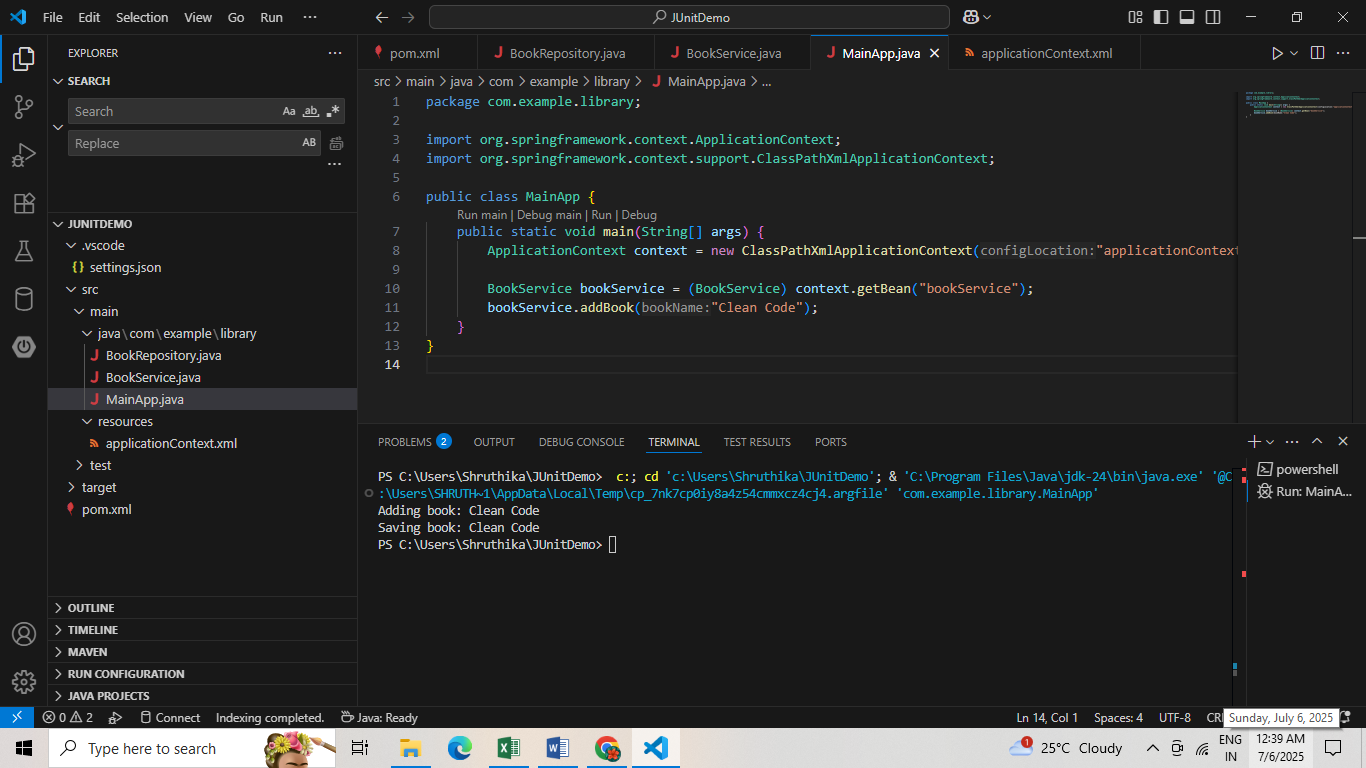
<artifactId>spring-context</artifactId>

<version>5.3.33</version> <!-- Use a version matching your JDK -->

</dependency>

</dependencies>

**OUTPUT:**

****

**Exercise 7: Implementing Constructor and Setter Injection**

**MY CODE:**

**BookService.java**

package com.example.library;

public class BookService {

private String libraryName;

private BookRepository bookRepository

public BookService(String libraryName) {

this.libraryName = libraryName;

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.out.println("[" + libraryName + "] Adding book: " + bookName);

bookRepository.saveBook(bookName);

}

}

**BookRepository.java**

package com.example.library;

public class BookRepository {

public void saveBook(String bookName) {

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

}

}

**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">

<!-- BookRepository Bean -->

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

<!-- BookService Bean with constructor and setter injection -->

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

<!-- Constructor Injection -->

<constructor-arg value="City Central Library" />

<!-- Setter Injection -->

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

</bean>

</beans>

**MainApp.java**

package com.example.library;

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");

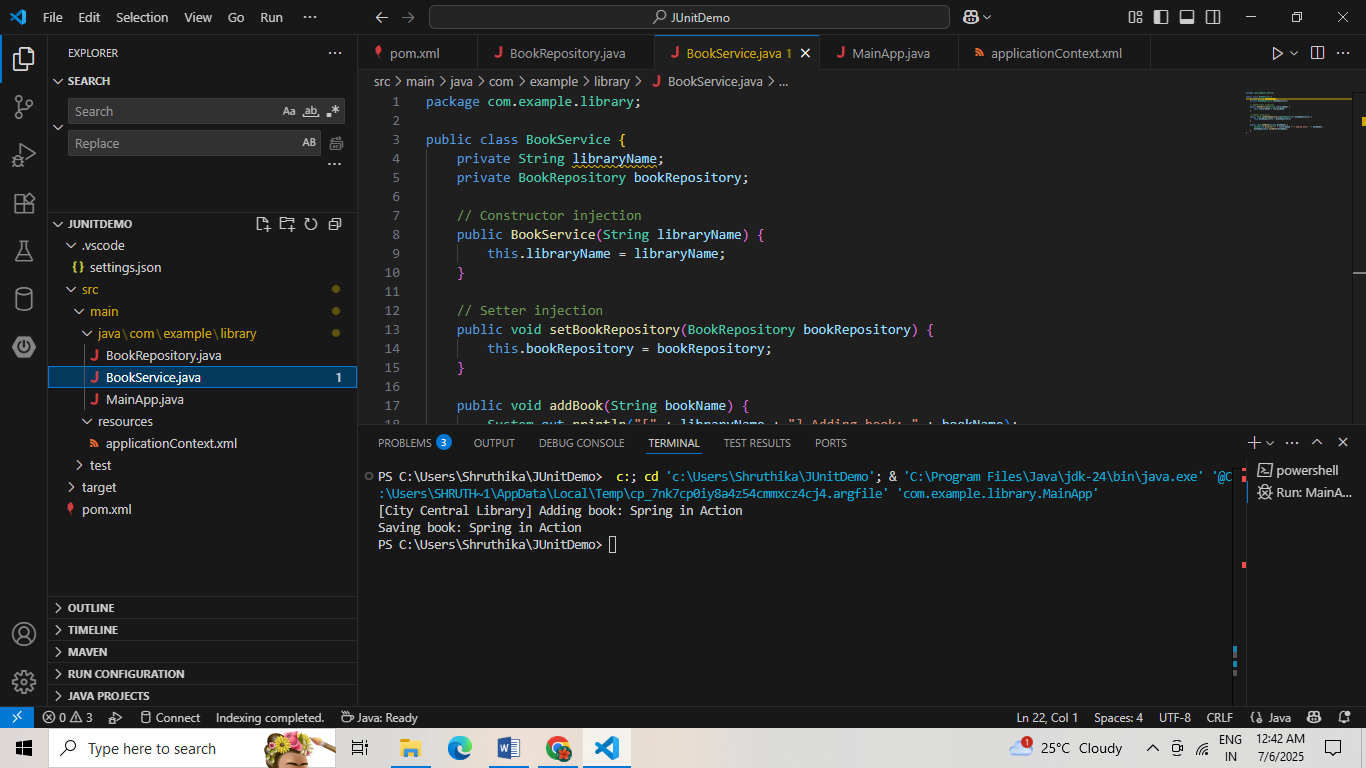
BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("Spring in Action");

}

}

**OUTPUT:**



**Exercise 9: Creating a Spring Boot Application**

Book.java

package com.model;

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

// Constructors

public Book() {}

public Book(String title, String author) {

this.title = title;

this.author = author;

}

// Getters & Setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

}

BookRepository.java

package com.repository;

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

import com.model.Book;

public interface BookRepository extends JpaRepository<Book, Long> {

}

**BookController.java**

package com.controller;

import java.util.List;

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

import org.springframework.web.bind.annotation.\*;

import com.model.Book;

import com.repository.BookRepository;

@RestController

@RequestMapping("/books")

public class BookController {

@Autowired

private BookRepository bookRepository;

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

}

HomeController.java

package com.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HomeController {

@GetMapping("/")

public String home() {

return "Welcome to the Library Management System!";

}

}

LibraryManagementApplication.java

package com;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryManagementApplication {

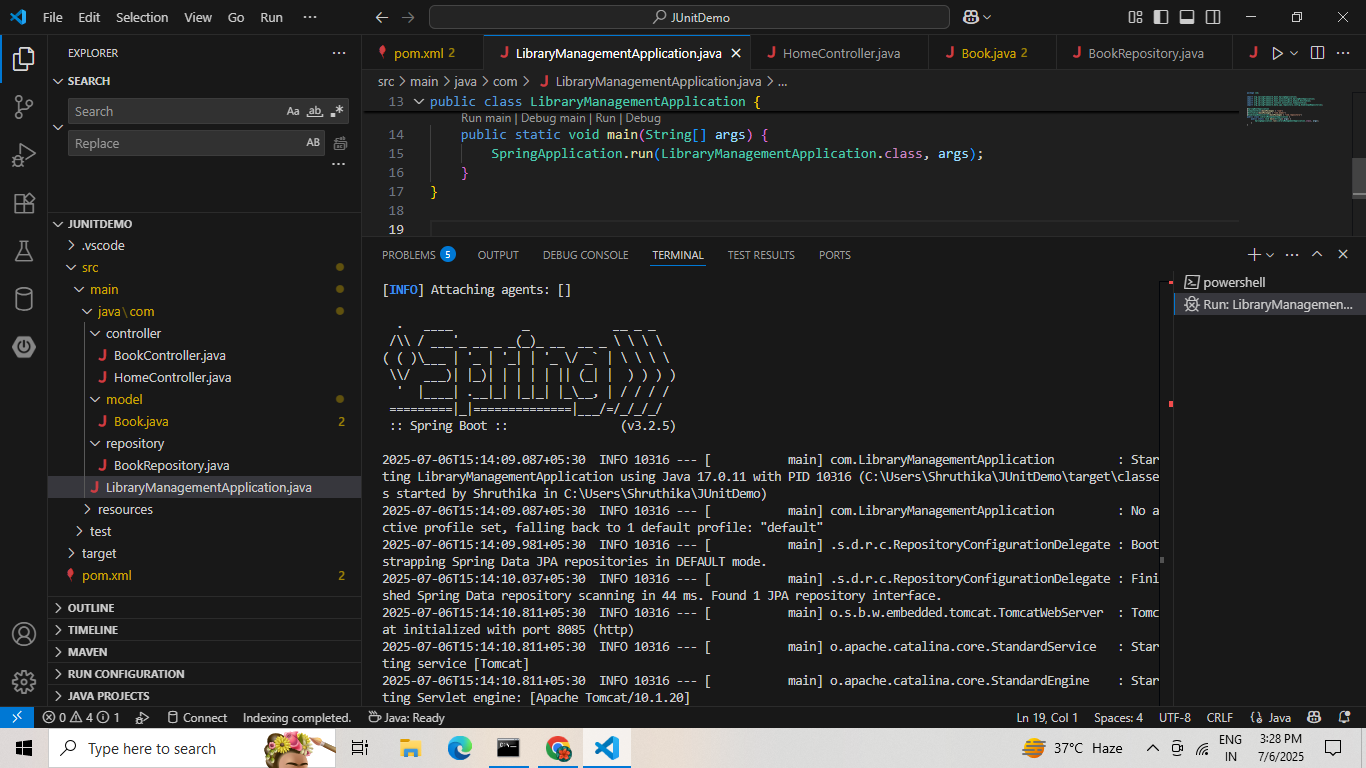
public static void main(String[] args) {

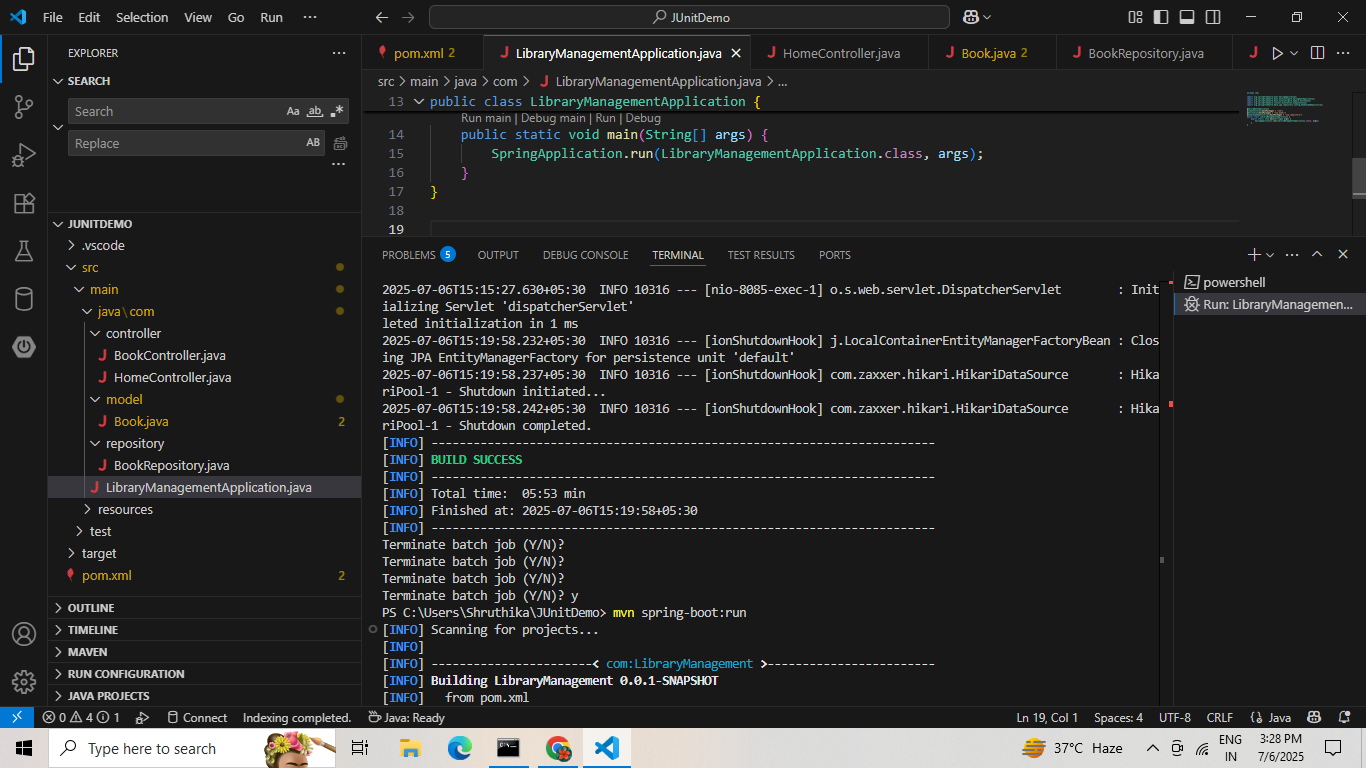
SpringApplication.run(LibraryManagementApplication.class, args);

}

}

**OUTPUT:**





**SPRING DATA\_JPA HANDSON**

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

**MY CODE:**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication {

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

}

**Country.java**

package com.cognizant.orm\_learn;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Country {

@Id

private String code;

private String name;

// getters and setters

}

CountryService.java

package com.cognizant.orm\_learn;

import jakarta.persistence.EntityManager;

import jakarta.persistence.PersistenceContext;

import jakarta.transaction.Transactional;

import org.springframework.stereotype.Service;

@Service

public class CountryService {

@PersistenceContext

private EntityManager entityManager;

@Transactional

public void addCountry() {

Country country = new Country();

country.setCode("IN");

country.setName("India");

entityManager.persist(country);

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

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

**build gradle**

plugins {

id 'org.springframework.boot' version '3.1.2'

id 'io.spring.dependency-management' version '1.1.0'

id 'java'

}

group = 'com.cognizant

version = '0.0.1-SNAPSHOT'

sourceCompatibility = '17'

repositories {

mavenCentral()

}

dependencies {

implementation 'org.springframework.boot:spring-boot-starter-data-jpa'

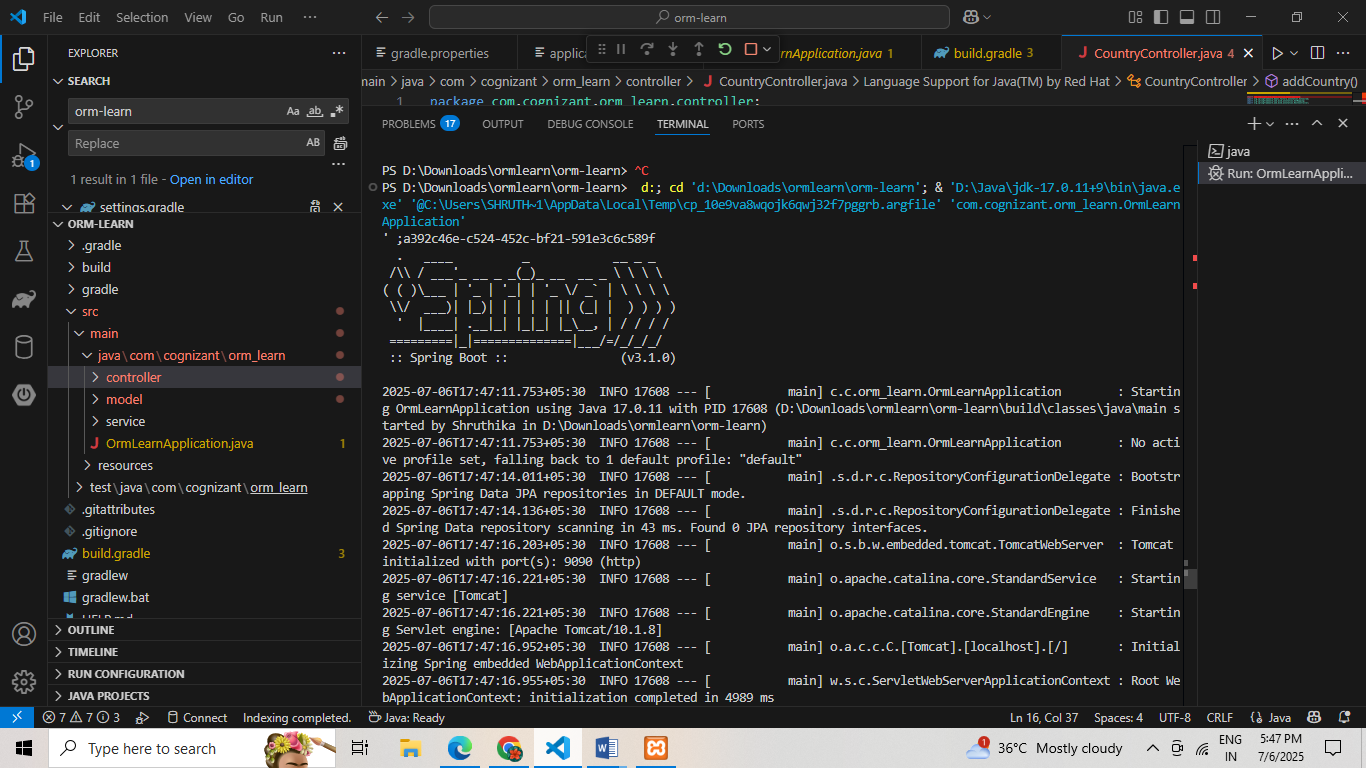
implementation 'org.springframework.boot:spring-boot-starter-web'

runtimeOnly 'com.mysql:mysql-connector-j'

testImplementation 'org.springframework.boot:spring-boot-starter-test'

}

**OUTPUT:**

****

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

**MY CODE:**

**Employee.java**

package com.example.employee\_demo.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Employee {

@Id

private int id;

private String name;

private String department;

// Getters and setters

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getDepartment() { return department; }

public void setDepartment(String department) { this.department = department; }

}

**EmployeeRepository.java**

package com.example.employee\_demo.repository;

import com.example.employee\_demo.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

**EmployeeService.java**

package com.example.employee\_demo.service;

import com.example.employee\_demo.model.Employee;

import com.example.employee\_demo.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**EmployeeController.java**

package com.example.employee\_demo.controller;

import com.example.employee\_demo.model.Employee;

import com.example.employee\_demo.service.EmployeeService;

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

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/employee")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping("/add")

public String addEmployee(@RequestBody Employee employee) {

employeeService.addEmployee(employee);

return "Employee added successfully!";

}

}

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/employee\_db

spring.datasource.username=root

spring.datasource.password=

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

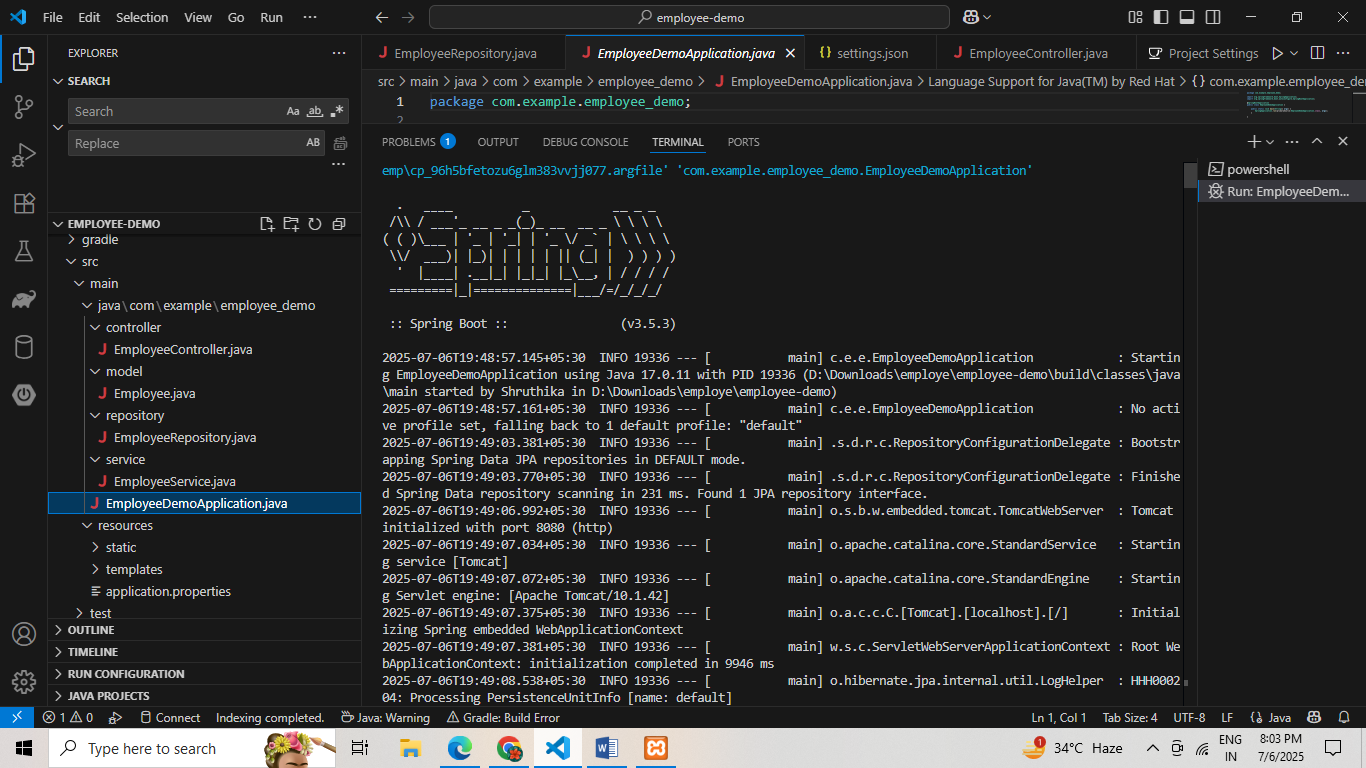
spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

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

server.port=8080

**OUTPUT:**

****