**WEEK-3\_Spring\_data\_Jpa\_HandsOn**

**SUPERSET ID :6393674**

**Hands on 1**

**Spring Data JPA - Quick Example   
  
SOLUTION:**

**MYSQL WORKBENCH:**

**MYSQL SCHEMA:**

mysql -u root -p

create schema ormlearn;

**CREATE TABLE:**

create table country (

code varchar(2) primary key,

name varchar(50)

);

insert into country values ('IN', 'India');

insert into country values ('US', 'United States of America');

**application.properties.xml**

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

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

# DB Config

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

spring.datasource.username=root

spring.datasource.password=Vasundhara@123

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

# Hibernate (New Dialect for Hibernate 6+)

spring.jpa.hibernate.ddl-auto=validate

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

**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>17</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>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>

**OrmLearnApplication.java**

**package** com.cognizant.orm\_learn;

**import** java.util.List;

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

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

**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) {

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");

}

}

**Country.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name="country")

public class Country {

@Id

@Column(name="code")

private String code;

@Column(name="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.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 java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

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

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

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

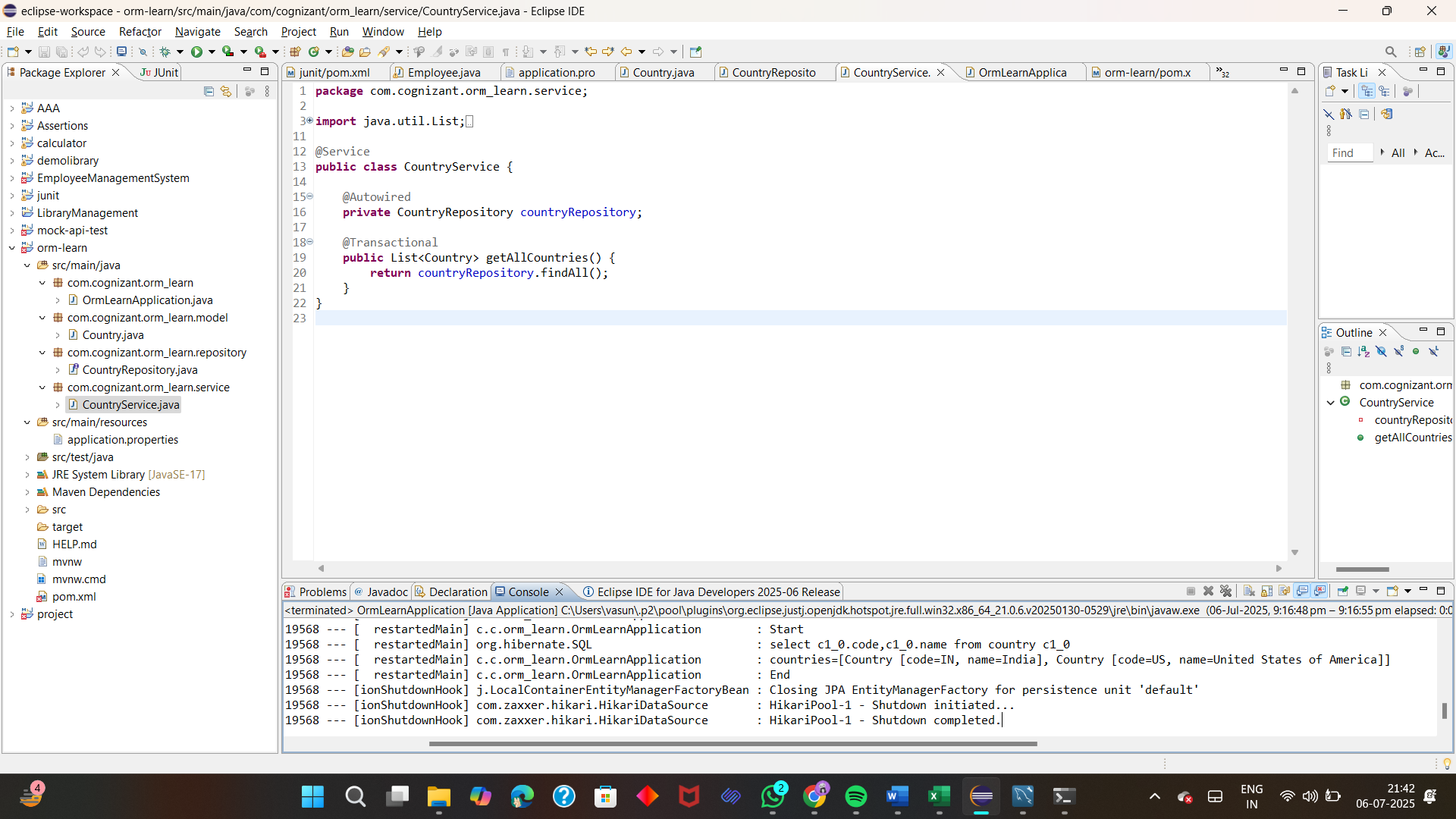
public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OUTPUT:**



**Hands on 4**

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

**SOLUTION:**

**MYSQL WORKBENCH:**

**MYSQL SCHEMA:**

mysql -u root -p

create schema jpadb;

**CREATE TABLE:**

CREATE DATABASE jpadb;

USE jpadb;

CREATE TABLE employee (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

department VARCHAR(100)

);

INSERT INTO employee(name, department) VALUES ('John', 'HR'), ('Alice', 'Finance');

**application.properties.xml**

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

spring.datasource.username=root

spring.datasource.password=Vasundhara@123

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

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

**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.example</groupId>

<artifactId>jpa-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jpa-demo</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>17</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>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>

**JpaDemoApplication.java**

**package** com.example.jpa\_demo;

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

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** com.example.jpa\_demo.model.Employee;

**import** com.example.jpa\_demo.repository.EmployeeRepository;

@SpringBootApplication

**public** **class** JpaDemoApplication **implements** CommandLineRunner {

@Autowired

**private** EmployeeRepository employeeRepository;

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(JpaDemoApplication.**class**, args);

}

@Override

**public** **void** run(String... args) {

employeeRepository.save(**new** Employee("John", "HR"));

employeeRepository.save(**new** Employee("Alice", "Finance"));

System.***out***.println("All Employees:");

employeeRepository.findAll().forEach(emp ->

System.***out***.println(emp.getId() + " | " + emp.getName() + " | " + emp.getDepartment())

);

}

}

**Employee.java**

**package** com.example.jpa\_demo.model;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

@Entity

**public** **class** Employee {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Integer id;

**private** String name;

**private** String department;

// Default constructor is required by JPA

**public** Employee() {

}

// Parameterized constructor for easy object creation

**public** Employee(String name, String department) {

**this**.name = name;

**this**.department = department;

}

// Getters and Setters

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getDepartment() {

**return** department;

}

**public** **void** setDepartment(String department) {

**this**.department = department;

}

}

**EmployeeRepository.java**

**package** com.example.jpa\_demo.repository;

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

**import** com.example.jpa\_demo.model.Employee;

**public** **interface** EmployeeRepository **extends** JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

package com.example.jpa\_demo.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.example.jpa\_demo.model.Employee;

import com.example.jpa\_demo.repository.EmployeeRepository;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repository;

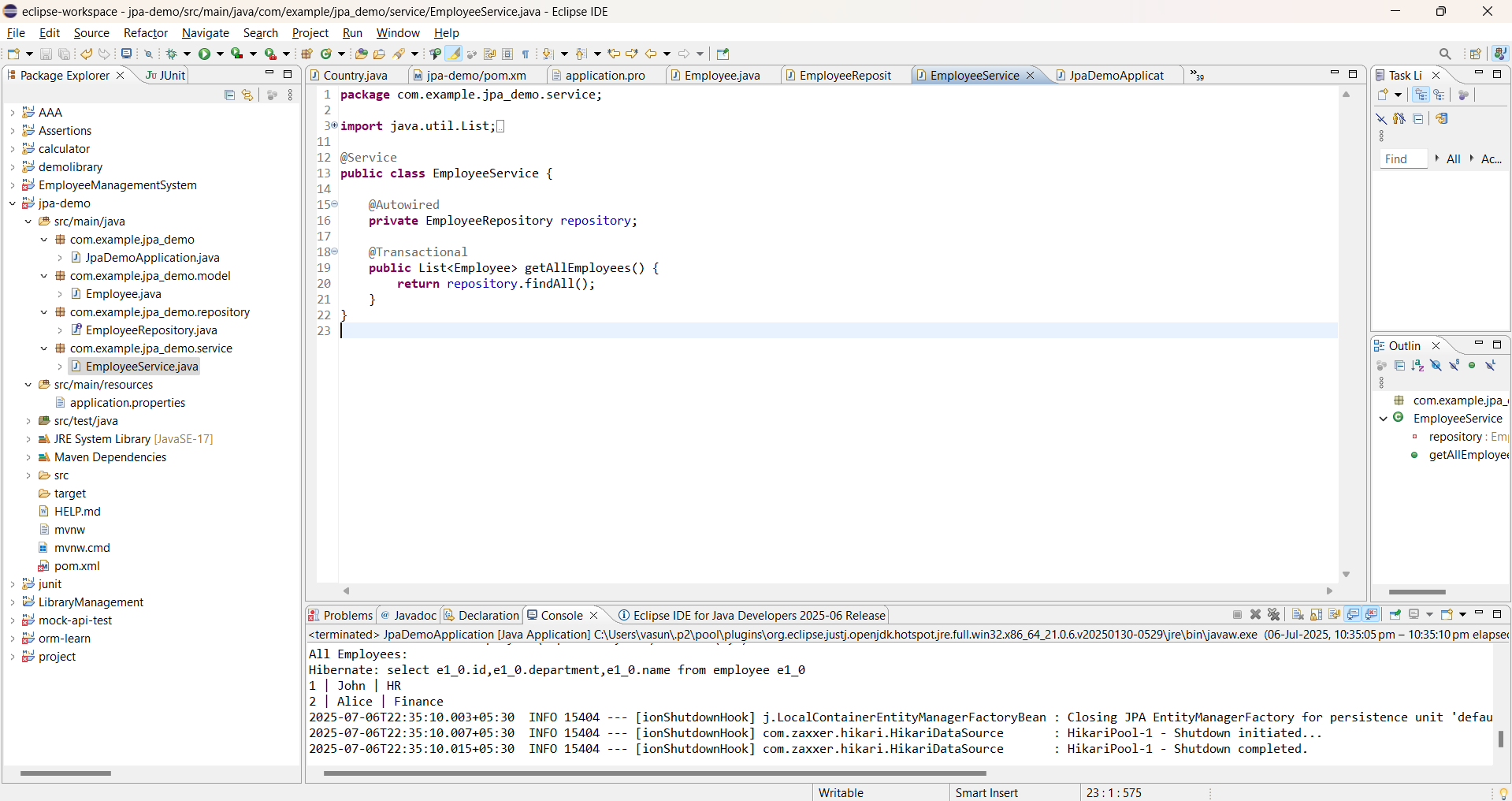
@Transactional

public List<Employee> getAllEmployees() {

return repository.findAll();

}

}  
**OUTPUT:**



**Hands On -5**

**Implement services for managing Country**

**Retrieve Country by Code**

**Add new Country**

**Update Country**

**Delete Country**

**Find List of Countries matching a partial country name**

**SOLUTION:**

**CountryService.java**

package com.cognizant.orm\_learn;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

@Transactional

public Country findCountryByCode(String code) throws Exception {

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

if (result.isEmpty()) {

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

}

return result.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);

}

@Transactional

public List<Country> findCountriesByNameContaining(String name) {

return countryRepository.findByNameContainingIgnoreCase(name);

}

}

**CountryRepository.java**

package com.cognizant.orm\_learn;

import java.util.List;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String name); // ✅ This is required for the search

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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);

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

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

LOGGER.info("Inside main");

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

testGetAllCountries(countryService); // ✅ List all

testAddCountry(countryService); // ✅ Add

testFindCountryByCode(countryService); // ✅ Find by code

testUpdateCountry(countryService); // ✅ Update

testDeleteCountry(countryService); // ✅ Delete

testFindCountriesByNameContaining(countryService); // ✅ Search by name

}

private static void testGetAllCountries(CountryService countryService) {

LOGGER.info("Start testGetAllCountries");

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

for (Country country : countries) {

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

}

LOGGER.info("End testGetAllCountries");

}

private static void testFindCountryByCode(CountryService countryService) throws Exception {

LOGGER.info("Start testFindCountryByCode");

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

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

LOGGER.info("End testFindCountryByCode");

}

private static void testAddCountry(CountryService countryService) {

LOGGER.info("Start testAddCountry");

Country country = new Country();

country.setCode("JP");

country.setName("Japan");

countryService.addCountry(country);

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

LOGGER.info("End testAddCountry");

}

private static void testUpdateCountry(CountryService countryService) throws Exception {

LOGGER.info("Start testUpdateCountry");

countryService.updateCountry("IN", "Bharat");

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

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

LOGGER.info("End testUpdateCountry");

}

private static void testDeleteCountry(CountryService countryService) {

LOGGER.info("Start testDeleteCountry");

countryService.deleteCountry("JP"); // Deleting the country we added

LOGGER.info("Deleted country with code: JP");

LOGGER.info("End testDeleteCountry");

}

private static void testFindCountriesByNameContaining(CountryService countryService) {

LOGGER.info("Start testFindCountriesByNameContaining");

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

for (Country country : countries) {

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

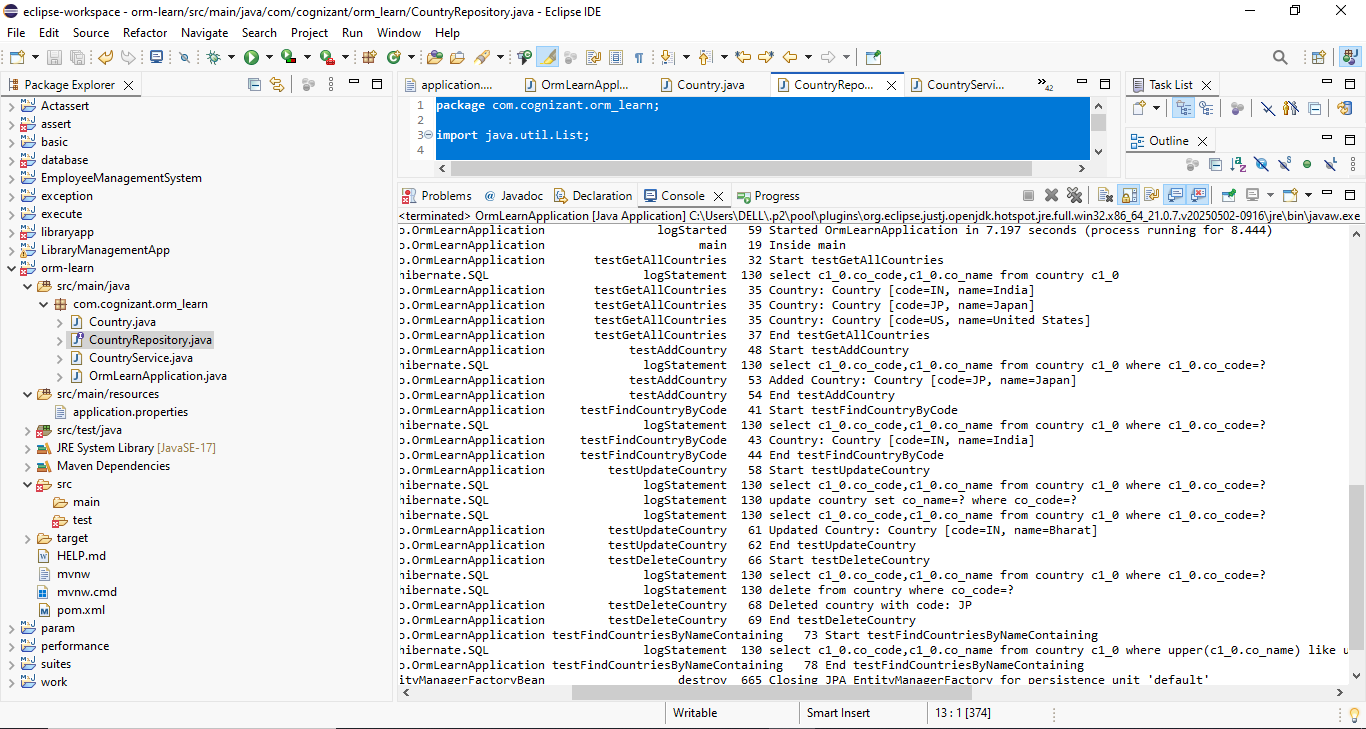
}

LOGGER.info("End testFindCountriesByNameContaining");

}

}

**OUTPUT:**



**Hands On 6**

**Find a country based on country code**

**SOLUTION:**

**CountryNotFoundException.java**

package com.cognizant.orm\_learn.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**CountryService.java**

package com.cognizant.orm\_learn;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.cognizant.orm\_learn.exception.CountryNotFoundException;

import jakarta.transaction.Transactional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (result.isEmpty()) {

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

}

return result.get();

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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.orm\_learn.exception.CountryNotFoundException;

@SpringBootApplication

public class OrmLearnApplication {

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

public static void main(String[] args) {

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

LOGGER.info("Inside main");

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

try {

// ✅ Fetch country by code

String countryCode = "IN";

Country country = countryService.findCountryByCode(countryCode);

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

} catch (CountryNotFoundException e) {

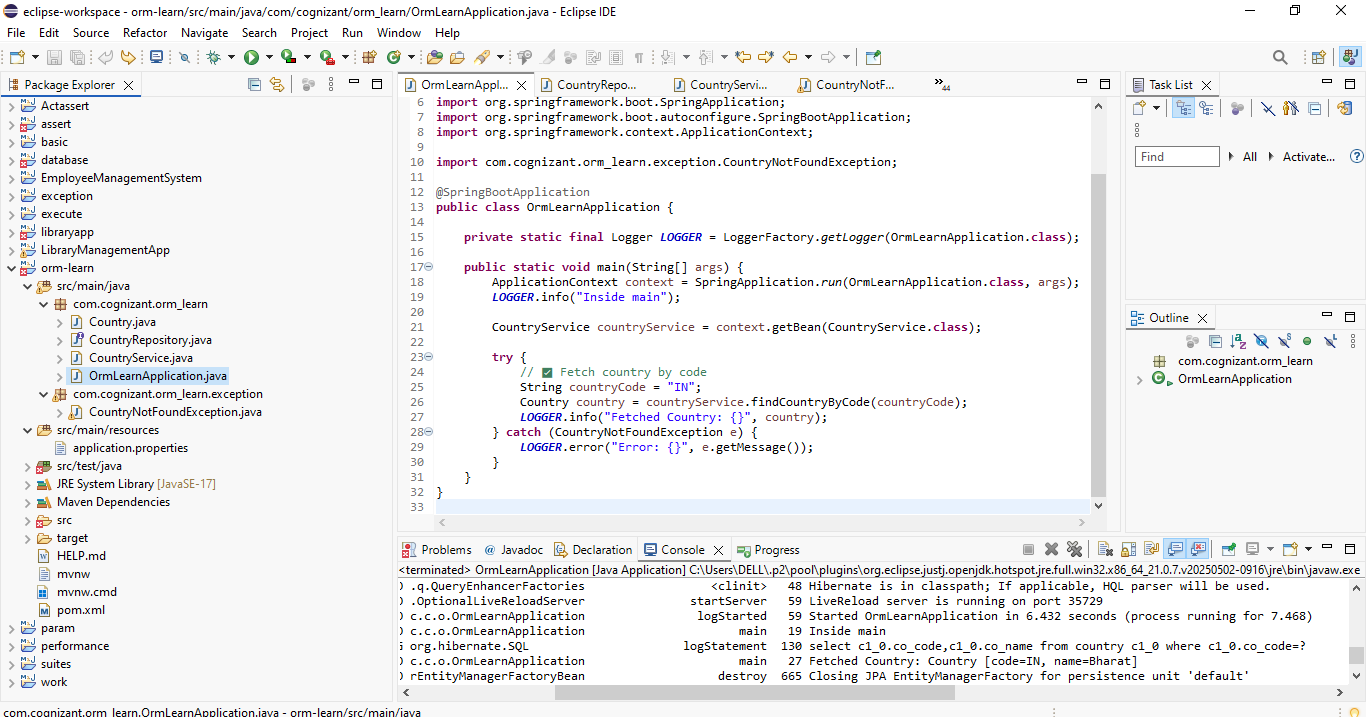
LOGGER.error("Error: {}", e.getMessage());

}

}

}

**OUTPUT:**



**Hands On 7**

**Add a new country**

**SOLUTION:**

**CountryNotFoundException.java**

package com.cognizant.orm\_learn.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**CountryRepository.java:**

package com.cognizant.orm\_learn;

import java.util.List;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

// 🔍 For searching countries by partial name (case-insensitive)

List<Country> findByNameContainingIgnoreCase(String name);

}

**CountryService.java:**

package com.cognizant.orm\_learn;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.cognizant.orm\_learn.exception.CountryNotFoundException;

import jakarta.transaction.Transactional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (result.isEmpty()) {

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

}

return result.get();

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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.orm\_learn.exception.CountryNotFoundException;

@SpringBootApplication

public class OrmLearnApplication {

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

public static void main(String[] args) {

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

LOGGER.info("Inside main");

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

testAddCountry(countryService);

}

private static void testAddCountry(CountryService countryService) {

LOGGER.info("Start testAddCountry");

// Step 1: Create new country instance

Country newCountry = new Country();

newCountry.setCode("JP");

newCountry.setName("Japan");

// Step 2: Add country to DB

countryService.addCountry(newCountry);

// Step 3: Verify it was added

try {

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End testAddCountry");

}

}

**OUTPUT:**

