**Week 3 : SPRING DATA JPA HANDSON**

**Handson 1**

**Country.java**

com.cognizant.ormlearn.model

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

}

**CountryRepository.java**

com.cognizant.ormlearn.repository

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

com.cognizant.ormlearn.service

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java**

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();

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

LOGGER.info("End");

}

private static CountryService countryService;

**application.properties**

logging.level.org.springframework=info

logging.level.com.cognizant=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

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

**Sql**

mysql -u root -p

create schema 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');

**Handson 2**

**Employee.java**

package com.example.hibernate;

public class Employee {

private int id;

private String firstName;

private String lastName;

private int salary;

public Employee() {}

public Employee(String firstName, String lastName, int salary) {

this.firstName = firstName;

this.lastName = lastName;

this.salary = salary;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + firstName + " " + lastName + ", salary=" + salary + "]";

}

}

**Employee.hbm.xml**

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.example.hibernate.Employee" table="EMPLOYEE">

<id name="id" column="ID">

<generator class="native"/>

</id>

<property name="firstName" column="FIRST\_NAME"/>

<property name="lastName" column="LAST\_NAME"/>

<property name="salary" column="SALARY"/>

</class>

</hibernate-mapping>

**Hibernate.cfg.xml**

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

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernatedb</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.show\_sql">true</property>

<mapping resource="Employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**MySQL Table**

CREATE DATABASE hibernatedb;

USE hibernatedb;

CREATE TABLE EMPLOYEE (

ID INT PRIMARY KEY AUTO\_INCREMENT,

FIRST\_NAME VARCHAR(50),

LAST\_NAME VARCHAR(50),

SALARY INT

);

**MainApp.java**

package com.example.hibernate;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class MainApp {

public static void main(String[] args) {

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee emp = new Employee("Leona", "Grace", 60000);

session.save(emp);

List<Employee> list = session.createQuery("from Employee").list();

for (Employee e : list) {

System.out.println(e);

}

Employee fetched = session.get(Employee.class, emp.getId()); // session.get()

System.out.println("Fetched: " + fetched);

session.delete(fetched);

tx.commit();

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

}

**Handson 3**

**Employee.java**

package com.example.hibernate;

import javax.persistence.\*;

@Entity

@Table(name = "EMPLOYEE")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "ID")

private int id;

@Column(name = "FIRST\_NAME")

private String firstName;

@Column(name = "LAST\_NAME")

private String lastName;

@Column(name = "SALARY")

private int salary;

public Employee() {}

public Employee(String firstName, String lastName, int salary) {

this.firstName = firstName;

this.lastName = lastName;

this.salary = salary;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + firstName + " " + lastName + ", salary=" + salary + "]";

}}

**Hibernate.cfg.xml**

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

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernatedb</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.show\_sql">true</property>

<mapping class="com.example.hibernate.Employee"/>

</session-factory>

</hibernate-configuration>

**MySQL**

CREATE DATABASE hibernatedb;

USE hibernatedb;

CREATE TABLE EMPLOYEE (

ID INT AUTO\_INCREMENT PRIMARY KEY,

FIRST\_NAME VARCHAR(50),

LAST\_NAME VARCHAR(50),

SALARY INT

);

**MainApp.java**

package com.example.hibernate;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class MainApp {

public static void main(String[] args) {

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee emp = new Employee("Leona", "Smith", 75000);

session.save(emp);

List<Employee> list = session.createQuery("from Employee").list();

for (Employee e : list) {

System.out.println(e);

}

tx.commit();

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();

factory.close();

}}}

**Handson 4**

**EntityManager(JPA)**

public interface EntityManager {

void persist(Object entity);

Object find(Class entityClass, Object primaryKey);

}

**Hibernate**

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**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**Service class**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**Handson 5**

**Country Entity**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@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**

package com.cognizant.ormlearn.repository;

import java.util.List;

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> {

List<Country> findByNameContainingIgnoreCase(String name);

}

**CountryService**

package com.cognizant.ormlearn.service;

import java.util.List;

import java.util.Optional;

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(readOnly = true)

public Country findCountryByCode(String code) {

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

return result.orElse(null);

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

public void updateCountry(Country country) {

if (countryRepository.existsById(country.getCode())) {

countryRepository.save(country);

} else {

throw new RuntimeException("Country not found");

}

}

@Transactional

public void deleteCountry(String code) {

if (countryRepository.existsById(code)) {

countryRepository.deleteById(code);

} else {

throw new RuntimeException("Country not found");

}

}

@Transactional(readOnly = true)

public List<Country> searchCountriesByName(String name) {

return countryRepository.findByNameContainingIgnoreCase(name);

}

}

**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.context.ApplicationContext;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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");

testCountryFeatures();

}

private static void testCountryFeatures() {

LOGGER.info("Start Test");

Country country = new Country();

country.setCode("ZZ");

country.setName("Zootopia");

countryService.addCountry(country);

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

country.setName("Zootopia Updated");

countryService.updateCountry(country);

LOGGER.info("Updated: {}", countryService.findCountryByCode("ZZ"));

List<Country> searchResults = countryService.searchCountriesByName("an");

LOGGER.info("Search Results: {}", searchResults);

countryService.deleteCountry("ZZ");

LOGGER.info("Deleted country ZZ");

LOGGER.info("End Test");

}

}

**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

**MySQL**

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(50)

);

**Handson 6**

**Exception Class**

package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**CountryService**

package com.cognizant.ormlearn.service;

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

@Transactional(readOnly = true)

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();

}

**OrmLearnApplication.java**

private static void testFindCountryByCode() {

LOGGER.info("Start");

try {

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

**Main**

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testFindCountryByCode();

}

**Handson 7**

**CountryService**

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);}

**OrmLearnApplication.java**

private static void testAddCountry() {

LOGGER.info("Start");

Country newCountry = new Country();

newCountry.setCode("ZZ");

newCountry.setName("Zootopia");

countryService.addCountry(newCountry);

try {

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

**Update Main**

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testFindCountryByCode();

testAddCountry();}

**Handson 8**

**UpdateCountry in CountryService**

@Transactional

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

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

if (!result.isPresent()) {

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

}

Country country = result.get();

country.setName(newName); // Update the name

countryRepository.save(country); // Save the updated entity

}

**OrmLearnApplication.java**

private static void testUpdateCountry() {

LOGGER.info("Start");

try {

countryService.updateCountry("ZZ", "Zootopia Prime");

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

**Update main()**

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testFindCountryByCode();

testAddCountry();

testUpdateCountry(); // <-- Add this

}

**Handson 9**

**Add deleteCountry() Method in CountryService**

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

**OrmLearnApplication.java**

private static void testDeleteCountry() {

LOGGER.info("Start");

countryService.deleteCountry("ZZ");

try {

countryService.findCountryByCode("ZZ");

} catch (CountryNotFoundException e) {

LOGGER.debug("Country not found as expected after deletion: {}", e.getMessage());

}

LOGGER.info("End");

}

**Update main() method**

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testFindCountryByCode();

testAddCountry();

testUpdateCountry();

testDeleteCountry(); // ← Add this

}