**Cognizant Deep Skilling - Digital Nurture 4.0**

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

**Hands on 1**

**Spring Data JPA – Quick Example**

**Database Setup:**

CREATE SCHEMA ormlearn;

USE 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');

**application.properties Configuration**

# Logging

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

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

**Country.java**

package com.cognizant.orm\_learn.model;

import javax.persistence.\*;

@Entity

@Table(name="country")

public class Country {

    @Id

    @Column(name="code")

    private String code;

    @Column(name="name")

    private String name;

    // Getters and Setters

    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;

    }

    // toString

    @Override

    public String toString() {

        return "Country [code=" + code + ", name=" + name + "]";

    }

}

**CountryRepository.java**

package com.cognizant.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import java.util.List;

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

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

@SpringBootApplication

public class OrmLearnApplication {

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

    private static 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");

    }

}

**CountryService.java**

package com.cognizant.orm\_learn.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

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

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

@Service

public class CountryService {

    @Autowired

    private CountryRepository countryRepository;

    @Transactional

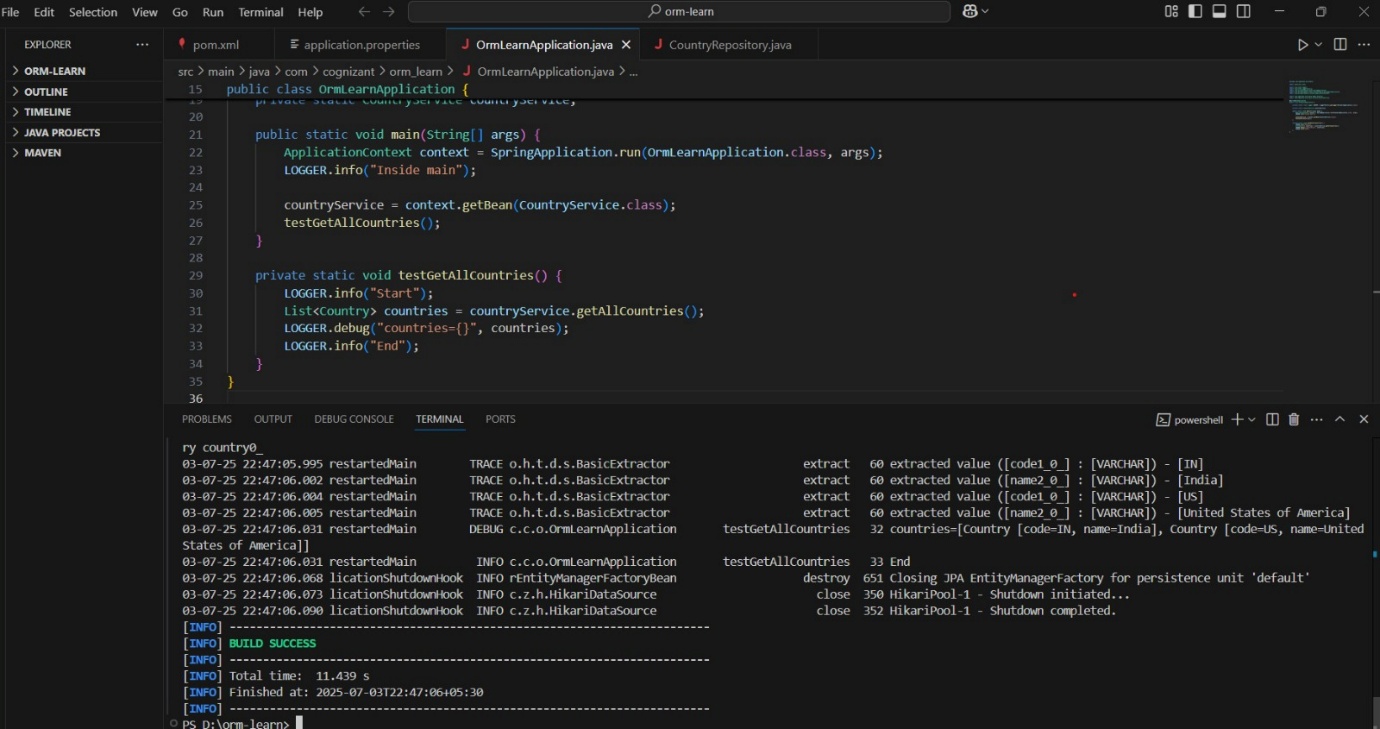
    public List<Country> getAllCountries() {

        return countryRepository.findAll();

    }

}

**Output:**



**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA**

**1. Java Persistence API (JPA)**

* JPA is a specification defined by JSR 338 for persisting, reading, and managing data between Java objects and relational databases.
* It provides a set of interfaces and annotations but does not include an implementation.
* JPA is just a standard — tools like Hibernate are used to implement JPA.

**2. Hibernate**

* Hibernate is a popular ORM (Object-Relational Mapping) framework that provides a concrete implementation of the JPA specification.
* It allows Java developers to map Java classes to database tables and provides data query and retrieval facilities.

**3. Spring Data JPA**

* Spring Data JPA is a part of the larger Spring ecosystem.
* It provides an abstraction layer over JPA, significantly reducing boilerplate code for data access operations.
* It does not provide a JPA implementation but builds upon a provider like Hibernate.
* Features include automatic query generation, transaction management, and repository support.

**Summary Table**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification | ORM Framework | Abstraction Layer |
| Implementation | Not provided | Yes (implements JPA) | No (depends on JPA provider like Hibernate) |
| Code Complexity | Moderate | High (manual session handling) | Low (auto-configuration, repository support) |
| Querying | JPQL | HQL, Criteria | Method naming conventions, JPQL, Criteria |
| Transaction | Manual or declarative | Manual | Declarative with @Transactional |

**Comparison Through Code Examples**

**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**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**JPA Code Example**

**Employee.java**

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

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;

}

}

**persistence.xml**

<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence

http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd"

version="2.1">

<persistence-unit name="EmployeePU">

<class>com.example.Employee</class>

<properties>

<property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/testdb"/>

<property name="javax.persistence.jdbc.user" value="root"/>

<property name="javax.persistence.jdbc.password" value="yourpassword"/>

<property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>

<property name="javax.persistence.schema-generation.database.action" value="create"/>

</properties>

</persistence-unit>

</persistence>

**EmployeeDAO.java**

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

public class EmployeeDAO {

private EntityManagerFactory emf = Persistence.createEntityManagerFactory("EmployeePU");

public void addEmployee(Employee employee) {

EntityManager em = emf.createEntityManager();

try {

em.getTransaction().begin();

em.persist(employee);

em.getTransaction().commit();

} catch (Exception e) {

if (em.getTransaction().isActive()) {

em.getTransaction().rollback();

}

e.printStackTrace();

} finally {

em.close();

}

}

}

**MainApp.java**

public class MainApp {

public static void main(String[] args) {

Employee employee = new Employee();

employee.setName("Abinaya");

employee.setDepartment("IT");

EmployeeDAO dao = new EmployeeDAO();

dao.addEmployee(employee);

System.out.println("Employee saved successfully.");

}

}