**Spring Data jpa**

**Hands-On**

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

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

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.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

@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);

*countryService* = context.getBean(CountryService.class);

*LOGGER*.info("Inside main");

*testGetAllCountries*();

}

private static void testGetAllCountries() {

*LOGGER*.info("Start");

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

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

*LOGGER*.info("End");

}

}

**application.properties:**

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=thushree76

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

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

**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>2.6.6</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>

<properties>

<java.version>11</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-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</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:**



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

**Java Persistence API (JPA)**

* JSR 338 Specification for persisting, reading and managing data from Java objects
* Does not contain concrete implementation of the specification
* Hibernate is one of the implementation of JPA

**Hibernate**

* ORM Tool that implements JPA

**Spring Data JPA**

* Does not have JPA implementation, but reduces boiler plate code
* This is another level of abstraction over JPA implementation provider like Hibernate
* Manages transactions

|  |  |  |
| --- | --- | --- |
| **Operation** | **Hibernate** | **Spring Data JPA** |
| Add Employee | 20 lines | 1 line |
| List Employees | 20 lines | 1 line |
| Update Employee | 18 lines | 1 line |
| Delete Employee | 18 lines | 1 line |
| **Total** | **76 lines** | **4 lines** |

**Code:**

**HibernateMain.java**

package com.example.main;

import com.example.dao.EmployeeDAO;

import com.example.entity.Employee;

import com.example.util.HibernateUtil;

import java.util.List;

public class HibernateMain {

public static void main(String[] args) {

EmployeeDAO employeeDAO = new EmployeeDAO();

System.*out*.println("=== HIBERNATE CRUD OPERATIONS ===");

System.*out*.println("\n1. Creating employees...");

Employee emp1 = new Employee("John", "Doe", "john@example.com", 50000.0);

Employee emp2 = new Employee("Jane", "Smith", "jane@example.com", 60000.0);

Employee emp3 = new Employee("Mike", "Johnson", "mike@example.com", 55000.0);

Integer emp1Id = employeeDAO.addEmployee(emp1);

Integer emp2Id = employeeDAO.addEmployee(emp2);

Integer emp3Id = employeeDAO.addEmployee(emp3);

System.*out*.println("Created Employee 1 with ID: " + emp1Id);

System.*out*.println("Created Employee 2 with ID: " + emp2Id);

System.*out*.println("Created Employee 3 with ID: " + emp3Id);

// READ - all employees

System.*out*.println("\n2. Listing all employees:");

List<Employee> employees = employeeDAO.listEmployees();

for (Employee employee : employees) {

System.*out*.println(employee);

}

// UPDATE - Update salary

System.*out*.println("\n3. Updating employee salary...");

employeeDAO.updateEmployee(emp1Id, 55000.0);

System.*out*.println("Updated Employee ID " + emp1Id + " salary to 55000.0");

// READ - after update

System.*out*.println("\n4. Listing employees after update:");

employees = employeeDAO.listEmployees();

for (Employee employee : employees) {

System.*out*.println(employee);

}

// DELETE - an employee

System.*out*.println("\n5. Deleting employee...");

employeeDAO.deleteEmployee(emp3Id);

System.*out*.println("Deleted Employee ID " + emp3Id);

// READ - Final list

System.*out*.println("\n6. Final employee list:");

employees = employeeDAO.listEmployees();

for (Employee employee : employees) {

System.*out*.println(employee);

}

// Shutdown

HibernateUtil.*shutdown*();

System.*out*.println("\n=== HIBERNATE DEMO COMPLETED ===");

}

}

**Output:**

