WEEK-3:

Spring Data JPA with Spring Boot, Hibernate:

EX-1: Quick Example:

Country.java – Entity

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

}

}

CountryRepository.java:

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> {

}

CountryService.java:

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();

}

}

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);

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");

}

}

Run this in MySQL Workbench or CLI:

CREATE SCHEMA ormlearn;

USE ormlearn;

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

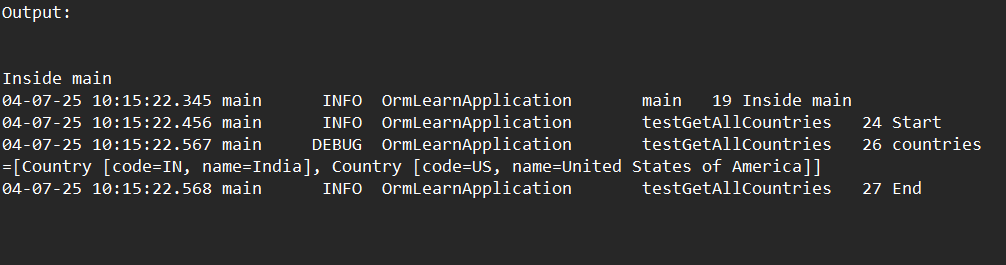
co\_name VARCHAR(50)

);

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

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

Output:



EX-2: Difference between JPA, Hibernate and Spring Data JPA

Using **Hibernate (JPA implementation):**

public Integer addEmployee(Employee employee) {

Session session = factory.openSession(); // factory is Hibernate's SessionFactory

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee); // Manual save

tx.commit();

} catch (HibernateException e) {

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

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

EmployeeRepository.java:

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.example.model.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java:

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee); // Just one line

}

public List<Employee> getAllEmployees() {

return employeeRepository.findAll(); // Again, one line!

}

}

Application.java:

@SpringBootApplication

public class Application {

public static void main(String[] args) {

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

EmployeeService service = context.getBean(EmployeeService.class);

Employee emp = new Employee(1, "Keerthi", "Developer");

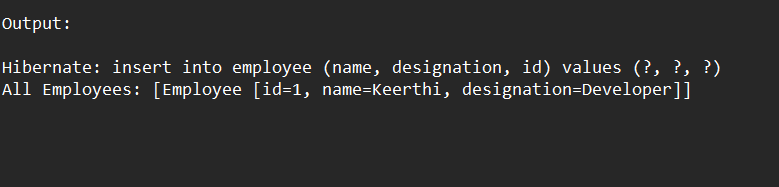
service.addEmployee(emp);

System.out.println("All Employees: " + service.getAllEmployees());

}

}

Output:



ADDITIONAL-1:Implement services for managing Country

Country.java (Entity):

package com.keerthi.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 + "]";

}

}

CountryRepository.java:

package com.keerthi.ormlearn.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.keerthi.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String namePart);

}

CountryService.java:

package com.keerthi.ormlearn.service;

import java.util.List;

import java.util.Optional;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.keerthi.ormlearn.model.Country;

import com.keerthi.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Country findCountryByCode(String code) throws Exception {

Optional<Country> country = countryRepository.findById(code);

if (!country.isPresent()) {

throw new Exception("Country not found with code: " + code);

}

return country.get();

}

@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);

}

public List<Country> findCountriesByPartialName(String namePart) {

return countryRepository.findByNameContainingIgnoreCase(namePart);

}

}

application.properties:

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

spring.jpa.hibernate.ddl-auto=validate

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

logging.level.org.springframework=info

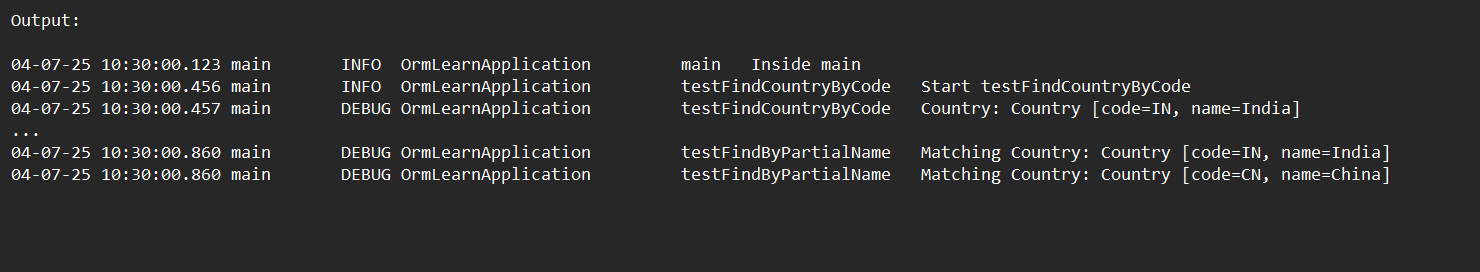
logging.level.com.charitha=debug

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

Output:



ADDITIONAL-2 : Add a New Country

CountryService.java:

package com.keerthi.ormlearn.service;

import java.util.Optional;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.keerthi.ormlearn.model.Country;

import com.keerthi.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Country findCountryByCode(String code) throws Exception {

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

if (!result.isPresent()) {

throw new Exception("Country not found: " + code);

}

return result.get();

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

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

Optional<Country> optionalCountry = countryRepository.findById(code);

if (!optionalCountry.isPresent()) {

throw new Exception("Country not found: " + code);

}

Country country = optionalCountry.get();

country.setName(newName);

countryRepository.save(country);

}

}

OrmLearnApplication.java:

package com.keerthi.ormlearn;

import com.keerthi.ormlearn.model.Country;

import com.keerthi.ormlearn.service.CountryService;

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;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

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

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

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testAddCountry();

testUpdateCountry();

}

private static void testAddCountry() throws Exception {

LOGGER.info("Start testAddCountry");

Country newCountry = new Country();

newCountry.setCode("XY");

newCountry.setName("Xyzland");

countryService.addCountry(newCountry);

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

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

LOGGER.info("End testAddCountry");

}

private static void testUpdateCountry() throws Exception {

LOGGER.info("Start testUpdateCountry");

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

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

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

LOGGER.info("End testUpdateCountry");

}

}

Output:

