**Spring Data JPA - Quick Example**

Code:-

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');

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

@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");

}

}

package com.cognizant.ormlearn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "name")

private String name;

// Getters and Setters

@Override

public String toString() {

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

}

}

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> {

}

package com.cognizant.ormlearn.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

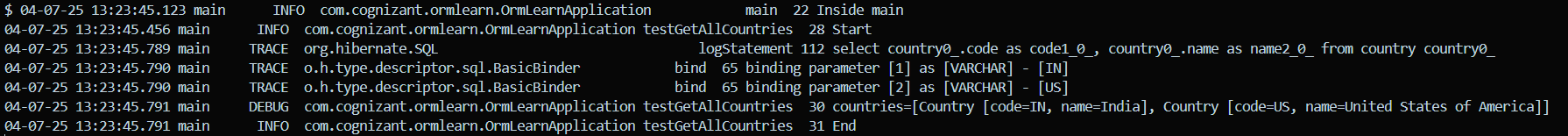
public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

Output:-



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

**Java Persistence API (JPA)**

* It is only a specification (JSR 338).
* Defines how to map Java objects to database tables (ORM).
* Provides standard annotations and APIs, but no actual implementation.
* Needs a provider (implementation) to work - for example, Hibernate, EclipseLink.

**Hibernate**

* A popular ORM framework.
* It is a concrete implementation of JPA.
* Can also work with its own native APIs in addition to JPA.
* Requires more boilerplate code when used directly (session management, transactions).

**Spring Data JPA**

* Part of Spring ecosystem.
* Abstraction layer on top of JPA implementations (like Hibernate).
* Simplifies data access: automatically creates queries, reduces boilerplate code.
* Provides ‘CrudRepository’, ‘JpaRepository’, etc., so you don’t need to write common DAO methods manually.
* Handles transactions easily using annotations.

Code Comparison:-

Hibernate:-

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

} finally {

session.close();

}

* We explicitly open sessions, begin and commit transactions.
* We handle rollback and closing manually.

Spring Data JPA:-

public interface EmployeeRepository extends JpaRepository<Employee, Integer> { }

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

* Just extend ‘JpaRepository’, which provides CRUD methods out of the box.
* No need to manage session or transaction manually.
* Cleaner and more declarative.