**WEEK-3 :- SPRING DATA JPA WITH SPRING BOOT, HIBERNATE**

**HANDS-ON- 1 :- Spring Data JPA - Quick Example**

**CODE:-**

application.properties (in src/main/resources)-

# 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

# MySQL 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

Country.java (in com.cognizant.ormlearn.model)-

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@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.java (in com.cognizant.ormlearn.repository)-

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.java (in com.cognizant.ormlearn.service)-

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

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");

}

}

SQL Setup (in MySQL Workbench)-

CREATE DATABASE ormlearn;

USE ormlearn;

CREATE TABLE country (

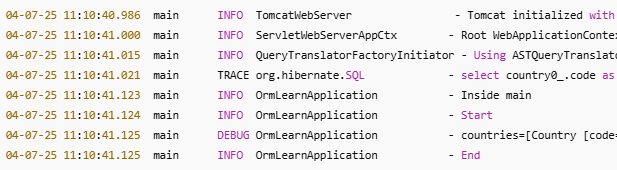
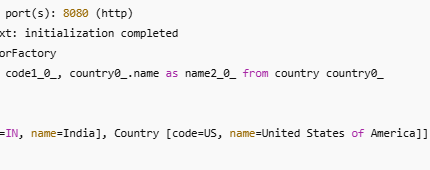
code VARCHAR(2) PRIMARY KEY,

name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

**OUTPUT:-**

**HANDS-ON- 2:- Difference between JPA, Hibernate and Spring Data JPA**

1. Java Persistence API (JPA):

* JPA is a specification (JSR 338) provided by Java for managing relational data in Java applications.
* It defines interfaces and rules, but it does not provide any implementation by itself.
* Think of JPA as a contract for how Java should interact with databases.

2. Hibernate:

* Hibernate is an Object Relational Mapping (ORM) tool and the most popular implementation of JPA.
* It provides the actual working code for the interfaces defined by JPA.
* You can use Hibernate with or without JPA.
* Hibernate provides additional features like caching, lazy loading, and custom queries.

3. Spring Data JPA:

* Spring Data JPA is a higher-level abstraction built on top of JPA (and typically Hibernate).
* It helps reduce boilerplate code (like writing queries and DAO layers).
* It provides features like CrudRepository and JpaRepository which offer built-in methods like findAll(), save(), deleteById() etc.
* Spring Data JPA uses Hibernate under the hood, but simplifies developer work.

JPA Example -

EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-persistence-unit");

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

Book book = new Book();

book.setId(1);

book.setTitle("JPA Book");

em.persist(book);

em.getTransaction().commit();

em.close();

emf.close();

Hibernate Example -

Configuration config = new Configuration().configure();

SessionFactory factory = config.buildSessionFactory();

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

Book book = new Book();

book.setId(2);

book.setTitle("Hibernate Book");

session.save(book);

tx.commit();

session.close();

Spring Data JPA Example -

@RestController

public class BookController {

@Autowired

private BookRepository bookRepository;

@PostMapping("/add")

public Book saveBook(@RequestBody Book book) {

return bookRepository.save(book); // Saves without needing manual transactions

}

}