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

**Name:Krishna prakash**

**Id: 6373973**

**1) Spring Data JPA - Quick Example**

**Software Pre-requisites**

* MySQL Server 8.0
* MySQL Workbench 8
* Eclipse IDE for Enterprise Java Developers 2019-03 R
* Maven 3.6.2

**Create a Eclipse Project using** **Spring Initializr:**

* Group: com.cognizant
* Artifact ID: orm-learn
* Description: Demo project for Spring Data JPA and Hibernate
* Dependencies: Select Spring Boot DevTools, Spring Data JPA, and MySQL Driver

**MySQL Schema and table creation:**

mysql> create schema ormlearn;

create table country(co\_code varchar(2) primary key, co\_name varchar(50));

insert into country values ('IN', 'India');

insert into country values ('US', 'United States of America');

**Code :**

**\*application.properties :**

# Spring Framework and application log

logging.level.org.springframework=info

logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output

logging.level.org.hibernate.SQL=trace

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

# Log pattern

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

# Database 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 configuration

spring.jpa.hibernate.ddl-auto=validate

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

**\*Persistence Class – Country :**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

@Entity

@Table(name="country")

public class Country {

@Id

@Column(name="co\_code")

private String code;

@Column(name="co\_name")

private String name;

public Country() { }

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 + "]";

}

}

**\*Repository Class – CountryRepository:**

package com.cognizant.orm\_learn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.orm\_learn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String>{

}

**\*Service Class – CountryService:**

package com.cognizant.orm\_learn.service;

import java.util.List;

import org.springframework.transaction.annotation.Transactional;

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

import org.springframework.stereotype.Service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**\*Testing in OrmLearnApplication.java:**

package com.cognizant.orm\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

private static void testGetAllCountries() {

LOGGER.info("Start");

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

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

LOGGER.info("End");

}

public static void main(String[] args) {

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

LOGGER.info("Inside main");

countryService = context.getBean(CountryService.class);

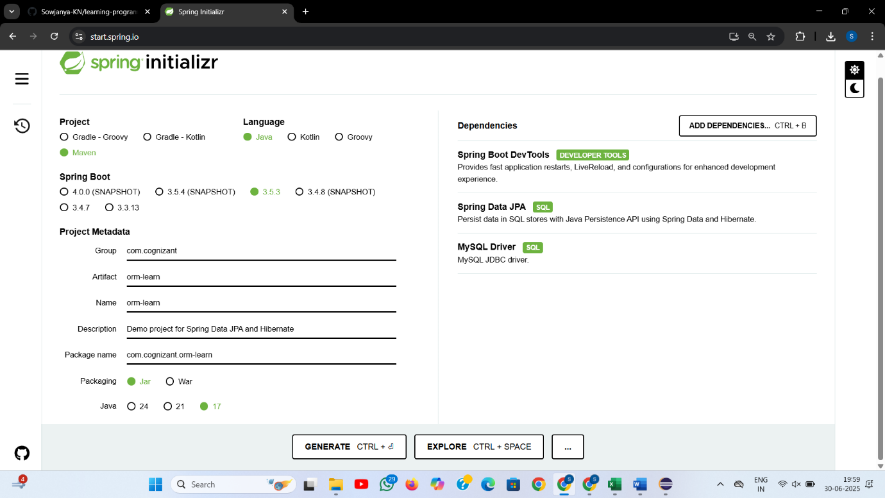
testGetAllCountries();

}

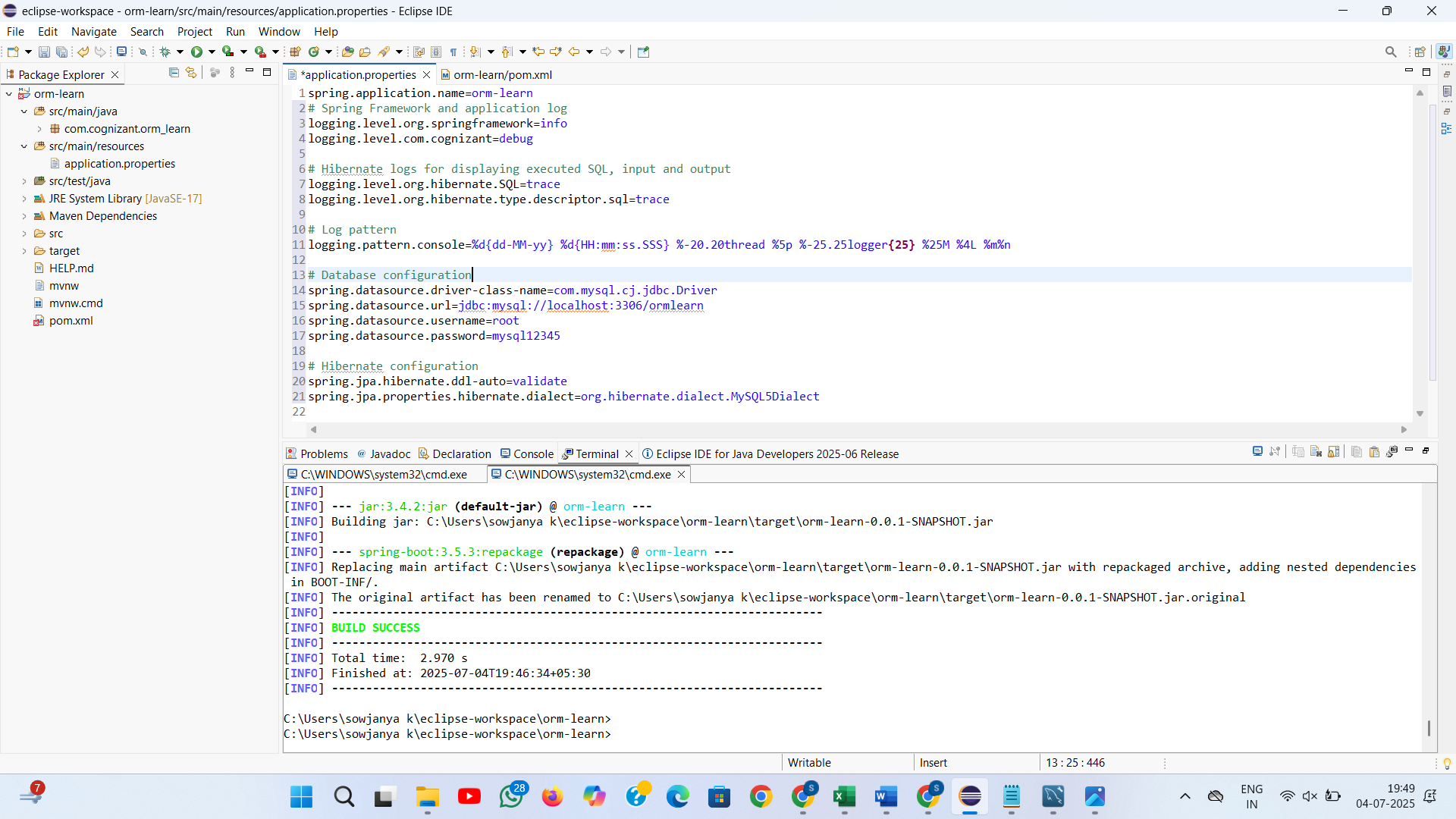
}

**OUTPUT :**

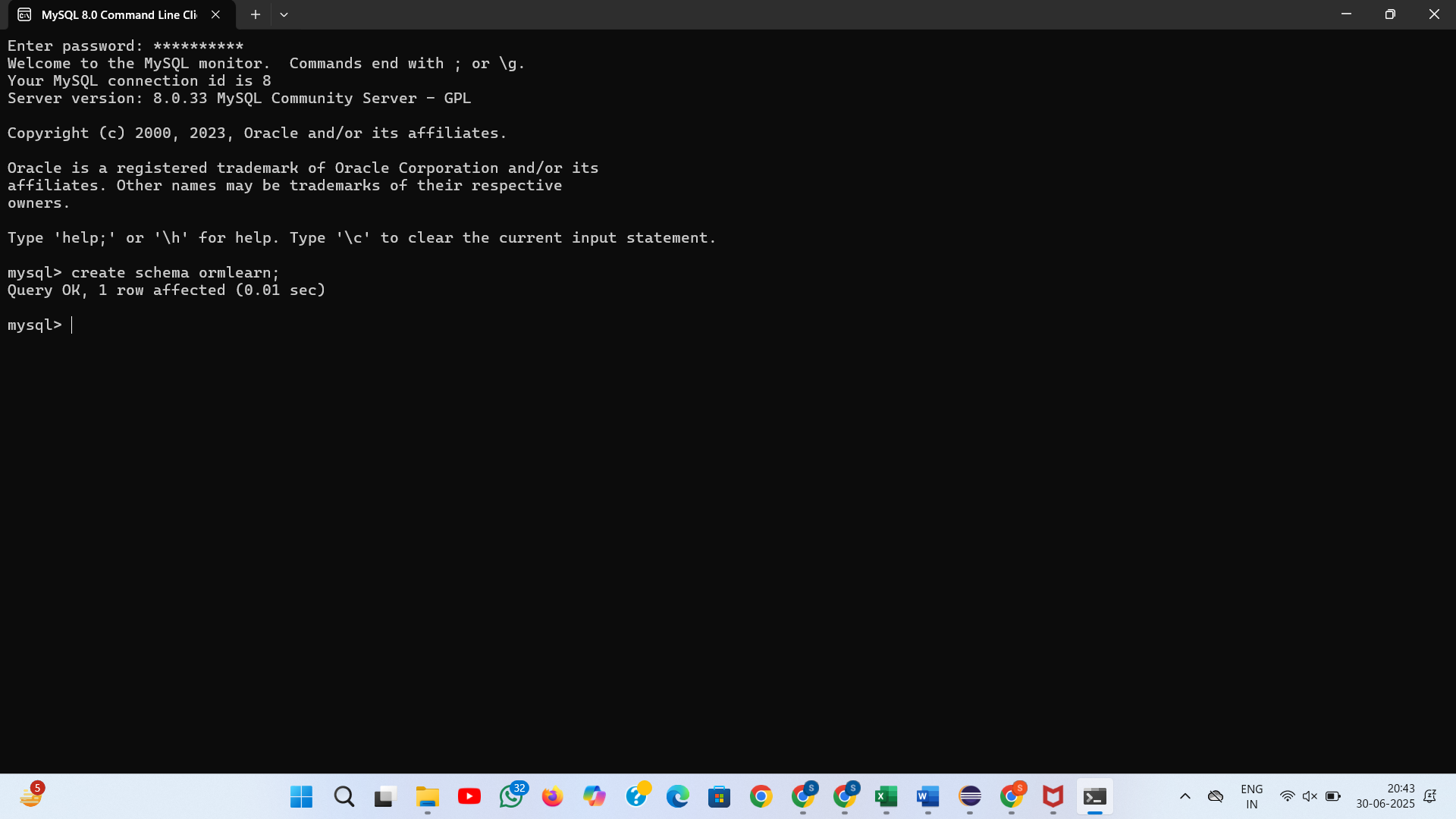
* **Spring Initializr:**



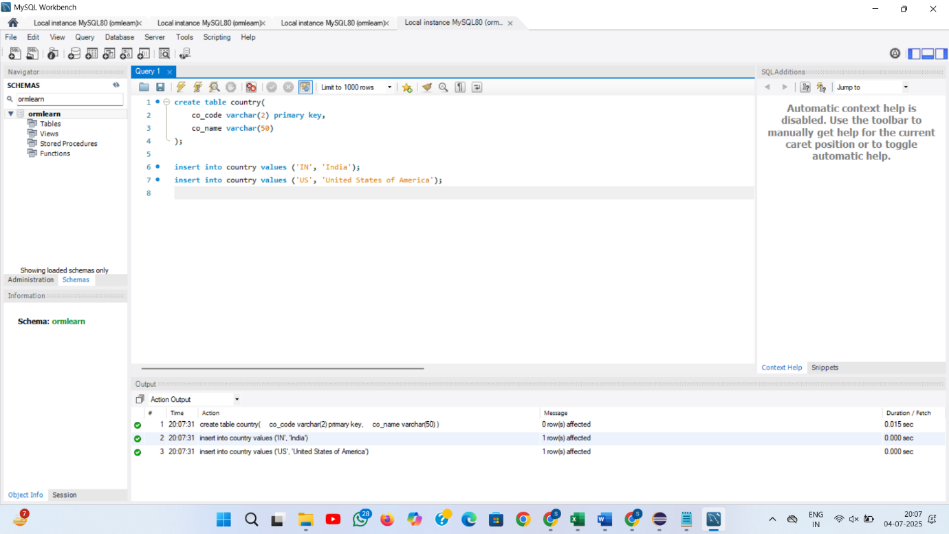
* **application.properties**



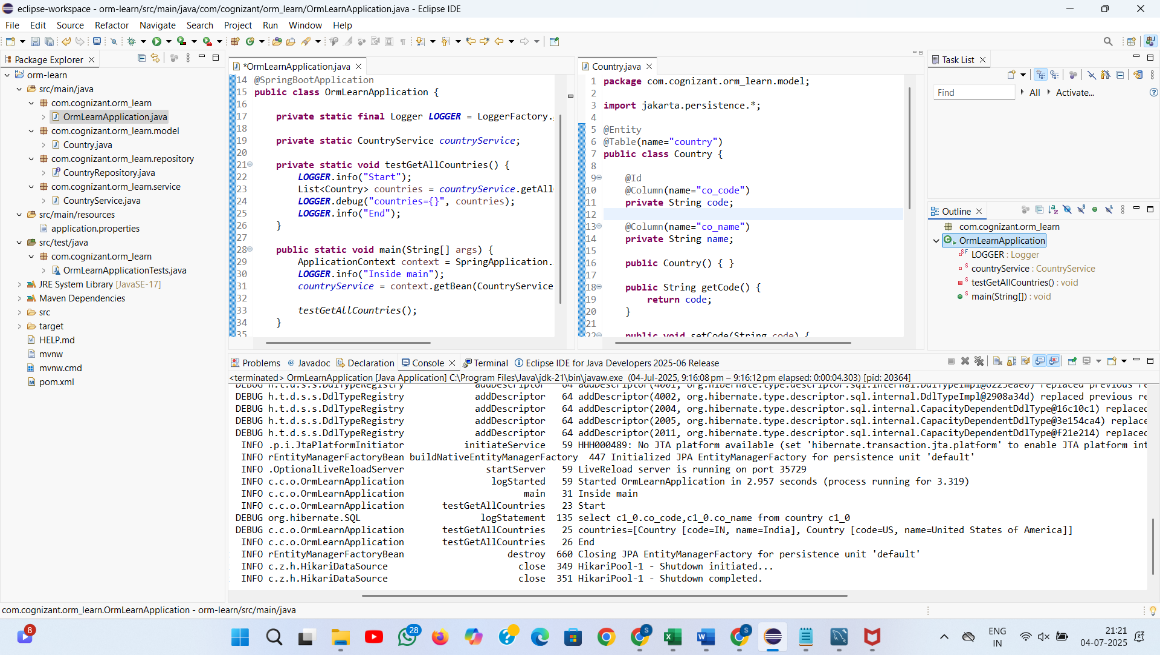
* **MySQL Schema creation:**



* **Country table creation:**

****

* **Data from ormlearn database is retrieved :**



**OUTPUT :** countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]

**2)** **Difference between JPA, Hibernate and Spring Data JPA:**

**Difference :**

**JPA (Java Persistence API)**

* JPA is a specification, not an implementation.
* It defines how Java objects are mapped to database tables.
* Cannot be used directly — needs a provider like Hibernate.

**Hibernate**

* Hibernate is a popular ORM tool that implements JPA.
* Handles all SQL operations, caching, and transaction management.
* You need to write code to manage sessions and transactions manually.

**Spring Data JPA**

* It is a Spring module that builds on top of JPA.
* Removes boilerplate code like writing DAO classes.
* Just extend JpaRepository and Spring handles everything.
* Internally, it uses Hibernate (or any JPA provider).

**Code snippets:**

* **Hibernate :**

/\* Method to CREATE an employee in the database \*/

    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:**

**EmployeeRespository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowire

  private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

  employeeRepository.save(employee);

  }

**Difference in Code: Hibernate vs Spring Data JPA**

**1. Session Management**

* Hibernate:  
  You have to manually open and close the Session using Session session = factory.openSession() and session.close().
* Spring Data JPA:  
  Spring automatically manages the EntityManager. You don’t write any session open/close code.

**2. Transaction Handling**

* Hibernate:  
  You manually begin and commit/rollback transactions using tx = session.beginTransaction() and tx.commit() or tx.rollback().
* Spring Data JPA:  
  You use @Transactional annotation and Spring handles transactions for you behind the scenes.

**3. Boilerplate Code**

* Hibernate:  
  You write a lot of repetitive code — session, transaction, exception handling, closing resources.
* Spring Data JPA:  
  No boilerplate code. Just call repository.save(entity) and Spring does everything.

**4. DAO/Repository**

* Hibernate:  
  You must write your own DAO classes and methods to save, update, delete, query data.
* Spring Data JPA:  
  Just create an interface that extends JpaRepository — Spring generates all CRUD methods automatically.

**5. Exception Handling**

* Hibernate:  
  You need to catch exceptions and handle rollback manually.
* Spring Data JPA:  
  Spring’s transaction manager handles exceptions and rollbacks automatically if @Transactional is used.

**OUTPUT:**

