**WEEK 3**

**Spring Core and Maven, Spring Data JPA with Spring Boot, Hibernate**

**Spring Core and Maven :-**

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.

**Steps:**

1. **Set Up a Spring Project:**
   1. Create a Maven project named **LibraryManagement**.
   2. Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   1. Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   2. Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   1. Create a package **com.library.service** and add a class **BookService**.
   2. Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**
   1. Create a main class to load the Spring context and test the configuration.

**Step – 1:- Create a Maven project named LibraryManager.**

**Step – 2:- Add Spring Dependency in Pom.xml:**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

**Step – 3:-** Create a file named spring-config.xml inside the resources directory

<?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="libraryRepo" class="com.example.library.data.LibraryRepository" />

<bean id="libraryService" class="com.example.library.services.LibraryService">

<property name="repository" ref="libraryRepo" />

</bean>

</beans>

**Step – 4:- Create Component Classes**

**LibraryRepository.java**

package com.example.library.data;

public class LibraryRepository {

public void insertBook(String title) {

System.out.println("Inserted '" + title + "' into library database.");

}

}

**LibraryService.java**

package com.example.library.services;

import com.example.library.data.LibraryRepository;

public class LibraryService {

private LibraryRepository repository;

public void setRepository(LibraryRepository repository) {

this.repository = repository;

}

public void registerBook(String title) {

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

repository.insertBook(title);

}

}

**Main Class - MainApp.java**

package com.example.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.library.services.LibraryService;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("spring-config.xml");

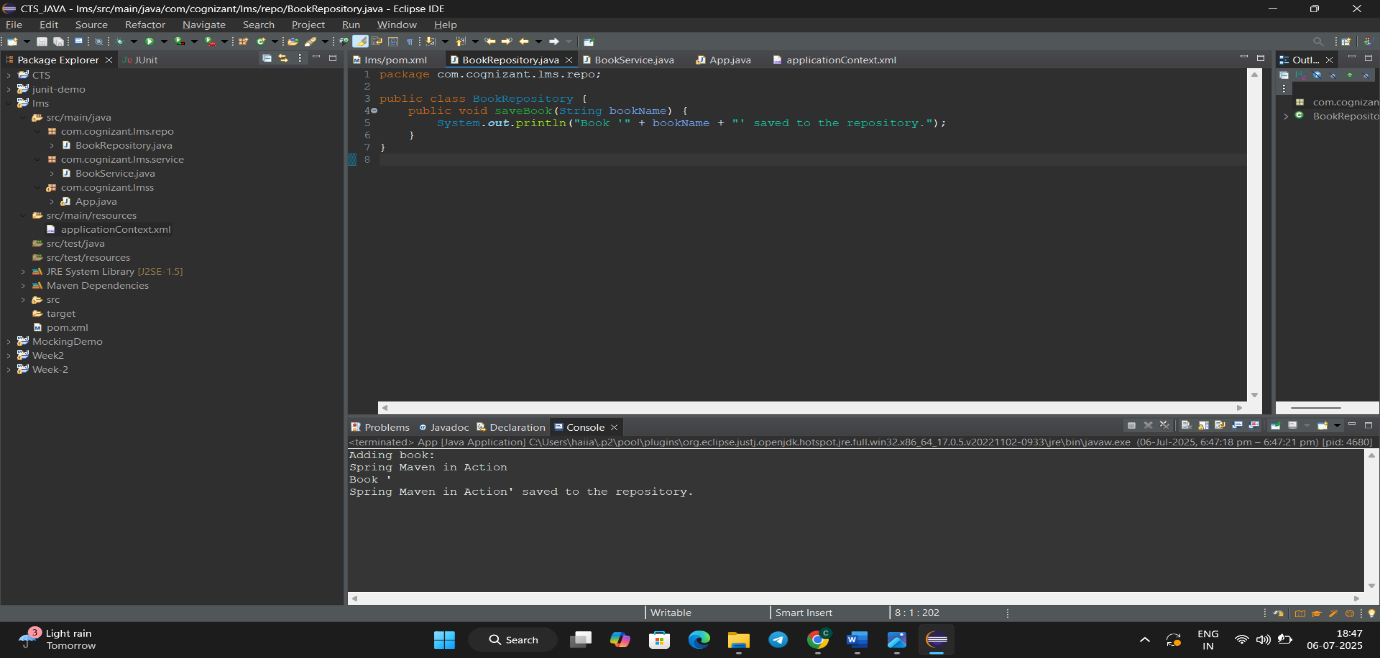
LibraryService service = (LibraryService) context.getBean("libraryService");

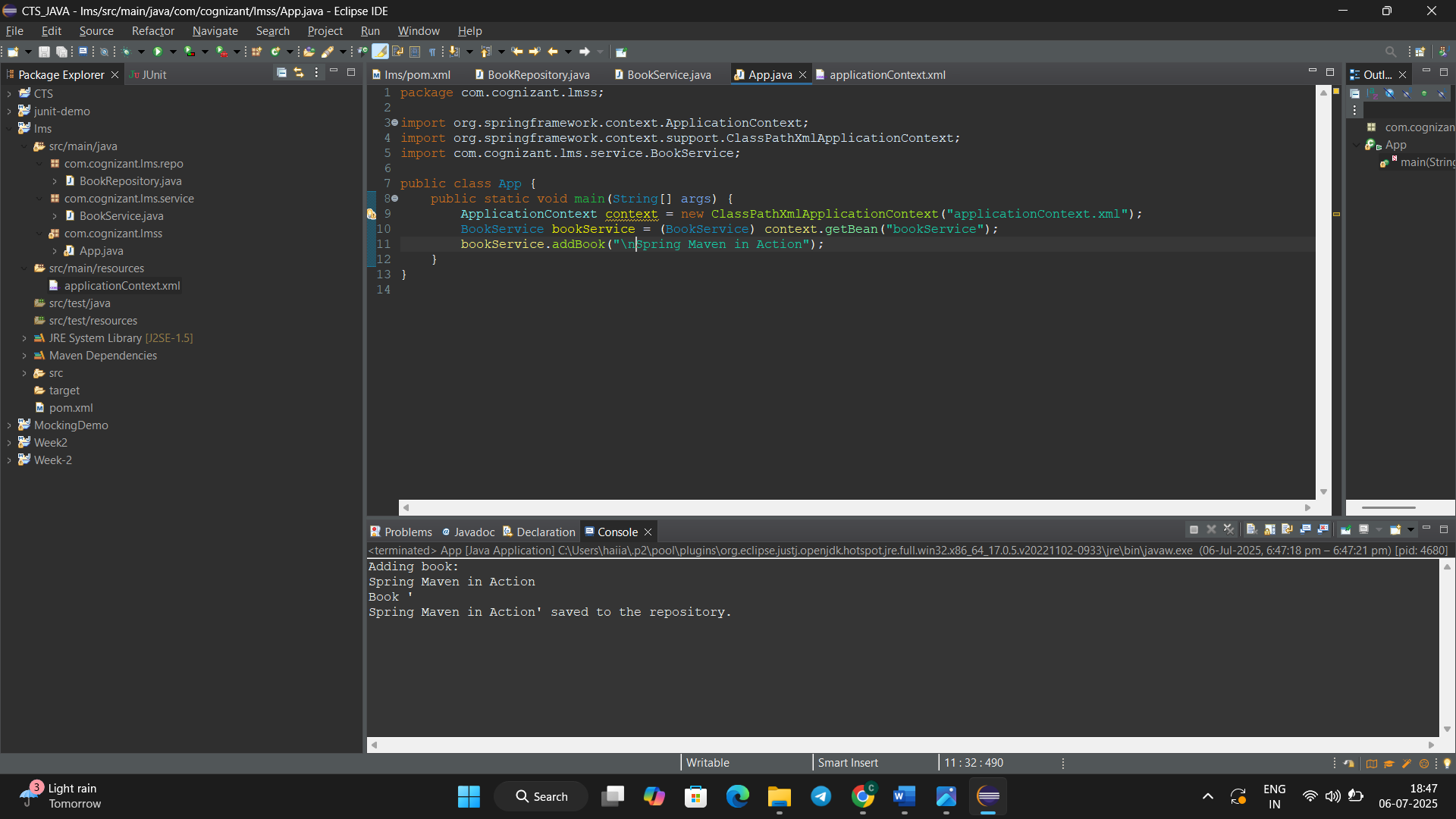
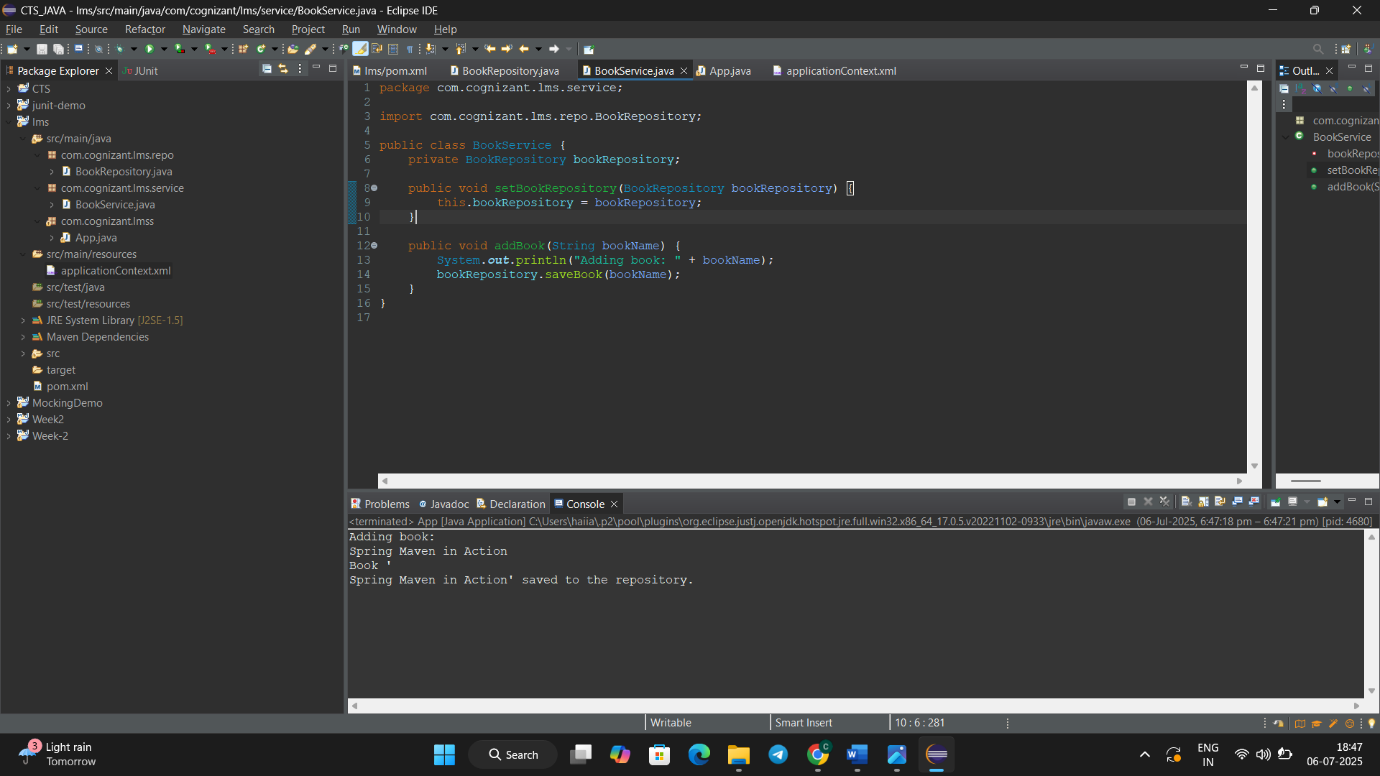
service.registerBook("Mastering Spring Framework");

}

}

**Output:-**





1. **Implementing Dependency Injection:**

**Step – 1:- Create a Maven project named LibraryManager.**

**Step – 2:- Add Spring Dependency in Pom.xml:**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

**Step – 3:-** Create a file named spring-config.xml inside the resources directory

<?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="libraryRepo" class="com.example.library.data.LibraryRepository" />

<bean id="libraryService" class="com.example.library.services.LibraryService">

<property name="repository" ref="libraryRepo" />

</bean>

</beans>

**Step – 4:- Create Component Classes**

**LibraryRepository.java**

package com.example.library.data;

public class LibraryRepository {

public void insertBook(String title) {

System.out.println("Inserted '" + title + "' into library database.");

}

}

**LibraryService.java**

package com.example.library.services;

import com.example.library.data.LibraryRepository;

public class LibraryService {

private LibraryRepository repository;

public void setRepository(LibraryRepository repository) {

this.repository = repository;

}

public void registerBook(String title) {

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

repository.insertBook(title);

}

}

**Main Class - MainApp.java**

package com.example.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.library.services.LibraryService;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("spring-config.xml");

LibraryService service = (LibraryService) context.getBean("libraryService");

service.registerBook("Mastering Spring Framework");

}

}

**Output:-**

**A computer screen shot of a program

AI-generated content may be incorrect.**

1. **Creating and Configuring a Maven Project**

**Create a maven project Library Management.**

**Add Spring Dependencies in Pom.xml :**

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Servlet API -->

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

**Configure Maven Compiler Plugin:**

<build>

<plugins>

<plugin>

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

<version>3.10.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

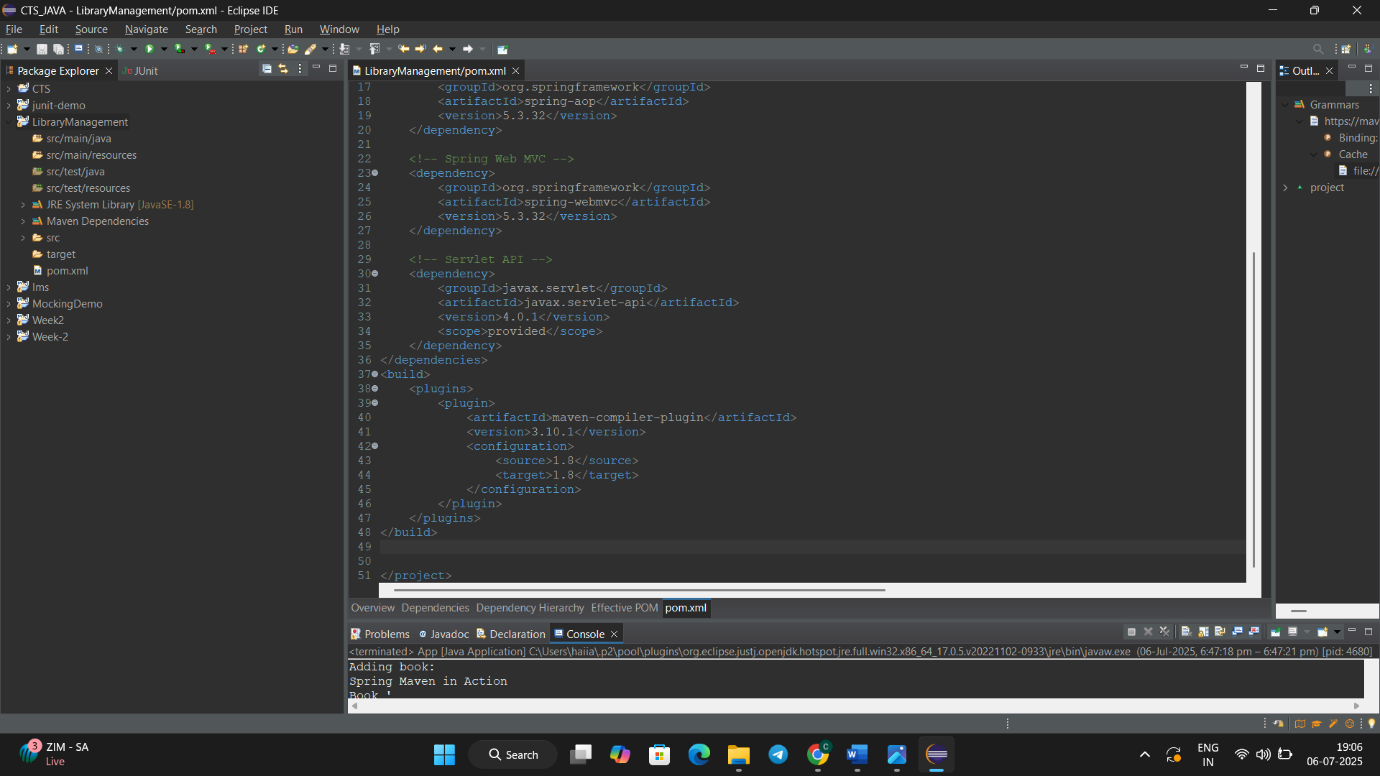
</plugins>

</build>

**Output:-**

**A computer screen shot of a program

AI-generated content may be incorrect.**

****

***Spring Data JPA with Spring Boot, Hibernate***

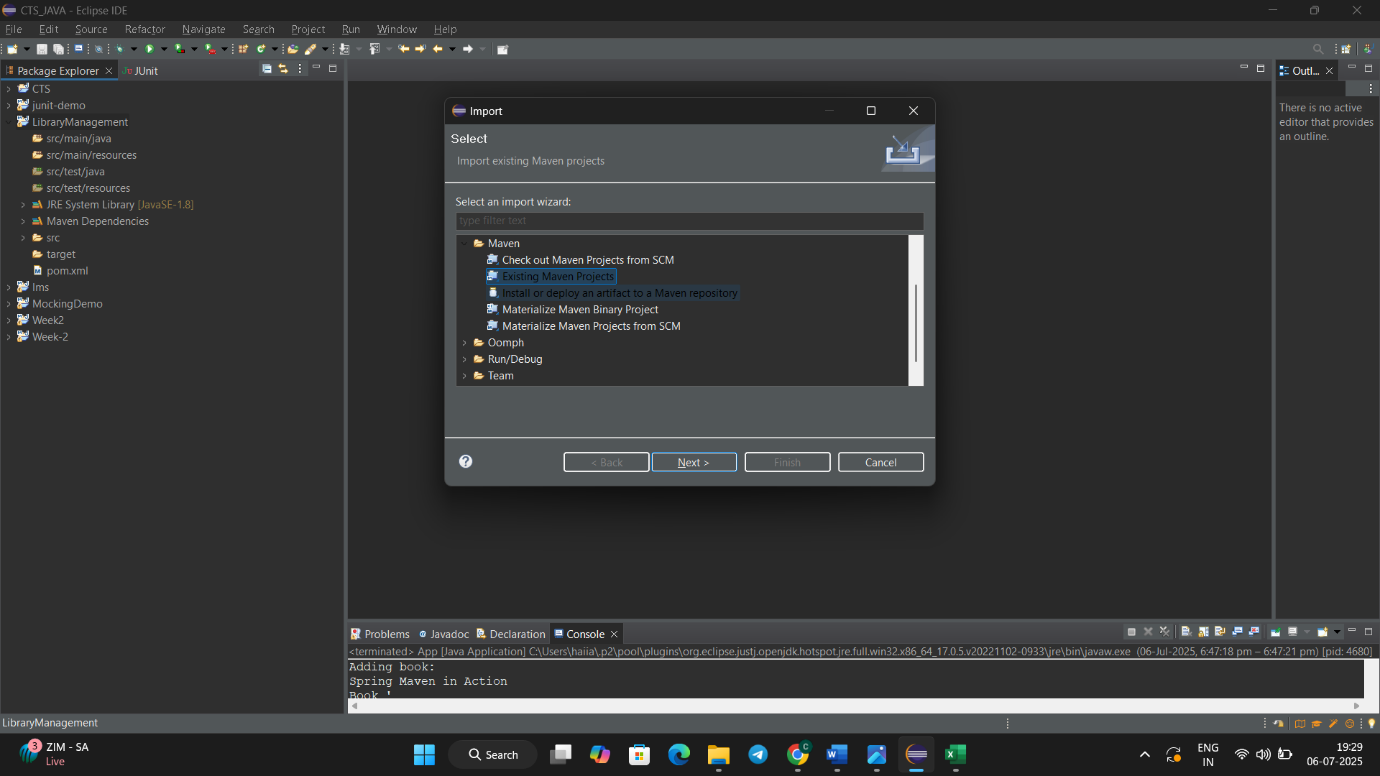
1. **Spring Data JPA - Quick Example:**

**Execution:**

**Steps :-**

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen with a black screen

AI-generated content may be incorrect.

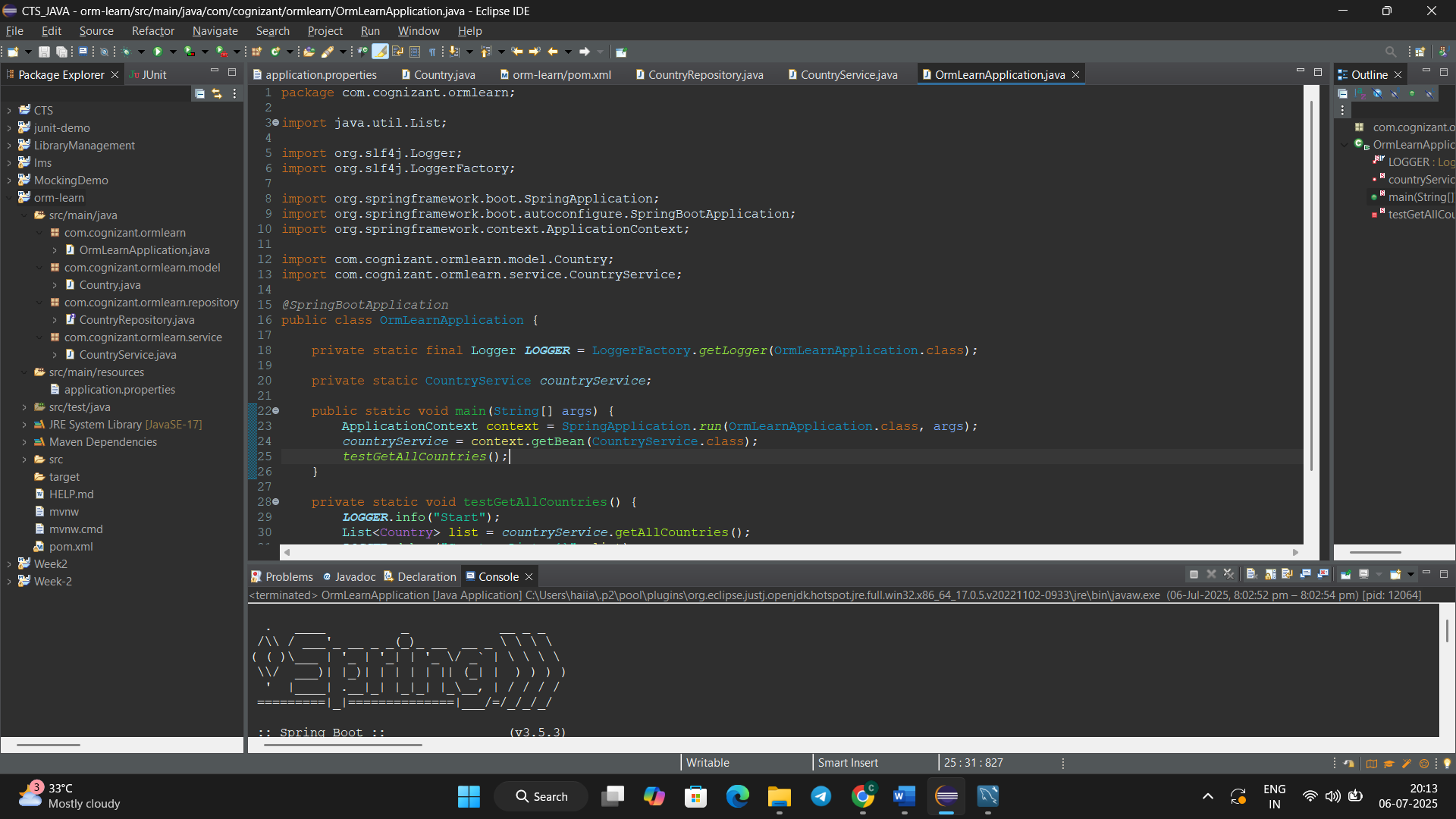
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.A computer screen shot of a computer program

AI-generated content may be incorrect.A computer screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

1. **Difference between JPA, Hibernate and Spring Data JPA:**
2. **JPA:-**

* Uses EntityManager
* JPA needs manual setup
* JPA/JPQL manual queries

**Code :-**

EntityManager em;

Country c = em.find(Country.class, "IN");

@PersistenceContext

EntityManager em;

List<Country> list = em.createQuery("SELECT c FROM Country c").getResultList();

1. **Hibernate:**

* Uses Session object for ORM operations
* Requires manual configuration of SessionFactory
* Uses HQL or native SQL for queries (must be written manually)

**Code:**

// Configuration

SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();

Session session = sessionFactory.openSession();

// Retrieving by ID

Country c = session.get(Country.class, "IN");

// Query all countries

List<Country> list = session.createQuery("FROM Country").list();

1. **Spring Data JPA:**

* Uses repository interfaces (JpaRepository)
* No manual configuration needed — auto-configured by Spring Boot
* No queries required — supports derived query methods like findById(), findAll()

**Code:**

// Repository interface

public interface CountryRepository extends JpaRepository<Country, String> { }

// Autowiring the repo

@Autowired

CountryRepository repo;

// Retrieving by ID

Country c = repo.findById("IN").get();

// Get all countries

List<Country> list = repo.findAll();