**Implement Services for Managing Country**

pom.xml

<!-- Spring Boot Starter for JPA -->

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- Spring Boot Starter for Web (if using REST endpoints) -->

<dependency>

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

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

</dependency>

<!-- H2 Database (in-memory) -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

Step 1: Entity - Country.java

package com.example.demo.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Country {

@Id

private String code; // countryCode

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = 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;

}

}

Step 2: Repository - CountryRepository.java

package com.example.demo.repository;

import com.example.demo.model.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String name);

}

Step 3: Service - CountryService.java

package com.example.demo.service;

import com.example.demo.model.Country;

import com.example.demo.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

// Find by code

public Optional<Country> getCountryByCode(String code) {

return countryRepository.findById(code);

}

// Add new

public Country addCountry(Country country) {

return countryRepository.save(country);

}

// Update

public Country updateCountry(Country country) {

return countryRepository.save(country);

}

// Delete

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

// Search by partial name

public List<Country> searchCountriesByName(String partialName) {

return countryRepository.findByNameContainingIgnoreCase(partialName);

}

}

Step 4: Controller - CountryController.java

package com.example.demo.controller;

import com.example.demo.model.Country;

import com.example.demo.service.CountryService;

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

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/{code}")

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

return countryService.getCountryByCode(code);

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

return countryService.addCountry(country);

}

@PutMapping

public Country updateCountry(@RequestBody Country country) {

return countryService.updateCountry(country);

}

@DeleteMapping("/{code}")

public void deleteCountry(@PathVariable String code) {

countryService.deleteCountry(code);

}

@GetMapping("/search")

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

return countryService.searchCountriesByName(name);

}

}

Step 5: Data Population - data.sql

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

INSERT INTO country (code, name) VALUES ('US', 'United States');

INSERT INTO country (code, name) VALUES ('CA', 'Canada');

INSERT INTO country (code, name) VALUES ('AU', 'Australia');

Step 6: Configuration - application.properties

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

**Output :**

{

"code": "IN",

"name": "India"

}

{

"code": "FR",

"name": "France"

}

[

{ "code": "CA", "name": "Canada" },

{ "code": "FR", "name": "France" }

]

**Find a Country Based on Country Code**

Step 1: Create CountryNotFoundException

package com.cognizant.spring\_learn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

Step 2: Update CountryService

package com.cognizant.spring\_learn.service;

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

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

import com.cognizant.spring\_learn.service.exception.CountryNotFoundException;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.Optional;

@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 with code " + countryCode + " not found.");

}

return result.get();

}

}

Step 3: Update OrmLearnApplication.java

package com.cognizant.spring\_learn;

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

import com.cognizant.spring\_learn.service.CountryService;

import com.cognizant.spring\_learn.service.exception.CountryNotFoundException;

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);

countryService = context.getBean(CountryService.class);

getCountryByCodeTest(); // Call the test

}

private static void getCountryByCodeTest() {

LOGGER.info("Start");

try {

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

}

**Output :**

INFO Start

DEBUG Country: Country(code=IN, name=India)

INFO End

**Add a New Country**

Step 1: Add addCountry() Method in CountryService

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

Step 2: Add testAddCountry() in OrmLearnApplication

private static void testAddCountry() {

LOGGER.info("Start");

Country newCountry = new Country();

newCountry.setCode("JP");

newCountry.setName("Japan");

countryService.addCountry(newCountry); // Add new country

try {

Country country = countryService.findCountryByCode("JP"); // Verify it was added

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

testAddCountry(); // Call the new test

}

**Console Output:**

INFO Start

DEBUG Added Country: Country(code=JP, name=Japan)

INFO End

**Demonstrate Implementation of Query Methods in Spring Data JPA**

Entity: Country

@Entity

public class Country {

@Id

private String code;

private String name;

private LocalDate independenceDate; // Example date field

}

Repository: CountryRepository

package com.example.demo.repository;

import com.example.demo.model.Country;

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

import java.time.LocalDate;

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

// Contains (case-insensitive)

List<Country> findByNameContainingIgnoreCase(String name);

// Starts with

List<Country> findByNameStartingWith(String prefix);

// Greater than a specific date

List<Country> findByIndependenceDateAfter(LocalDate date);

// Between two dates

List<Country> findByIndependenceDateBetween(LocalDate start, LocalDate end);

// Top N records

List<Country> findTop3ByOrderByNameAsc();

}

**Example Usage in Service or Test**

countryRepository.findByNameContainingIgnoreCase("in");

**Output :**

[

{ "code": "IN", "name": "India" },

{ "code": "UG", "name": "Uganda" },

{ "code": "UK", "name": "United Kingdom" }

]

**Demonstrate Implementation of O/R Mapping**

Entity: Country.java

@Entity

public class Country {

@Id

private String code;

private String name;

@OneToMany(mappedBy = "country", fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private List<City> cities;

// Getters & Setters

}

Entity: City.java

@Entity

public class City {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "country\_code")

private Country country;

// Getters & Setters

}

**Example: Fetch a Country and Its Cities**

Country country = countryRepository.findById("IN").get();

System.out.println("Country: " + country.getName());

for (City city : country.getCities()) {

System.out.println("City: " + city.getName());

}

**Console Output:**

Country: India

City: Delhi

City: Mumbai

City: Bangalore

**Demonstrate Writing Hibernate Query Language (HQL) and Native Query**

Country.java

@Entity

public class Country {

@Id

private String code;

private String name;

@OneToMany(mappedBy = "country", fetch = FetchType.LAZY)

private List<City> cities;

}

City.java

@Entity

public class City {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@ManyToOne

@JoinColumn(name = "country\_code")

private Country country;

}

Repository: CountryRepository.java

public interface CountryRepository extends JpaRepository<Country, String> {

// Simple HQL query

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

Country findByNameHQL(@Param("name") String name);

// Using fetch to load country with cities

@Query("SELECT c FROM Country c LEFT JOIN FETCH c.cities WHERE c.code = :code")

Country findWithCities(@Param("code") String code);

// Aggregate function: count cities in a country

@Query("SELECT COUNT(ci) FROM City ci WHERE ci.country.code = :code")

Long countCitiesInCountry(@Param("code") String code);

}

**Output for HQL Queries:**

findByNameHQL("India")

{ "code": "IN", "name": "India" }

findWithCities("IN")

{

"code": "IN",

"name": "India",

"cities": [

{ "id": 1, "name": "Delhi" },

{ "id": 2, "name": "Mumbai" }

]

}