Implement services for managing Country

An application requires for features to be implemented with regards to country. These features needs to be supported by implementing them as service using Spring Data JPA.

- Find a country based on country code
- Add new country
- Update country
- Delete country
- Find list of countries matching a partial country name

Before starting the implementation of the above features, there are few configuration and data population that needs to be incorporated. Please refer each topic below and implement the same.

Explanation for Hibernate table creation configuration

- Moreover the ddl-auto defines how hibernate behaves if a specific table or column is not present in the database.
    - create - drops existing tables data and structure, then creates new tables
    - validate - check if the table and columns exist or not, throws an exception if a matching table or column is not found
    - update - if a table does not exists, it creates a new table; if a column does not exists, it creates a new column
    - create-drop - creates the table, once all operations are completed, the table is dropped

```
# Hibernate ddl auto (create, create-drop, update, validate)

spring.jpa.hibernate.ddl-auto=validate
```

Populate country table

- Delete all the records in Country table and then use the below script to create the actual list of all countries in our world.

```
insert into country (co_code, co_name) values ("AF", "Afghanistan");

insert into country (co_code, co_name) values ("AL", "Albania");

insert into country (co_code, co_name) values ("DZ", "Algeria");

insert into country (co_code, co_name) values ("AS", "American Samoa");

insert into country (co_code, co_name) values ("AD", "Andorra");
```

```
insert into country (co_code, co_name) values ("AO", "Angola");
insert into country (co_code, co_name) values ("AI", "Anguilla");
insert into country (co_code, co_name) values ("AQ", "Antarctica");
insert into country (co_code, co_name) values ("AG", "Antigua and Barbuda");
insert into country (co_code, co_name) values ("AR", "Argentina");
insert into country (co_code, co_name) values ("AM", "Armenia");
insert into country (co_code, co_name) values ("AW", "Aruba");
insert into country (co_code, co_name) values ("AU", "Australia");
insert into country (co_code, co_name) values ("AT", "Austria");
insert into country (co_code, co_name) values ("AZ", "Azerbaijan");
insert into country (co_code, co_name) values ("BS", "Bahamas");
insert into country (co_code, co_name) values ("BH", "Bahrain");
insert into country (co_code, co_name) values ("BD", "Bangladesh");
insert into country (co_code, co_name) values ("BB", "Barbados");
insert into country (co_code, co_name) values ("BY", "Belarus");
insert into country (co_code, co_name) values ("BE", "Belgium");
insert into country (co_code, co_name) values ("BZ", "Belize");
insert into country (co_code, co_name) values ("BJ", "Benin");
insert into country (co_code, co_name) values ("BM", "Bermuda");
insert into country (co_code, co_name) values ("BT", "Bhutan");
insert into country (co_code, co_name) values ("BO", "Bolivia, Plurinational State of");
insert into country (co_code, co_name) values ("BQ", "Bonaire, Sint Eustatius and Saba");
insert into country (co_code, co_name) values ("BA", "Bosnia and Herzegovina");
insert into country (co_code, co_name) values ("BW", "Botswana")
insert into country (co_code, co_name) values ("BR", "Brazil");
insert into country (co_code, co_name) values ("IO", "British Indian Ocean Territory");
insert into country (co_code, co_name) values ("BN", "Brunei Darussalam");
insert into country (co_code, co_name) values ("BG", "Bulgaria");
insert into country (co_code, co_name) values ("BF", "Burkina Faso");
insert into country (co_code, co_name) values ("BI", "Burundi");
insert into country (co_code, co_name) values ("KH", "Cambodia");
```

```
insert into country (co_code, co_name) values ("CM", "Cameroon");
insert into country (co_code, co_name) values ("CA", "Canada");
insert into country (co_code, co_name) values ("CV", "Cape Verde");
insert into country (co_code, co_name) values ("KY", "Cayman Islands");
insert into country (co_code, co_name) values ("CF", "Central African Republic");
insert into country (co_code, co_name) values ("TD", "Chad");
insert into country (co_code, co_name) values ("CL", "Chile");
insert into country (co_code, co_name) values ("CN", "China");
insert into country (co_code, co_name) values ("CX", "Christmas Island");
insert into country (co_code, co_name) values ("CC", "Cocos (Keeling) Islands");
insert into country (co_code, co_name) values ("CO", "Colombia");
insert into country (co_code, co_name) values ("KM", "Comoros");
insert into country (co_code, co_name) values ("CG", "Congo");
insert into country (co_code, co_name) values ("CD", "Congo, the Democratic Republic of the");
insert into country (co_code, co_name) values ("CK", "Cook Islands");
insert into country (co_code, co_name) values ("CR", "Costa Rica");
insert into country (co_code, co_name) values ("HR", "Croatia");
```

1. Exception Class – CountryNotFoundException

```
package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {
    public CountryNotFoundException(String message) {
        super(message);
    }
}
```

2. Service Class – CountryService

```
package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;

@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.isPresent()) {
            throw new CountryNotFoundException("Country with code " + code + " not found");
        }
        return result.get();
    }

    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }

    @Transactional
    public void updateCountry(String code, String name) throws CountryNotFoundException {
        Optional<Country> optionalCountry = countryRepository.findById(code);
        if (!optionalCountry.isPresent()) {
            throw new CountryNotFoundException("Country with code " + code + " not found");
        }
        Country country = optionalCountry.get();
        country.setName(name);
```

```
        countryRepository.save(country);
    }

    @Transactional
    public void deleteCountry(String code) {
        countryRepository.deleteById(code);
    }

    @Transactional
    public List<Country> findByNameContaining(String partialName) {
        return countryRepository.findByNameContainingIgnoreCase(partialName);
    }
}
```

3. Custom Query in Repository

```
package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public interface CountryRepository extends JpaRepository<Country, String> {
    List<Country> findByNameContainingIgnoreCase(String name);
}
```

4. Test Methods in OrmLearnApplication

```
private static void testFindCountryByCode() {
   LOGGER.info("Start testFindCountryByCode");
   try {
      Country country = countryService.findCountryByCode("IN");
      LOGGER.debug("Country: {}", country);
   } catch (CountryNotFoundException e) {
      LOGGER.error("Exception: {}", e.getMessage());
   }
   LOGGER.info("End testFindCountryByCode");
}

private static void testAddCountry() {
   LOGGER.info("Start testAddCountry");
   Country country = new Country();
   country.setCode("XY");
   country.setName("Xyzland");
   countryService.addCountry(country);
   try {
      Country result = countryService.findCountryByCode("XY");
      LOGGER.debug("Added Country: {}", result);
   } catch (CountryNotFoundException e) {
      LOGGER.error("Exception: {}", e.getMessage());
   }
```

```
    LOGGER.info("End testAddCountry");
}

private static void testUpdateCountry() {
    LOGGER.info("Start testUpdateCountry");
    try {
        countryService.updateCountry("XY", "Updated Xyzland");
        Country updated = countryService.findCountryByCode("XY");
        LOGGER.debug("Updated Country: {}", updated);
    } catch (CountryNotFoundException e) {
        LOGGER.error("Exception: {}", e.getMessage());
    }
    LOGGER.info("End testUpdateCountry");
}

private static void testDeleteCountry() {
    LOGGER.info("Start testDeleteCountry");
    countryService.deleteCountry("XY");
    try {
        countryService.findCountryByCode("XY");
    } catch (CountryNotFoundException e) {
        LOGGER.debug("Country deletion verified: {}", e.getMessage());
    }
    LOGGER.info("End testDeleteCountry");
}
```

```
private static void testFindByNameContaining() {
   LOGGER.info("Start testFindByNameContaining");
   List<Country> countries = countryService.findByNameContaining("land");
   LOGGER.debug("Matching countries: {}", countries);
   LOGGER.info("End testFindByNameContaining");
}
```

5. Update main() to Call Tests

```
public static void main(String[] args) {
   ApplicationContext context =
SpringApplication.run(OrmLearnApplication.class, args);
   LOGGER.info("Inside main");

   countryService = context.getBean(CountryService.class);

   testGetAllCountries();
   testFindCountryByCode();
   testAddCountry();
   testUpdateCountry();
   testDeleteCountry();
   testFindByNameContaining();
}
```

Add a new country

- Create new method in CountryService.

@Transactional

public void addCountry(Country country)

- Invoke save() method of repository to get the country added.

countryRepository.save(country)

- Include new testAddCountry() method in OrmLearnApplication. Perform steps below:

o Create new instance of country with a new code and name

o Call countryService.addCountry() passing the country created in the previous step.

o Invoke countryService.findCountryByCode() passing the same code used when adding a new country

o Check in the database if the country is added

Step 1: Add addCountry() in CountryService

```
@Transactional
public void addCountry(Country country) {
   countryRepository.save(country);
}
```

Step 2: Add testAddCountry() Method in OrmLearnApplication

```
private static void testAddCountry() {
   LOGGER.info("Start testAddCountry");

   // Step 1: Create new Country object
   Country newCountry = new Country();
   newCountry.setCode("XY");
   newCountry.setName("Xyzland");

   // Step 2: Call addCountry()
   countryService.addCountry(newCountry);
```

```
    // Step 3: Retrieve and verify using findCountryByCode()
    try {
        Country country = countryService.findCountryByCode("XY");
        LOGGER.debug("Added Country: {}", country);
    } catch (CountryNotFoundException e) {
        LOGGER.error("Exception: {}", e.getMessage());
    }
    LOGGER.info("End testAddCountry");
}
```

Step 3: Call testAddCountry() in main() Method

```
public static void main(String[] args) {
    ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
    LOGGER.info("Inside main");

    countryService = context.getBean(CountryService.class);

    testAddCountry();  // Call add test method
}
```

Step 4: Verify in Database

Open **MySQL Workbench** or MySQL client and run:

```
SELECT * FROM country WHERE co_code = 'XY';
```

EXPECTED OUTPUT:

co_code	co_name
XY	Xyzland

Find a country based on country code

- Create new exception class CountryNotFoundException in com.cognizant.spring-learn.service.exception
- Create new method findCountryByCode() in CountryService with @Transactional annotation
- In findCountryByCode() method, perform the following steps:
  - Method signature

```
@Transactional
public Country findCountryByCode(String countryCode) throws CountryNotFoundException
```

- Get the country based on findById() built in method

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

- From the result, check if a country is found. If not found, throw CountryNotFoundException

```
if (!result.isPresent())
```

- Use get() method to return the country fetched.

```
Country country = result.get();
```

- Include new test method in OrmLearnApplication to find a country based on country code and compare the country name to check if it is valid.

```
private static void getAllCountriesTest() {
    LOGGER.info("Start");
    Country country = countryService.findCountryByCode("IN");
    LOGGER.debug("Country:{}", country);
    LOGGER.info("End");
}
```

- Invoke the above method in main() method and test it.

1. Exception Class

```
package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {
   public CountryNotFoundException(String message) {
      super(message);
   }
}
```

2. Update to CountryService Class

```
package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;
```

```
@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }

    @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();
    }
}
```

3. Update OrmLearnApplication to Test

```
package com.cognizant.ormlearn;
```

```
import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.service.CountryService;
import com.cognizant.ormlearn.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;

import java.util.List;

@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();
        testFindCountryByCode();  // <-- Invoke test for find by code
    }
```

```
    private static void testGetAllCountries() {
        LOGGER.info("Start");
        List<Country> countries = countryService.getAllCountries();
        LOGGER.debug("countries={}", countries);
        LOGGER.info("End");
    }
    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");
    }
}
```

Importance of @Transactional Annotation

* It ensures that database operations are executed within a transaction context.

* Spring opens a Hibernate session for the method and handles commit/rollback based on success or exception.

* Without @Transactional, you may face lazy loading issues or session closure errors when fetching entities.

EXPECTED OUTPUT:

```
12-07-25 10:02:01.123 main    INFO  OrmLearnApplication        main   Inside main
12-07-25 10:02:01.456 main    INFO  OrmLearnApplication testFindCountryByCode  Start testFindCountryByCode
12-07-25 10:02:01.789 main    DEBUG OrmLearnApplication testFindCountryByCode  Country: Country [code=IN, name=India]
12-07-25 10:02:01.789 main    INFO  OrmLearnApplication testFindCountryByCode  Country name is valid.
12-07-25 10:02:01.789 main    INFO  OrmLearnApplication testFindCountryByCode  End testFindCountryByCode
```

SUPERSET ID: 6407636

KANMANI MURUGHAIYAN