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

**1.Spring-data-jpa-handson**

**Hands on 1**

**Spring Data JPA - Quick Example**

**1. Project Setup**

* Group: com.cognizant
* Artifact: orm-learn
* Dependencies: Spring Boot DevTools, Spring Data JPA, MySQL Driver

**2. application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=validate

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

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

**3. Country Table Creation**

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(50)

);

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

**4. Country Entity**

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_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; }

@Override

public String toString() {

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

}

}

**6. Country Service**

package com.cognizant.ormlearn.service;

import java.util.List;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

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();

}

}

**7. Main Class Test**

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.context.ApplicationContext;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication {

private static CountryService countryService;

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

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();

for (Country c : countries) {

LOGGER.debug("Country: {}", c);

}

LOGGER.info("End");

}

}

**OUTPUT:**

Country: Country [code=IN, name=India]

Country: Country [code=US, name=United States of America]

**Hands on 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

**Spring Data JPA vs Hibernate example**

// Spring Data JPA

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Hands on 5**

**Implement services for managing Country**

**Hibernate Table Creation Configuration**

* create: Drops and recreates all tables every run
* validate: Checks for table/column existence; fails if not present
* update: Creates missing tables/columns without deleting data
* create-drop: Creates on startup and drops on shutdown

**application.properties**

spring.jpa.hibernate.ddl-auto=validate

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

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

**Country Entity:**

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

// Getters and Setters

@Override

public String toString() {

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

}

}

**CountryRepository:**

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String code) throws Exception {

return countryRepository.findById(code)

.orElseThrow(() -> new Exception("Country not found"));

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

public void updateCountry(String code, String newName) throws Exception {

Country country = findCountryByCode(code);

country.setName(newName);

countryRepository.save(country);

}

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

@Transactional

public List<Country> findCountriesByPartialName(String name) {

return countryRepository.findByNameContainingIgnoreCase(name);

}

}

**CountryService:**

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String code) throws Exception {

return countryRepository.findById(code)

.orElseThrow(() -> new Exception("Country not found"));

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

public void updateCountry(String code, String newName) throws Exception {

Country country = findCountryByCode(code);

country.setName(newName);

countryRepository.save(country);

}

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

@Transactional

public List<Country> findCountriesByPartialName(String name) {

return countryRepository.findByNameContainingIgnoreCase(name);

}

}

**OrmLearnApplication.java**

@SpringBootApplication

public class OrmLearnApplication {

private static CountryService countryService;

public static void main(String[] args) throws Exception {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

testFindByCode();

testAddCountry();

testUpdateCountry();

testDeleteCountry();

testFindByPartialName();

}

private static void testFindByCode() throws Exception {

Country country = countryService.findCountryByCode("IN");

System.out.println("Found: " + country);

}

private static void testAddCountry() throws Exception {

Country newCountry = new Country();

newCountry.setCode("XY");

newCountry.setName("Xyland");

countryService.addCountry(newCountry);

Country added = countryService.findCountryByCode("XY");

System.out.println("Added: " + added);

}

private static void testUpdateCountry() throws Exception {

countryService.updateCountry("XY", "Xyland Republic");

Country updated = countryService.findCountryByCode("XY");

System.out.println("Updated: " + updated);

}

private static void testDeleteCountry() {

countryService.deleteCountry("XY");

try {

countryService.findCountryByCode("XY");

} catch (Exception e) {

System.out.println("Deleted successfully: " + e.getMessage());

}

}

private static void testFindByPartialName() {

List<Country> countries = countryService.findCountriesByPartialName("land");

for (Country c : countries) {

System.out.println("Match: " + c);

}

}

}

**OUTPUT:**

Found: Country [code=IN, name=India]

Added: Country [code=XY, name=Xyland]

Updated: Country [code=XY, name=Xyland Republic]

Deleted successfully: Country not found

Match: Country [code=FI, name=Finland]

Match: Country [code=IS, name=Iceland]

**Hands on 6**

**Find a country based on country code**

**1.Create CountryNotFoundException**

package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**2. Modify CountryService – Method findCountryByCode**

import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

Optional<Country> result = countryRepository.findById(countryCode);

if (!result.isPresent()) {

throw new CountryNotFoundException("Country with code " + countryCode + " not found");

}

return result.get();

}

**3. Update OrmLearnApplication.java**

private static void getAllCountriesTest() {

LOGGER.info("Start");

try {

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country: {}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Error: {}", e.getMessage());

}

LOGGER.info("End");

}

**Call this method in main()**:

getAllCountriesTest();

**OUTPUT:**

INFO OrmLearnApplication - Start

DEBUG OrmLearnApplication - Country: Country [code=IN, name=India]

INFO OrmLearnApplication – End

**Hands on 7**

**Add a new country**

**1: Add Method in CountryService**

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

**2: Test in OrmLearnApplication**

private static void testAddCountry() {

LOGGER.info("Start - Add Country");

Country newCountry = new Country();

newCountry.setCode("ZZ");

newCountry.setName("Zetland");

countryService.addCountry(newCountry);

try {

Country country = countryService.findCountryByCode("ZZ");

LOGGER.debug("Added Country: {}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Country not found after add: {}", e.getMessage());

}

LOGGER.info("End - Add Country");

}

**Call in main()**:

testAddCountry();

**OUTPUT:**

Start - Add Country

Added Country: Country [code=ZZ, name=Zetland]

End - Add Country

**Hands on 8**

**Update a country based on code**

**1: Add Method in CountryService**

@Transactional

public void updateCountry(String code, String newName) throws CountryNotFoundException {

Country country = findCountryByCode(code);

country.setName(newName);

countryRepository.save(country);

}

**2: Test in OrmLearnApplication**

private static void testUpdateCountry() {

LOGGER.info("Start - Update Country");

try {

countryService.updateCountry("ZZ", "Republic of Zetland");

Country updated = countryService.findCountryByCode("ZZ");

LOGGER.debug("Updated Country: {}", updated);

} catch (CountryNotFoundException e) {

LOGGER.error("Update failed: {}", e.getMessage());

}

LOGGER.info("End - Update Country");

}

**OUTPUT:**

Start - Update Country

Updated Country: Country [code=ZZ, name=Republic of Zetland]

End - Update Country

**Hands on 9**

**Delete a country based on code**

**1: Add Method in CountryService**

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

**2: Test in OrmLearnApplication**

private static void testDeleteCountry() {

LOGGER.info("Start - Delete Country");

countryService.deleteCountry("ZZ");

try {

countryService.findCountryByCode("ZZ");

} catch (CountryNotFoundException e) {

LOGGER.debug("Country deleted successfully: {}", e.getMessage());

}

LOGGER.info("End - Delete Country");

}

**Call in main()**:

testDeleteCountry();

**OUTPUT:**

Start - Delete Country

Country deleted successfully: Country with code ZZ not found

End - Delete Country