COGNIZANT

Digital Nurture 4.0

Deep Skilling - Java FSE

WEEK-3 HANDS ON

By Kaviya P

**1.SPRING DATA JPA HANSON**

**Hands on 1**

**Spring Data JPA - Quick Example**   
  
**Software Pre-requisites**

* MySQL Server 8.0
* MySQL Workbench 8
* Eclipse IDE for Enterprise Java Developers 2019-03 R
* Maven 3.6.2

**Create a Eclipse Project using Spring Initializr**

* Go to <https://start.spring.io/>
* Change Group as “com.cognizant”
* Change Artifact Id as “orm-learn”
* In Options > Description enter "Demo project for Spring Data JPA and Hibernate"
* Click on menu and select "Spring Boot DevTools", "Spring Data JPA" and "MySQL Driver"
* Click Generate and download the project as zip
* Extract the zip in root folder to Eclipse Workspace
* Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"

Create a new schema "ormlearn" in MySQL database. Execute the following commands to open MySQL client and create schema

**Country.java**

**package** com.week3.SpringAndMaven.model;

**import** jakarta.persistence.\*;

@Entity

**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.week3.SpringAndMaven.repository;

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model.Country;

@Repository

**public** **interface** CountryRepository **extends** JpaRepository<Country,String>{

}

**CountryService.java**

**package** com.week3.SpringAndMaven.service;

**import** java.util.List;

**import** javax.management.loading.ClassLoaderRepository;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.repository.CountryRepository;

**import** jakarta.transaction.Transactional;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository countryRepository;

@Transactional

**public** List<Country> getAllCountries() {

**return** countryRepository.findAll();

}

}

**application.xml**

spring.application.name=SpringAndMaven

# 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

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger**{25}** %25M %4L %m%n

# Database

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=Kavi@123PS

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

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

spring.jpa.hibernate.ddl-auto=update

**mysql**

CREATE DATABASE ormlearn;

USE ormlearn;

CREATE TABLE country (

    co\_code VARCHAR(2) PRIMARY KEY,

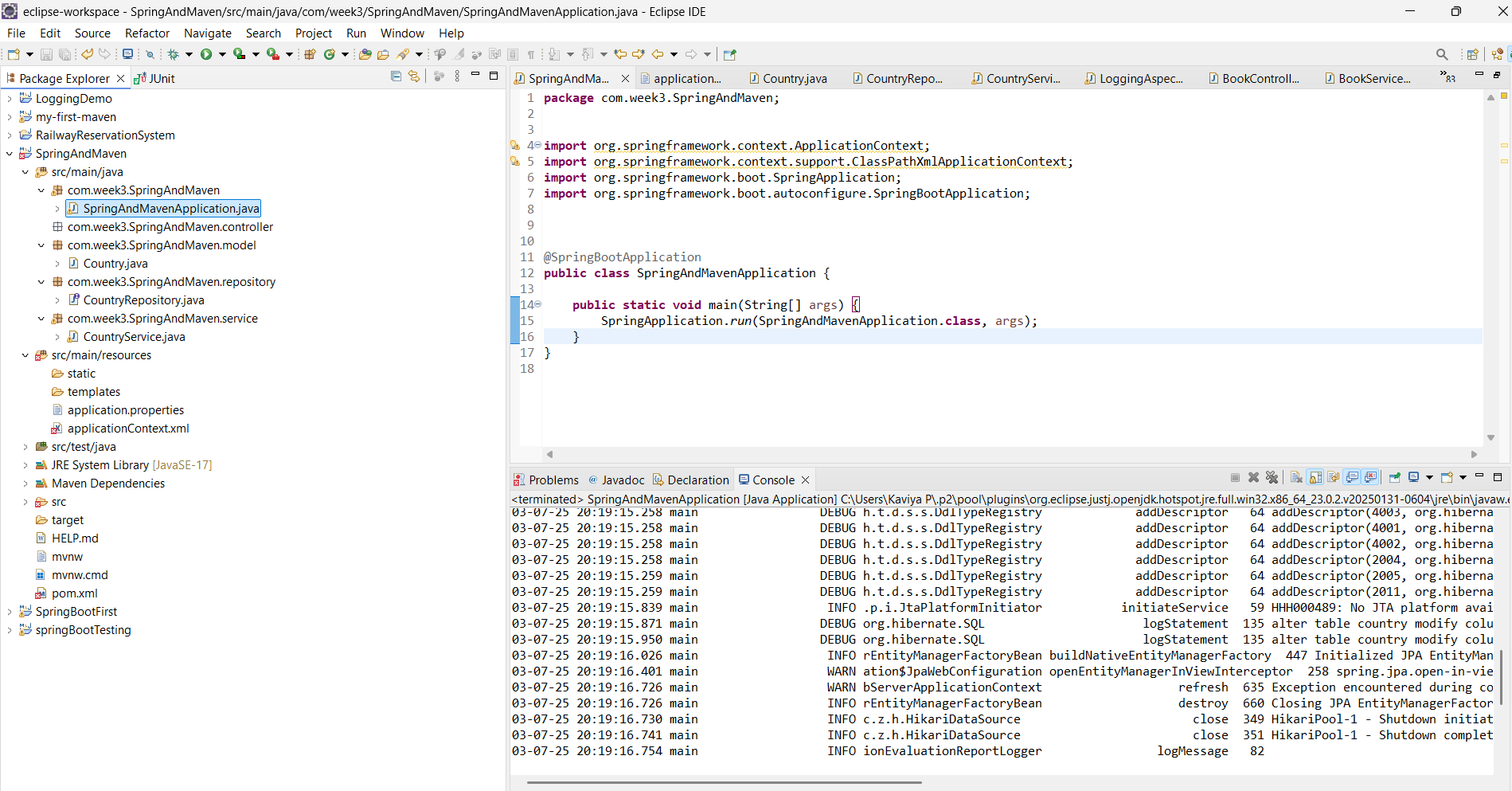
    co\_name VARCHAR(50)

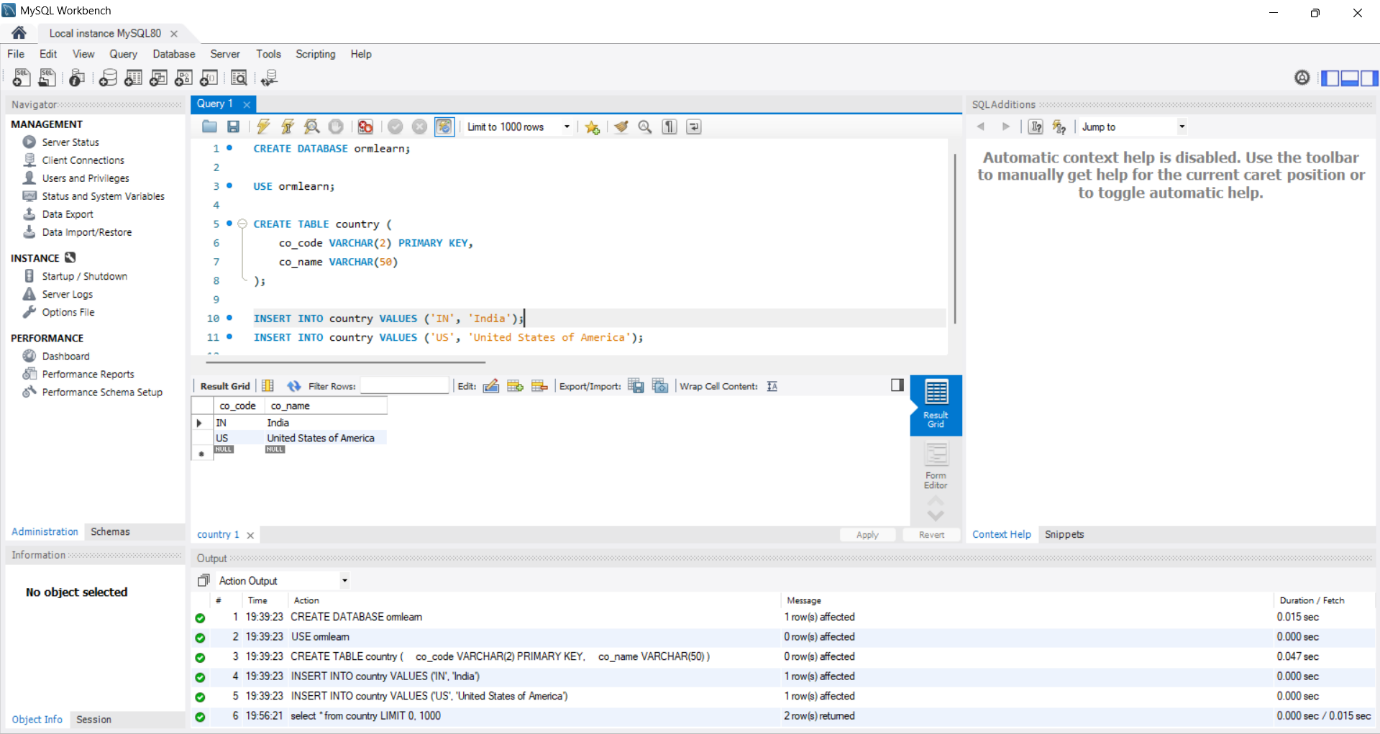
);

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

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

OUTPUT





**EXERCISE 2: Hibernate XML Config implementation walk through**   
  
SME to provide explanation on the sample Hibernate implementation available in the link below:  
https://www.tutorialspoint.com/hibernate/hibernate\_examples.htm  
  
Explanation Topics

1)Explain how object to relational database mapping done in hibernate xml configuration file

Hibernate maps Java classes to database tables using either annotations or XML. In XML, the mapping is done in a .hbm.xml file.

<hibernate-mapping>

  <class name="country.Country" table="COUNTRY">

    <id name="id" column="ID">

      <generator class="native"/>

    </id>

    <property name="name" column="NAME"/>

    <property name="population" column="POPULATION"/>

  </class>

</hibernate-mapping>

2)Explain about following aspects of implementing the end to end operations in Hibernate:

* + SessionFactory

A heavyweight object created once per application.

Used to create Session objects.

Configured using hibernate.cfg.xml.

SessionFactory factory = new Configuration()

    .configure("hibernate.cfg.xml")

    .addResource("country/Country.hbm.xml")

    .buildSessionFactory();

* + Session
* Represents a single-threaded unit of work with the database.
* Created by SessionFactory.

Session session = factory.openSession();

* + Transaction
* Represents a database transaction.
* Ensures operations are atomic (commit/rollback).
  + beginTransaction()
  + Starts a new database transaction.

Transaction tx = session.beginTransaction();

* + commit()
* Saves changes permanently to the database.

tx.commit();

* + rollback()
* Cancels changes if something goes wrong.

tx.rollback();

* + session.save()
  + Persists a new object (insert).

Country c = new Country("India", 1400000000);

session.save(c);

* + session.createQuery().list()
  + Fetches a list of records using HQL.

List<Country> list = session.createQuery("FROM Country", Country.class).list();

* + session.get()
  + Retrieves a single object by its primary key.

Country c = session.get(Country.class, 1);

* + session.delete()
  + Deletes a persistent object from the database.

session.delete(c);

**Hands on 3**

**Hibernate Annotation Config implementation walk through**   
  
SME to provide explanation on the sample Hibernate implementation available in the link below:  
https://www.tutorialspoint.com/hibernate/hibernate\_annotations.htm  
  
Explanation Topics

* Explain how object to relational database mapping done in persistence class file Employee
* Explain about following aspects of implementing the end to end operations in Hibernate:
  + @Entity
  + @Table
  + @Id
  + @GeneratedValue
  + @Column
  + Hibernate Configuration (hibernate.cfg.xml)
    - Dialect
    - Driver
    - Connection URL
    - Username
    - Password

Employee.java

**package** com.kaviyaP.SpringBootFirst.model;

**import** jakarta.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;

// Getters and setters

}

hibernate.cfg.xml

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

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

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

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

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

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

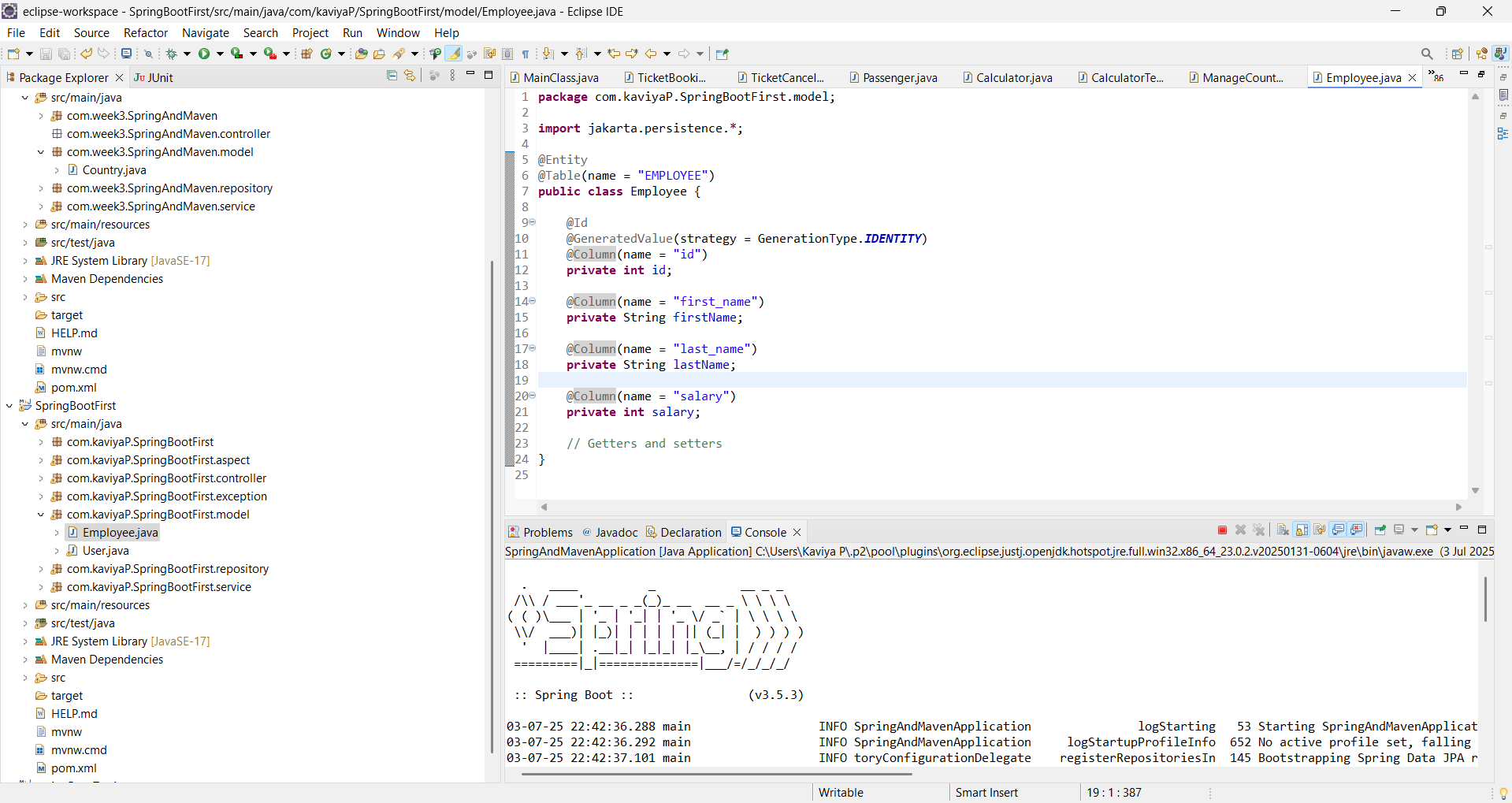
<!-- Entity class -->

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

</session-factory>

</hibernate-configuration>

OUTPUT



**Hands on 4**

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

**-> Java Persistence API (JPA)**

* **JSR 338 Specification** for persisting Java objects in relational DBs.
* **Defines annotations**, transaction boundaries, and ORM mapping rules.
* **No implementation itself**, just the standard interface.
* Implementations include **Hibernate**, **EclipseLink**, etc.

**-> Hibernate**

* **ORM tool** that implements JPA.
* Requires manual:
  + SessionFactory, Session handling
  + Transaction management (begin, commit, rollback)
  + Entity save, fetch, delete manually using session.\*
* Full control, but more boilerplate.

Example Hibernate Code:

**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**

* Another layer on top of JPA + Hibernate.
* Removes boilerplate via Repository pattern.
* Auto-generates queries by method names.
* Handles sessions & transactions **automatically**.

**Example Spring Data JPA Code:**

EmployeeRepository.java

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

*EmployeeService.java*

@Service

**public** **class** EmployeeService {

@Autowired

**private** EmployeeRepository employeeRepository;

@Transactional

**public** **void** addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**Hands on 5**

**Implement services for managing Country**

SpringAndMavenApplication.java

**package** com.week3.SpringAndMaven;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

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

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

}

}

CountryController.java

**package** com.week3.SpringAndMaven.controller;

**import** java.util.List;

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

**import** org.springframework.web.bind.annotation.\*;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.service.CountryService;

@RestController

@RequestMapping("/countries")

**public** **class** CountryController {

@Autowired

**private** CountryService service;

@GetMapping("/{code}")

**public** Country getCountry(@PathVariable String code) {

**return** service.getCountryByCode(code);

}

@PostMapping

**public** Country addCountry(@RequestBody Country country) {

**return** service.addCountry(country);

}

@PutMapping

**public** Country updateCountry(@RequestBody Country country) {

**return** service.updateCountry(country);

}

@DeleteMapping("/{code}")

**public** **void** deleteCountry(@PathVariable String code) {

service.deleteCountry(code);

}

@GetMapping("/search")

**public** List<Country> searchByName(@RequestParam String name) {

**return** service.searchByPartialName(name);

}

@GetMapping

**public** List<Country> getAllCountries() {

**return** service.getAllCountries();

}

}

Country.java

**package** com.week3.SpringAndMaven.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** Country() {}

**public** Country(String code, String name) {

**this**.code = code;

**this**.name = 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;

}

}

**CountryRepository.java**

**package** com.week3.SpringAndMaven.repository;

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

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model.Country;

**import** java.util.List;

@Repository

**public** **interface** CountryRepository **extends** JpaRepository<Country, String> {

@Query("SELECT c FROM Country c WHERE c.name LIKE %:name%")

List<Country> findByPartialName(String name);

}

**CountryService.java**

**package** com.week3.SpringAndMaven.service;

**import** java.util.List;

**import** javax.management.loading.ClassLoaderRepository;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.repository.CountryRepository;

**import** jakarta.transaction.Transactional;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository repository;

**public** Country getCountryByCode(String code) {

**return** repository.findById(code).orElse(**null**);

}

**public** Country addCountry(Country country) {

**return** repository.save(country);

}

**public** Country updateCountry(Country country) {

**return** repository.save(country);

}

**public** **void** deleteCountry(String code) {

repository.deleteById(code);

}

**public** List<Country> searchByPartialName(String name) {

**return** repository.findByPartialName(name);

}

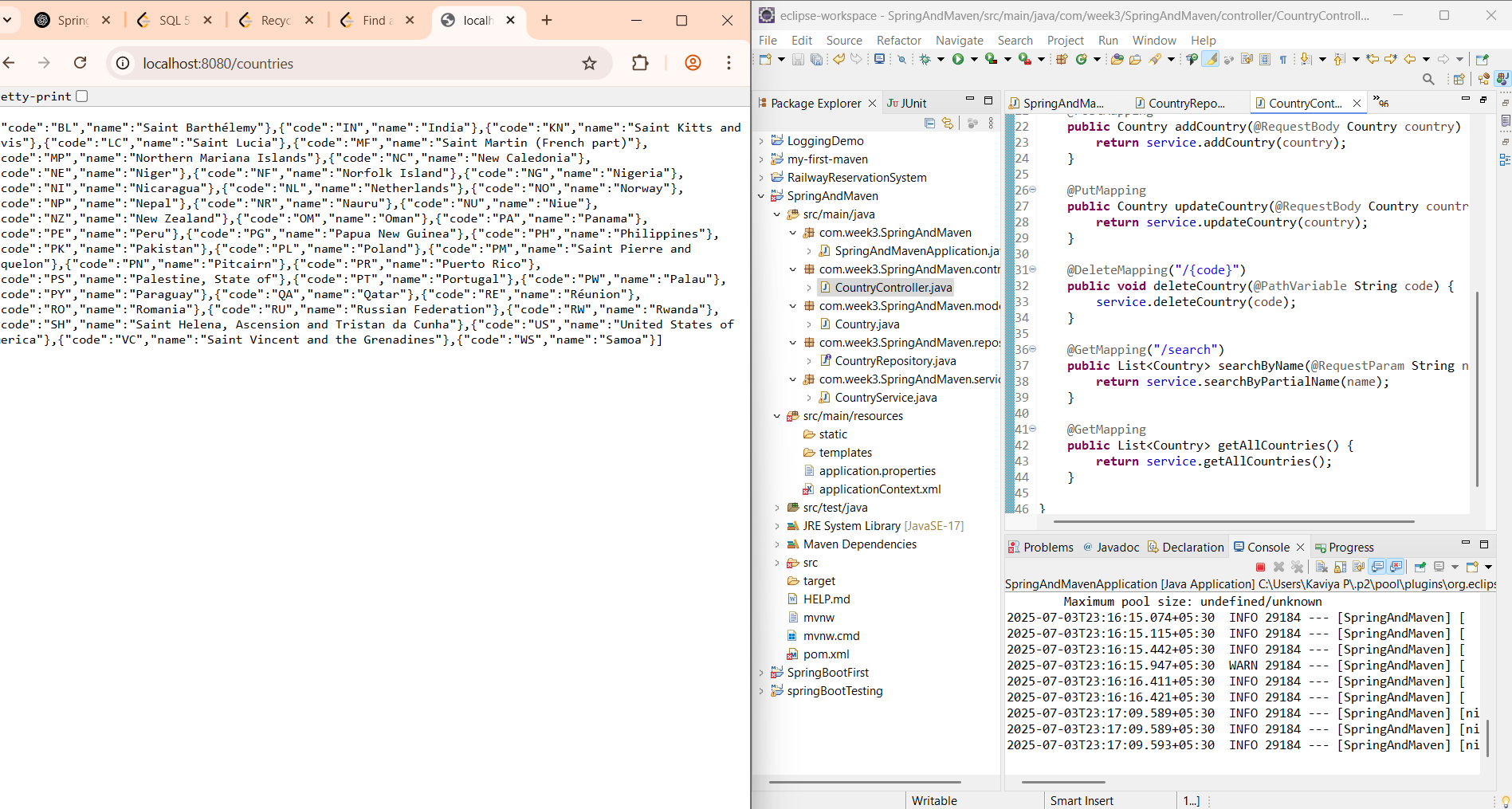
**public** List<Country> getAllCountries() {

**return** repository.findAll();

}

}

Output:



**Hands on 6**

**Find a country based on country code**

**SpringAndMavenApplication.java**

**package** com.week3.SpringAndMaven;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.service.CountryService;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

**import** jakarta.annotation.PostConstruct;

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

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

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

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

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

@Autowired

**private** CountryService countryService;

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

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

}

@PostConstruct

**public** **void** testCountryFetch() {

getCountryByCodeTest();

}

**private** **void** getCountryByCodeTest() {

LOGGER.info("Start");

**try** {

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

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

} **catch** (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

}

**CountryController.java**

**package** com.week3.SpringAndMaven.controller;

**import** java.util.List;

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

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.\*;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.service.CountryService;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

@RestController

@RequestMapping("/countries")

**public** **class** CountryController {

@Autowired

**private** CountryService service;

@GetMapping("{code}")

**public** ResponseEntity<Country> getCountry(@PathVariable String code) {

**try** {

**return** ResponseEntity.ok(service.findCountryByCode(code));

} **catch** (CountryNotFoundException e) {

**return** ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}

}

}

**CountryService.java**

**package** com.week3.SpringAndMaven.service;

**import** java.util.\*;

**import** javax.management.loading.ClassLoaderRepository;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.repository.CountryRepository;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

**import** jakarta.transaction.Transactional;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository repository;

@Transactional

**public** Country findCountryByCode(String countryCode) **throws** CountryNotFoundException {

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

**if** (!result.isPresent()) {

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

}

**return** result.get();

}

}

**CountryNotFoundException.java**

**package** com.week3.SpringAndMaven.service.exception;

**public** **class** CountryNotFoundException **extends** Exception{

**public** CountryNotFoundException(String message) {

**super**(message);

}

}

**pom.xml**

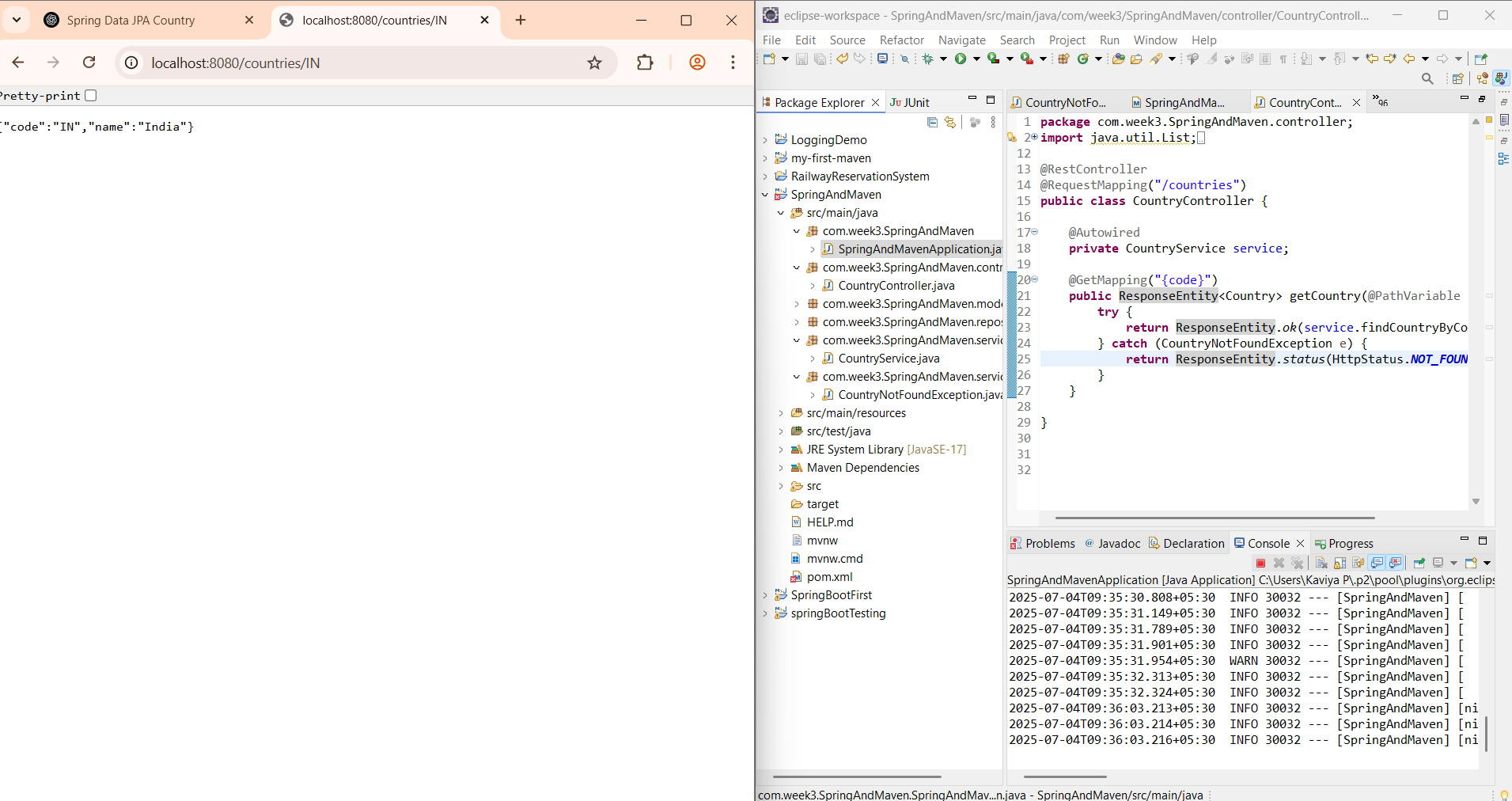
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

**OUTPUT**

****

**Hands on 7**

**Add a new country**

**SpringAndMavenApplication.java**

 **package** com.week3.SpringAndMaven;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.service.CountryService;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

**import** jakarta.annotation.PostConstruct;

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

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

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

@SpringBootApplication

**public** **class** SpringAndMavenApplication {

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

@Autowired

**private** CountryService countryService;

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

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

}

**private** **void** testAddCountry() **throws** CountryNotFoundException {

LOGGER.info("Start");

Country country = **new** Country("VC", "Saint Vincent and the Grenadines");

countryService.addCountry(country);

Country fetched = countryService.findCountryByCode("VC");

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

LOGGER.info("End");

}

}

**CountryController.java**

**package** com.week3.SpringAndMaven.controller;

**import** java.util.List;

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

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.\*;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.service.CountryService;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

@RestController

@RequestMapping("/countries")

**public** **class** CountryController {

@Autowired

**private** CountryService service;

@GetMapping("{code}")

**public** ResponseEntity<Country> getCountry(@PathVariable String code) {

**try** {

**return** ResponseEntity.ok(service.findCountryByCode(code));

} **catch** (CountryNotFoundException e) {

**return** ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}

}

}

**Country.java**

**package** com.week3.SpringAndMaven.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** Country() {}

**public** Country(String code, String name) {

**this**.code = code;

**this**.name = 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;

}

}

**CountryRepository.java**

**package** com.week3.SpringAndMaven.repository;

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

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

**import** org.springframework.stereotype.Repository;

**import** com.week3.SpringAndMaven.model.Country;

**import** java.util.List;

@Repository

**public** **interface** CountryRepository **extends** JpaRepository<Country, String> {

}

**CountryService.java**

**package** com.week3.SpringAndMaven.service;

**import** java.util.\*;

**import** javax.management.loading.ClassLoaderRepository;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.repository.CountryRepository;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

**import** jakarta.transaction.Transactional;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository countryRepository;

@Transactional

**public** Country findCountryByCode(String countryCode) **throws** CountryNotFoundException {

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

**if** (!result.isPresent()) {

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

}

**return** result.get();

}

@Transactional

**public** **void** addCountry(Country country) {

countryRepository.save(country);

}

}

**CountryNotFoundException.java**

**package** com.week3.SpringAndMaven.service.exception;

**public** **class** CountryNotFoundException **extends** Exception{

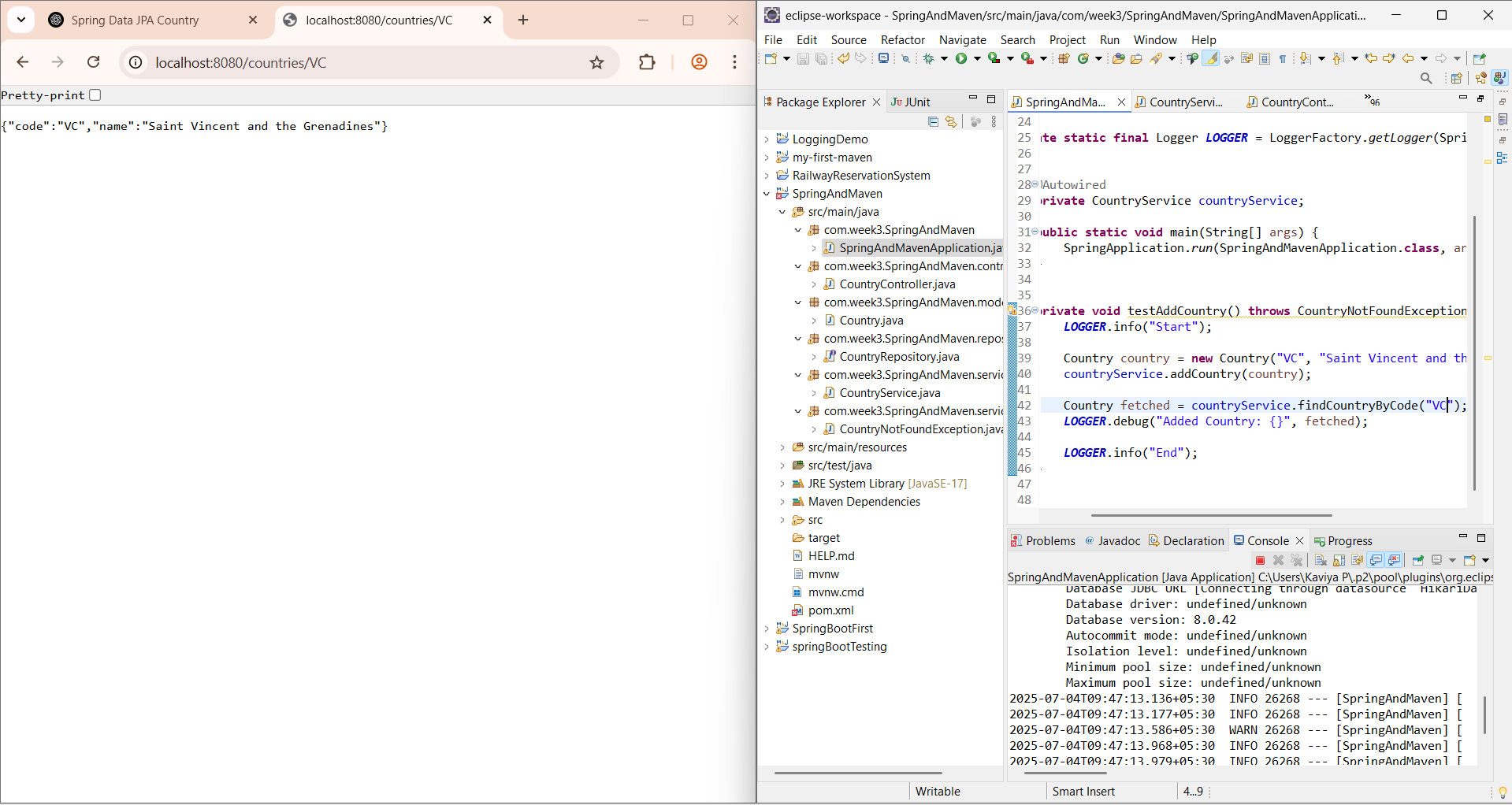
**public** CountryNotFoundException(String message) {

**super**(message);

}

}

**OUTPUT**

****

**Hands on 8**

**Update a country based on code**

**CountryService.java**

**package** com.week3.SpringAndMaven;

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

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.service.CountryService;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

**import** jakarta.annotation.PostConstruct;

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

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

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

@SpringBootApplication

**public** **class** SpringAndMavenApplication **implements** CommandLineRunner{

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

@Autowired

**private** CountryService countryService;

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

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

}

@Override

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

testUpdateCountry(); // or any method like testAddCountry()

}

**private** **void** testUpdateCountry() **throws** CountryNotFoundException {

LOGGER.info("Start");

countryService.updateCountry("VC", "Saint Vincent UPDATED");

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

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

LOGGER.info("End");

}

}

**CountryService.java**

**package** com.week3.SpringAndMaven.service;

**import** java.util.\*;

**import** javax.management.loading.ClassLoaderRepository;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.repository.CountryRepository;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

**import** jakarta.transaction.Transactional;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository countryRepository;

@Transactional

**public** Country findCountryByCode(String countryCode) **throws** CountryNotFoundException {

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

**if** (!result.isPresent()) {

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

}

**return** result.get();

}

@Transactional

**public** **void** addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

**public** **void** updateCountry(String code, String name) **throws** CountryNotFoundException {

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

**if** (!result.isPresent()) {

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

}

Country country = result.get();

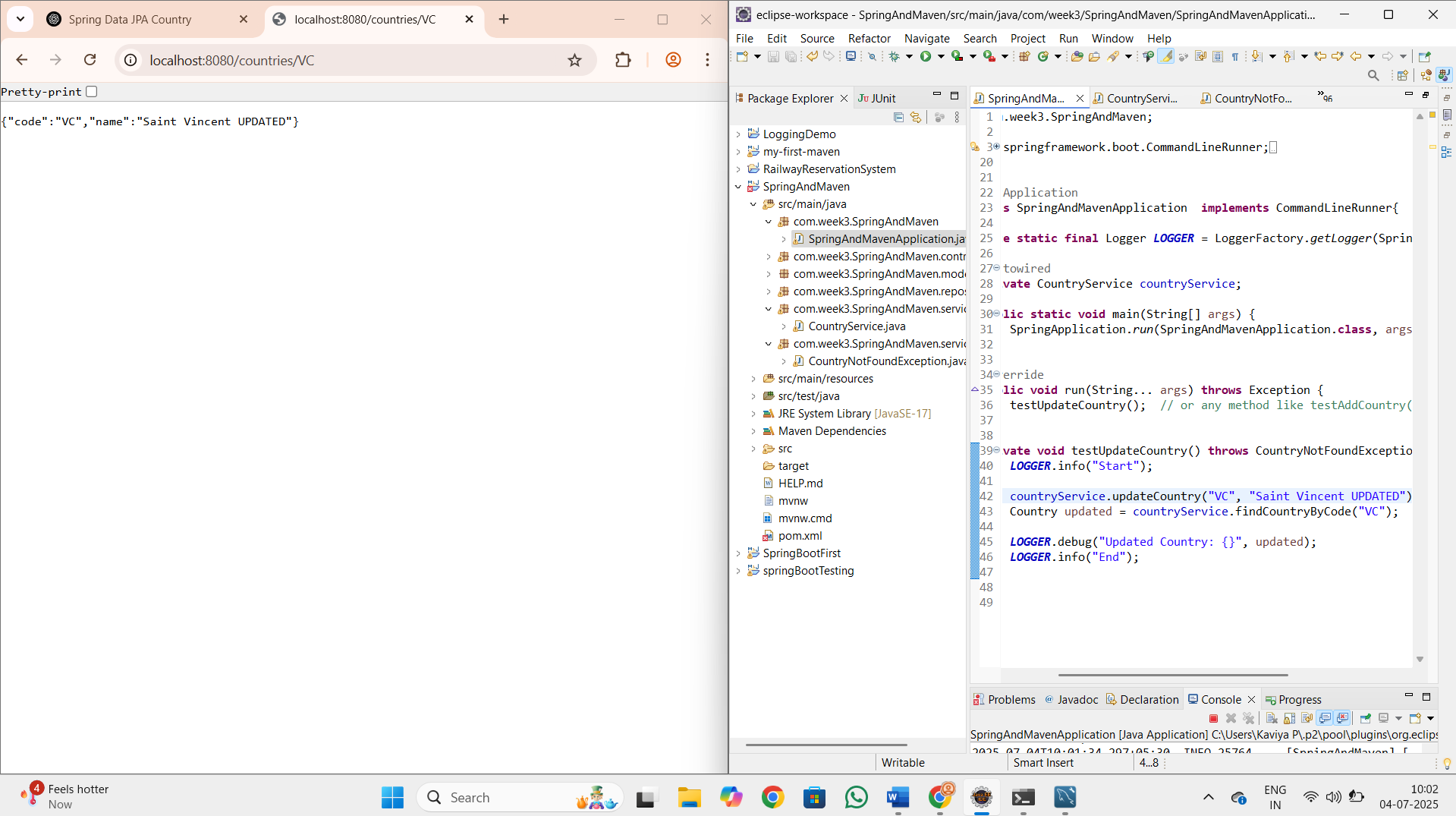
country.setName(name);

countryRepository.save(country);

}

}

**OUTPUT:**

****

**Hands on 9**

**Delete a country based on code**

**SpringAndMavenApplication.java**

 **package** com.week3.SpringAndMaven;

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

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.service.CountryService;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

**import** jakarta.annotation.PostConstruct;

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

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

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

@SpringBootApplication

**public** **class** SpringAndMavenApplication **implements** CommandLineRunner{

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

@Autowired

**private** CountryService countryService;

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

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

}

@Override

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

testDeleteCountry();

}

**private** **void** testDeleteCountry() {

***LOGGER***.info("Start");

// Step 2: Delete the added country

countryService.deleteCountry("vc");

***LOGGER***.debug("VC DELETED");

// Step 3: Try to find the deleted country (should throw exception)

**try** {

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

***LOGGER***.debug("Country still exists: {}", deleted);

} **catch** (CountryNotFoundException e) {

***LOGGER***.debug("Confirmed Deletion: Country not found");

}

***LOGGER***.info("End");

}

}

**CountryService.java**

**package** com.week3.SpringAndMaven.service;

**import** java.util.\*;

**import** javax.management.loading.ClassLoaderRepository;

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

**import** org.springframework.stereotype.Service;

**import** com.week3.SpringAndMaven.model.Country;

**import** com.week3.SpringAndMaven.repository.CountryRepository;

**import** com.week3.SpringAndMaven.service.exception.CountryNotFoundException;

**import** jakarta.transaction.Transactional;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository countryRepository;

@Transactional

**public** Country findCountryByCode(String countryCode) **throws** CountryNotFoundException {

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

**if** (!result.isPresent()) {

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

}

**return** result.get();

}

@Transactional

**public** **void** addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

**public** **void** updateCountry(String code, String name) **throws** CountryNotFoundException {

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

**if** (!result.isPresent()) {

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

}

Country country = result.get();

country.setName(name);

countryRepository.save(country);

}

@Transactional

**public** **void** deleteCountry(String code) {

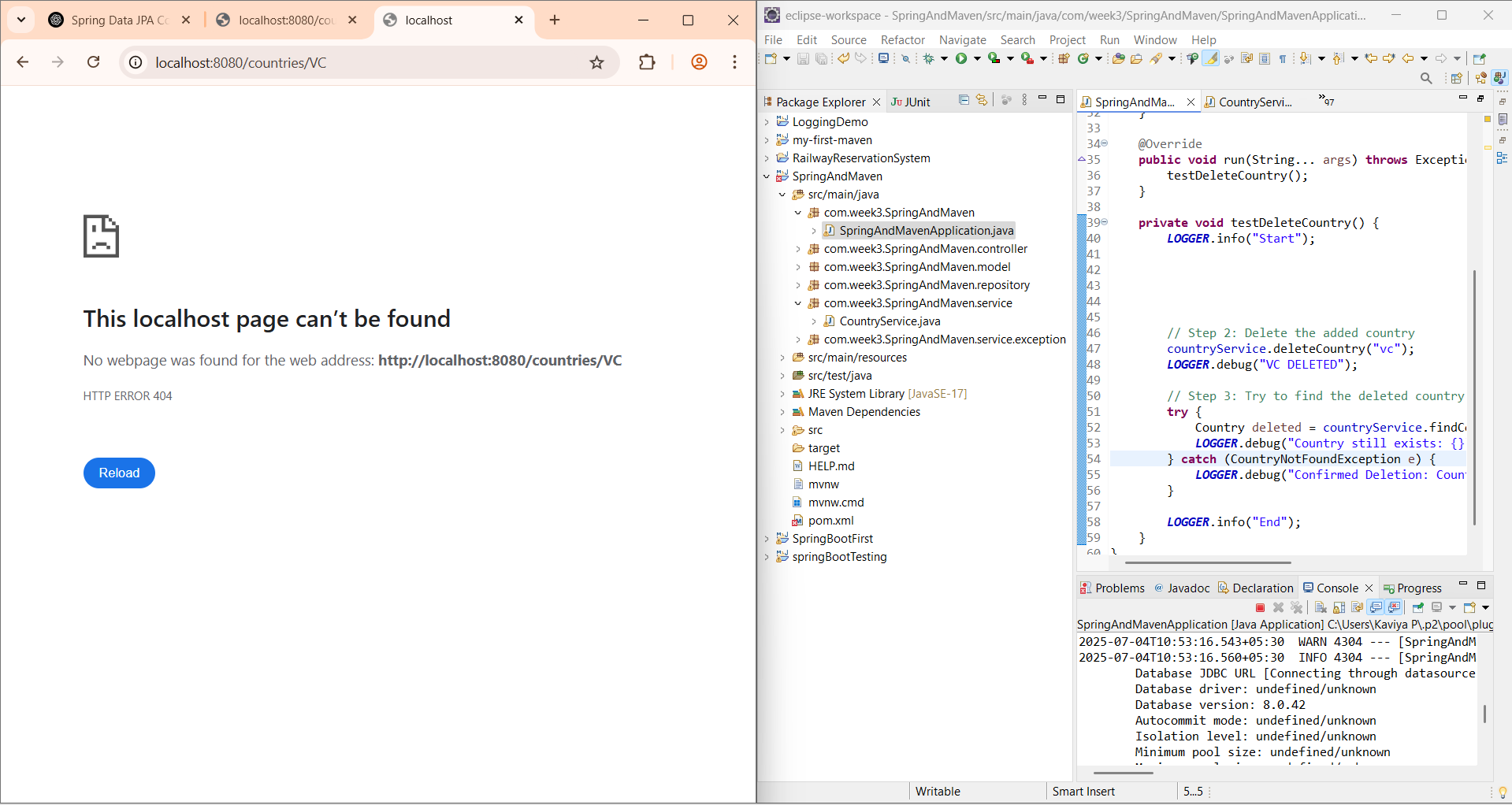
countryRepository.deleteById(code);

}

}

**OUTPUT**

**IT IS SHOWING NO PAGE FOUND BECAUSE WE HAVE DELETED**

****