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

**CODE:**

**OrmLearnApplication.java:**

package com.cognizant.orm\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.LoggerFactory;

import org.slf4j.Logger;

import org.springframework.context.ApplicationContext;

import java.util.List;

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

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

@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();

public static void testGetAllCountries() {

LOGGER.info("Start");

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

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

LOGGER.info("End");

}

}

**application.properties:**

spring.application.name=orm-learn

# 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/orm\_learn

spring.datasource.username=root

spring.datasource.password=Snisha\*2005

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**Country.java:**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Table;

import jakarta.persistence.Column;

import jakarta.persistence.Id;

@Entity

@Table(name="country")

public class Country {

@Id

@Column(name="co\_code")

private String code;

@Column(name="co\_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 String.format("Code : %s Name : %s", code, name);

}

}

**CountryRepository.java:**

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> {

}

**CountryService.java:**

package com.cognizant.orm\_learn.service;

import org.springframework.stereotype.Service;

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

import java.util.List;

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

import jakarta.transaction.Transactional;

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

@Service

public class CountryService {

private final CountryRepository countryRepository;

public CountryService( @Autowired CountryRepository countryRepository) {

this.countryRepository = countryRepository;

}

@Transactional

public List<Country> getAllCountry() {

return countryRepository.findAll();

}

}

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>orm-learn</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

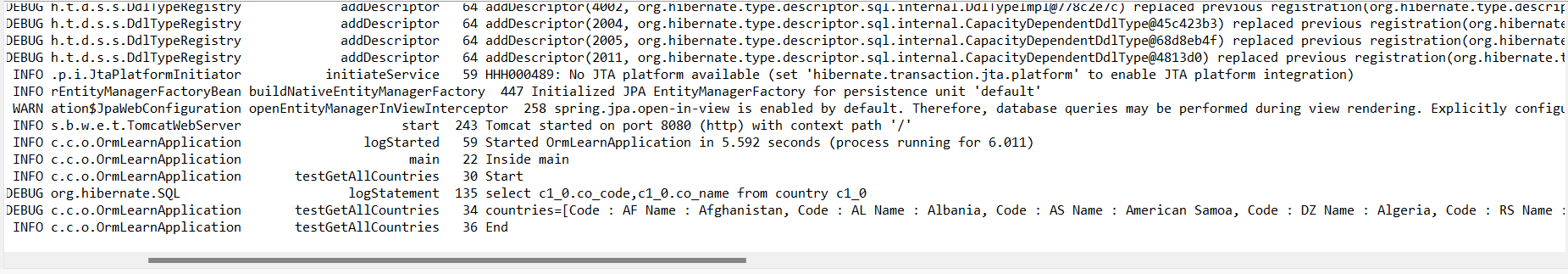
</plugin>

</plugins>

</build>

</project>

**OUTPUT:**



**Exercise:4 Difference between JPA, Hibernate and Spring Data JPA**

**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);

  }

|  |  |  |  |
| --- | --- | --- | --- |
| Feature / Aspect | JPA (Java Persistence API) | Hibernate | Spring Data JPA |
| Type | Specification (Interface) | Implementation (Framework) | Abstraction Layer over JPA |
| Purpose | Defines standard for ORM in Java | Provides actual implementation of JPA | Simplifies JPA-based data access |
| Contains Implementation | No (only interfaces and annotations) | Yes | No (uses JPA provider like Hibernate internally) |
| Who Provides It | Oracle (as part of Java EE standard) | Red Hat | Spring Framework (Pivotal) |
| Ease of Use | Requires more boilerplate code | Slightly easier with extra features | Very easy, minimal boilerplate with method query derivation |
| Query Language | JPQL (Java Persistence Query Language) | HQL (Hibernate Query Language) + supports JPQL | Supports JPQL + Query Methods + @Query annotation |
| Configuration Required | Manual setup | Manual setup + additional Hibernate configs | Auto-configuration with Spring Boot |
| Vendor Dependent | No (works with any JPA provider) | Yes (specific to Hibernate) | No (works with any JPA provider but typically uses Hibernate) |
| CRUD Operations | Manual implementation | Manual or using Hibernate features | Automatically handled using JpaRepository and derived methods |
| Learning Curve | Moderate | Moderate | Easy |
| Example Class | @Entity, @Id, EntityManager | Session, Criteria, @Entity, @Id | JpaRepository, @Repository, @Entity, @Id |