**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.

> mysql -u root -p

mysql> create schema ormlearn;

* In orm-learn Eclipse project, open src/main/resources/application.properties and include the below database and log configuration.

# Spring Framework and application log

logging.level.org.springframework=info

logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output

logging.level.org.hibernate.SQL=trace

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

# Log pattern

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

# Database configuration

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

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

* Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456’ command in command line
* Include logs for verifying if main() method is called.

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

}

* Execute the OrmLearnApplication and check in log if main method is called.

SME to walk through the following aspects related to the project created:

* src/main/java - Folder with application code
* src/main/resources - Folder for application configuration
* src/test/java - Folder with code for testing the application
* OrmLearnApplication.java - Walkthrough the main() method.
* Purpose of @SpringBootApplication annotation
* pom.xml
  + Walkthrough all the configuration defined in XML file
  + Open 'Dependency Hierarchy' and show the dependency tree.

**Country table creation**

* Create a new table country with columns for code and name. For sample, let us insert one country with values 'IN' and 'India' in this table.

create table country(co\_code varchar(2) primary key, co\_name varchar(50));

* Insert couple of records into the table

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

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

**Persistence Class - com.cognizant.orm-learn.model.Country**

* Open Eclipse with orm-learn project
* Create new package com.cognizant.orm-learn.model
* Create Country.java, then generate getters, setters and toString() methods.
* Include @Entity and @Table at class level
* Include @Column annotations in each getter method specifying the column name.

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.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

// toString()

}

*Notes:*

* @Entity is an indicator to Spring Data JPA that it is an entity class for the application
* @Table helps in defining the mapping database table
* @Id helps is defining the primary key
* @Column helps in defining the mapping table column

**Repository Class - com.cognizant.orm-learn.CountryRepository**

* Create new package com.cognizant.orm-learn.repository
* Create new interface named CountryRepository that extends JpaRepository<Country, String>
* Define @Repository annotation at class level

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> {

}

**Service Class - com.cognizant.orm-learn.service.CountryService**

* Create new package com.cognizant.orm-learn.service
* Create new class CountryService
* Include @Service annotation at class level
* Autowire CountryRepository in CountryService
* Include new method getAllCountries() method that returns a list of countries.
* Include @Transactional annotation for this method
* In getAllCountries() method invoke countryRepository.findAll() method and return the result

**Testing in OrmLearnApplication.java**

* Include a static reference to CountryService in OrmLearnApplication class

private static CountryService countryService;

* Define a test method to get all countries from service.

private static void testGetAllCountries() {

LOGGER.info("Start");

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

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

LOGGER.info("End");

}

* Modify SpringApplication.run() invocation to set the application context and the CountryService reference from the application context.

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

countryService = context.getBean(CountryService.class);

testGetAllCountries();

* Execute main method to check if data from ormlearn database is retrieved.

**Solution**

**application.properties**

spring.application.name=orm-learn

# Spring Framework and application log

logging.level.org.springframework=info

logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output

logging.level.org.hibernate.SQL=trace

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

# Log pattern

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

# Database configuration

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

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn?useSSL=false&serverTimezone=UTCspring.datasource.username=root

spring.datasource.password=Saty1911%40

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

debug=true

**Country.java**

package com.cognizant.ormlearn.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 = "co\_code")

private String code;

@Column(name = "co\_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;

}

// toString

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**CountryRepository.java**

package com.cognizant.ormlearn.repository;

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> {

}

**CountryService.java**

package com.cognizant.ormlearn.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.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

import org.springframework.transaction.annotation.Transactional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplicatuion.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.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

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);

LOGGER.info("Inside main");

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

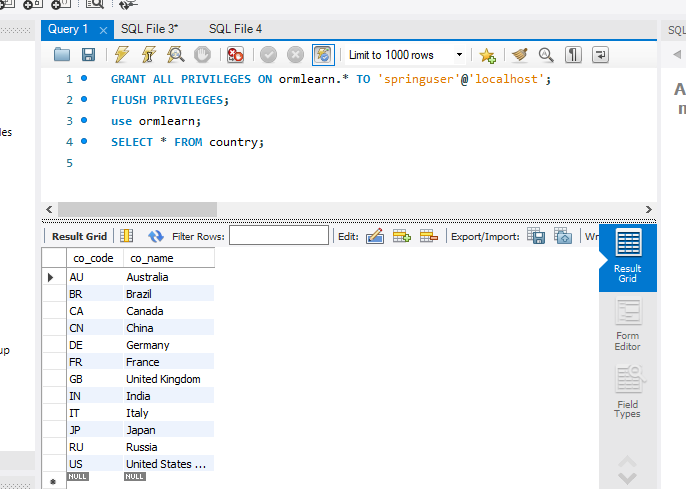
List<Country> countries = countryService.getAllCountries();

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

LOGGER.info("End");

}

}



**Hands on 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

* Explain how object to relational database mapping done in hibernate xml configuration file
* Explain about following aspects of implementing the end to end operations in Hibernate:
  + SessionFactory
  + Session
  + Transaction
  + beginTransaction()
  + commit()
  + rollback()
  + session.save()
  + session.createQuery().list()
  + session.get()
  + session.delete()

**Solution**

## **What is Hibernate?**

Hibernate is like a bridge between our Java objects and our relational database tables. We write and work with normal Java classes, and Hibernate takes care of saving those objects into tables and retrieving them later. It handles all the conversion for us automatically.

## **How does Object to Relational Mapping (ORM) work in Hibernate XML?**

Let’s say we have a Java class like this:

java

CopyEdit

public class Employee {  
 private int id;  
 private String name;  
 private double salary;  
 // getters and setters  
}

We want to map this class to a table in the database called employee. For that, we create an XML mapping file like Employee.hbm.xml:

xml

CopyEdit

<hibernate-mapping>  
 <class name="Employee" table="employee">  
 <id name="id" column="id">  
 <generator class="increment"/>  
 </id>  
 <property name="name" column="name"/>  
 <property name="salary" column="salary"/>  
 </class>  
</hibernate-mapping>

### **Conclusion:**

* We are telling Hibernate: “This Java class Employee should map to the employee table in the database.”
* Each field like name and salary will go into its respective column.
* The id field is our primary key, and Hibernate will auto-increment it.

With this setup, whenever we create an Employee object and tell Hibernate to save it, it will automatically insert it into the correct table.

## **Core Components of Hibernate (in simple terms)**

### **1. SessionFactory**

This is like a setup hub. We configure it once, and it helps us open sessions to connect to the database. We usually keep only one SessionFactory throughout the app.

java

CopyEdit

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

### **2. Session**

A session is like a doorway through which we interact with the database. We can open it, perform operations, and then close it.

java

CopyEdit

Session session = factory.openSession();

### **3. Transaction**

A transaction is like a safe zone where we group multiple database operations together. If everything goes well, we commit it. If anything goes wrong, we can undo it.

## **Lifecycle Methods**

### **beginTransaction()**

Starts a new transaction. It tells Hibernate, “We are starting a set of changes.”

java

CopyEdit

Transaction tx = session.beginTransaction();

### **commit()**

Once all our work is done correctly, we finalize it with commit. It permanently saves the changes.

java

CopyEdit

tx.commit();

### **rollback()**

If something goes wrong during the transaction, we call rollback to undo all changes and leave the database untouched.

java

CopyEdit

tx.rollback();

## **Saving Objects**

java

CopyEdit

session.save(employee);

This line tells Hibernate to save our Employee object into the database.

## **Retrieving Data**

### **createQuery().list()**

java

CopyEdit

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

We ask Hibernate to fetch all records from the employee table. It returns them as a list of Employee objects.

### **session.get()**

java

CopyEdit

Employee emp = session.get(Employee.class, 1);

This gets a specific employee with ID 1. If found, we get the object. If not, we get null.

## **Deleting Records**

java

CopyEdit

session.delete(employee);

This deletes the given employee object from the database.

**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

**Solution**

### **1. Object to Relational Database Mapping in Persistence Class (Employee.java)**

In annotation-based Hibernate configuration, the Java class itself contains metadata that instructs Hibernate on how to map the class to a database table. This eliminates the need for an external XML mapping file (.hbm.xml).

Example:

java

CopyEdit

@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 uses these annotations to:

* Identify which class represents an entity.
* Map the class to a specific table.
* Map fields to columns.
* Specify primary keys and their generation strategy.

This mapping is performed at runtime using Java reflection.

### **2. Explanation of Annotations and Configuration**

#### **a) @Entity**

* Marks the class as a Hibernate entity.
* Tells Hibernate that this class should be mapped to a table in the database.

#### **b) @Table(name = "employee")**

* Specifies the actual table name in the database that this entity is mapped to.
* If this annotation is omitted, Hibernate uses the class name as the table name by default.

#### **c) @Id**

* Indicates the primary key field of the entity.
* Every entity must have a field annotated with @Id.

#### **d) @GeneratedValue**

* Specifies how the primary key should be generated.
* Common strategies include AUTO, IDENTITY, SEQUENCE, and TABLE.
* In the example, GenerationType.IDENTITY means the database will auto-generate the primary key (commonly used with MySQL auto-increment).

#### **e) @Column(name = "column\_name")**

* Maps a Java class field to a column in the database.
* The name attribute specifies the exact column name.
* If not used, Hibernate maps the field using its variable name.

### **3. Hibernate Configuration File (hibernate.cfg.xml)**

This file is used to set up the Hibernate framework, define the database connection, and register annotated classes.

Example configuration:

xml

CopyEdit

<hibernate-configuration>  
 <session-factory>  
 <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/testdb</property>  
 <property name="hibernate.connection.username">root</property>  
 <property name="hibernate.connection.password">password</property>  
   
 <mapping class="com.example.Employee"/>  
 </session-factory>  
</hibernate-configuration>

#### **a) hibernate.dialect**

* Specifies the SQL dialect that Hibernate should use.
* It tells Hibernate how to generate SQL queries specific to the target database (e.g., MySQLDialect, OracleDialect).

#### **b) hibernate.connection.driver\_class**

* Specifies the JDBC driver class used to connect to the database.
* For MySQL 8+, it's usually com.mysql.cj.jdbc.Driver.

#### **c) hibernate.connection.url**

* JDBC connection URL to connect to the database.
* It includes the database type, host, port, and database name.

#### **d) hibernate.connection.username**

* Username used to connect to the database.

#### **e) hibernate.connection.password**

* Password associated with the database user account.

**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA**   
   
Java Persistence API (JPA)

* JSR 338 Specification for persisting, reading and managing data from Java objects
* Does not contain concrete implementation of the specification
* Hibernate is one of the implementation of JPA

Hibernate

* ORM Tool that implements JPA

Spring Data JPA

* Does not have JPA implementation, but reduces boiler plate code
* This is another level of abstraction over JPA implementation provider like Hibernate
* Manages transactions

**Refer code snippets below on how the code compares between Hibernate and Spring Data JPA**  
 **Hibernate**

/\* Method to CREATE an employee in the database \*/

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**  
 EmployeeRespository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java

@Autowire

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Reference Links:**   
 <https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1>   
 <https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>

## **1. Java Persistence API (JPA)**

* JPA is a specification defined under JSR 338.
* It provides a standard way for Java applications to interact with databases using objects.
* It includes interfaces like EntityManager, Query, and annotations like @Entity, @Id, @OneToMany.
* JPA itself is not an implementation; it only defines *how things should be done*, not *how they are done*.
* Popular implementations of JPA include:
  + Hibernate
  + EclipseLink
  + OpenJPA

## **2. Hibernate**

* Hibernate is a **popular ORM (Object-Relational Mapping) framework**.
* It provides a **concrete implementation** of the JPA specification.
* It also includes **non-JPA-specific features** (like native Hibernate Query Language - HQL).
* Developers can use Hibernate:
  + With **pure Hibernate APIs**
  + Or using the **JPA interfaces** provided by the specification.

## **3. Spring Data JPA**

* Spring Data JPA is **not an implementation of JPA**.
* It is a **Spring framework module** that builds **on top of JPA**.
* It provides **repository interfaces** to reduce boilerplate code (like writing save, delete, findById manually).
* It supports **automatic query derivation**, pagination, sorting, and transaction management.
* It works with any JPA provider (like Hibernate, EclipseLink), but by default, most Spring Boot apps use **Hibernate** under the hood.

## **4. Code Comparison**

### **Using Hibernate Directly**

java

CopyEdit

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;  
}

#### **Key Points:**

* Manual session management (openSession(), close()).
* Manual transaction handling (beginTransaction(), commit(), rollback()).
* Direct use of Hibernate API.

### **Using Spring Data JPA**

java

CopyEdit

// Repository interface  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}  
  
// Service class  
@Autowired  
private EmployeeRepository employeeRepository;  
  
@Transactional  
public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
}

#### **Key Points:**

* No session or transaction management code written manually.
* Repository layer handles persistence operations (CRUD).
* Spring automatically manages transactions using @Transactional.

## **5. Summary Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| Type | Specification | ORM Framework | Spring Module (Abstraction) |
| Provides Implementation | No | Yes | No |
| Requires Boilerplate | Yes | Yes | No |
| Transactions | Via EntityManager | Via Session/Transaction | Handled automatically with @Transactional |
| Query Mechanism | JPQL | JPQL + HQL | JPQL + Derived Queries |
| Repository Layer | Not Provided | Not Provided | Built-in Repository Interface (JpaRepository) |
| Use in Spring Boot | Not directly | Yes | Yes (commonly used) |

**Hands on 5**

**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 ("BV", "Bouvet Island");

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");

insert into country (co\_code, co\_name) values ("CU", "Cuba");

insert into country (co\_code, co\_name) values ("CW", "Curaçao");

insert into country (co\_code, co\_name) values ("CY", "Cyprus");

insert into country (co\_code, co\_name) values ("CZ", "Czech Republic");

insert into country (co\_code, co\_name) values ("CI", "Côte d'Ivoire");

insert into country (co\_code, co\_name) values ("DK", "Denmark");

insert into country (co\_code, co\_name) values ("DJ", "Djibouti");

insert into country (co\_code, co\_name) values ("DM", "Dominica");

insert into country (co\_code, co\_name) values ("DO", "Dominican Republic");

insert into country (co\_code, co\_name) values ("EC", "Ecuador");

insert into country (co\_code, co\_name) values ("EG", "Egypt");

insert into country (co\_code, co\_name) values ("SV", "El Salvador");

insert into country (co\_code, co\_name) values ("GQ", "Equatorial Guinea");

insert into country (co\_code, co\_name) values ("ER", "Eritrea");

insert into country (co\_code, co\_name) values ("EE", "Estonia");

insert into country (co\_code, co\_name) values ("ET", "Ethiopia");

insert into country (co\_code, co\_name) values ("FK", "Falkland Islands (Malvinas)");

insert into country (co\_code, co\_name) values ("FO", "Faroe Islands");

insert into country (co\_code, co\_name) values ("FJ", "Fiji");

insert into country (co\_code, co\_name) values ("FI", "Finland");

insert into country (co\_code, co\_name) values ("FR", "France");

insert into country (co\_code, co\_name) values ("GF", "French Guiana");

insert into country (co\_code, co\_name) values ("PF", "French Polynesia");

insert into country (co\_code, co\_name) values ("TF", "French Southern Territories");

insert into country (co\_code, co\_name) values ("GA", "Gabon");

insert into country (co\_code, co\_name) values ("GM", "Gambia");

insert into country (co\_code, co\_name) values ("GE", "Georgia");

insert into country (co\_code, co\_name) values ("DE", "Germany");

insert into country (co\_code, co\_name) values ("GH", "Ghana");

insert into country (co\_code, co\_name) values ("GI", "Gibraltar");

insert into country (co\_code, co\_name) values ("GR", "Greece");

insert into country (co\_code, co\_name) values ("GL", "Greenland");

insert into country (co\_code, co\_name) values ("GD", "Grenada");

insert into country (co\_code, co\_name) values ("GP", "Guadeloupe");

insert into country (co\_code, co\_name) values ("GU", "Guam");

insert into country (co\_code, co\_name) values ("GT", "Guatemala");

insert into country (co\_code, co\_name) values ("GG", "Guernsey");

insert into country (co\_code, co\_name) values ("GN", "Guinea");

insert into country (co\_code, co\_name) values ("GW", "Guinea-Bissau");

insert into country (co\_code, co\_name) values ("GY", "Guyana");

insert into country (co\_code, co\_name) values ("HT", "Haiti");

insert into country (co\_code, co\_name) values ("HM", "Heard Island and McDonald Islands");

insert into country (co\_code, co\_name) values ("VA", "Holy See (Vatican City State)");

insert into country (co\_code, co\_name) values ("HN", "Honduras");

insert into country (co\_code, co\_name) values ("HK", "Hong Kong");

insert into country (co\_code, co\_name) values ("HU", "Hungary");

insert into country (co\_code, co\_name) values ("IS", "Iceland");

insert into country (co\_code, co\_name) values ("IN", "India");

insert into country (co\_code, co\_name) values ("ID", "Indonesia");

insert into country (co\_code, co\_name) values ("IR", "Iran, Islamic Republic of");

insert into country (co\_code, co\_name) values ("IQ", "Iraq");

insert into country (co\_code, co\_name) values ("IE", "Ireland");

insert into country (co\_code, co\_name) values ("IM", "Isle of Man");

insert into country (co\_code, co\_name) values ("IL", "Israel");

insert into country (co\_code, co\_name) values ("IT", "Italy");

insert into country (co\_code, co\_name) values ("JM", "Jamaica");

insert into country (co\_code, co\_name) values ("JP", "Japan");

insert into country (co\_code, co\_name) values ("JE", "Jersey");

insert into country (co\_code, co\_name) values ("JO", "Jordan");

insert into country (co\_code, co\_name) values ("KZ", "Kazakhstan");

insert into country (co\_code, co\_name) values ("KE", "Kenya");

insert into country (co\_code, co\_name) values ("KI", "Kiribati");

insert into country (co\_code, co\_name) values ("KP", "Democratic People's Republic of Korea");

insert into country (co\_code, co\_name) values ("KR", "Republic of Korea");

insert into country (co\_code, co\_name) values ("KW", "Kuwait");

insert into country (co\_code, co\_name) values ("KG", "Kyrgyzstan");

insert into country (co\_code, co\_name) values ("LA", "Lao People's Democratic Republic");

insert into country (co\_code, co\_name) values ("LV", "Latvia");

insert into country (co\_code, co\_name) values ("LB", "Lebanon");

insert into country (co\_code, co\_name) values ("LS", "Lesotho");

insert into country (co\_code, co\_name) values ("LR", "Liberia");

insert into country (co\_code, co\_name) values ("LY", "Libya");

insert into country (co\_code, co\_name) values ("LI", "Liechtenstein");

insert into country (co\_code, co\_name) values ("LT", "Lithuania");

insert into country (co\_code, co\_name) values ("LU", "Luxembourg");

insert into country (co\_code, co\_name) values ("MO", "Macao");

insert into country (co\_code, co\_name) values ("MK", "Macedonia, the Former Yugoslav Republic of");

insert into country (co\_code, co\_name) values ("MG", "Madagascar");

insert into country (co\_code, co\_name) values ("MW", "Malawi");

insert into country (co\_code, co\_name) values ("MY", "Malaysia");

insert into country (co\_code, co\_name) values ("MV", "Maldives");

insert into country (co\_code, co\_name) values ("ML", "Mali");

insert into country (co\_code, co\_name) values ("MT", "Malta");

insert into country (co\_code, co\_name) values ("MH", "Marshall Islands");

insert into country (co\_code, co\_name) values ("MQ", "Martinique");

insert into country (co\_code, co\_name) values ("MR", "Mauritania");

insert into country (co\_code, co\_name) values ("MU", "Mauritius");

insert into country (co\_code, co\_name) values ("YT", "Mayotte");

insert into country (co\_code, co\_name) values ("MX", "Mexico");

insert into country (co\_code, co\_name) values ("FM", "Micronesia, Federated States of");

insert into country (co\_code, co\_name) values ("MD", "Moldova, Republic of");

insert into country (co\_code, co\_name) values ("MC", "Monaco");

insert into country (co\_code, co\_name) values ("MN", "Mongolia");

insert into country (co\_code, co\_name) values ("ME", "Montenegro");

insert into country (co\_code, co\_name) values ("MS", "Montserrat");

insert into country (co\_code, co\_name) values ("MA", "Morocco");

insert into country (co\_code, co\_name) values ("MZ", "Mozambique");

insert into country (co\_code, co\_name) values ("MM", "Myanmar");

insert into country (co\_code, co\_name) values ("NA", "Namibia");

insert into country (co\_code, co\_name) values ("NR", "Nauru");

insert into country (co\_code, co\_name) values ("NP", "Nepal");

insert into country (co\_code, co\_name) values ("NL", "Netherlands");

insert into country (co\_code, co\_name) values ("NC", "New Caledonia");

insert into country (co\_code, co\_name) values ("NZ", "New Zealand");

insert into country (co\_code, co\_name) values ("NI", "Nicaragua");

insert into country (co\_code, co\_name) values ("NE", "Niger");

insert into country (co\_code, co\_name) values ("NG", "Nigeria");

insert into country (co\_code, co\_name) values ("NU", "Niue");

insert into country (co\_code, co\_name) values ("NF", "Norfolk Island");

insert into country (co\_code, co\_name) values ("MP", "Northern Mariana Islands");

insert into country (co\_code, co\_name) values ("NO", "Norway");

insert into country (co\_code, co\_name) values ("OM", "Oman");

insert into country (co\_code, co\_name) values ("PK", "Pakistan");

insert into country (co\_code, co\_name) values ("PW", "Palau");

insert into country (co\_code, co\_name) values ("PS", "Palestine, State of");

insert into country (co\_code, co\_name) values ("PA", "Panama");

insert into country (co\_code, co\_name) values ("PG", "Papua New Guinea");

insert into country (co\_code, co\_name) values ("PY", "Paraguay");

insert into country (co\_code, co\_name) values ("PE", "Peru");

insert into country (co\_code, co\_name) values ("PH", "Philippines");

insert into country (co\_code, co\_name) values ("PN", "Pitcairn");

insert into country (co\_code, co\_name) values ("PL", "Poland");

insert into country (co\_code, co\_name) values ("PT", "Portugal");

insert into country (co\_code, co\_name) values ("PR", "Puerto Rico");

insert into country (co\_code, co\_name) values ("QA", "Qatar");

insert into country (co\_code, co\_name) values ("RO", "Romania");

insert into country (co\_code, co\_name) values ("RU", "Russian Federation");

insert into country (co\_code, co\_name) values ("RW", "Rwanda");

insert into country (co\_code, co\_name) values ("RE", "Réunion");

insert into country (co\_code, co\_name) values ("BL", "Saint Barthélemy");

insert into country (co\_code, co\_name) values ("SH", "Saint Helena, Ascension and Tristan da Cunha");

insert into country (co\_code, co\_name) values ("KN", "Saint Kitts and Nevis");

insert into country (co\_code, co\_name) values ("LC", "Saint Lucia");

insert into country (co\_code, co\_name) values ("MF", "Saint Martin (French part)");

insert into country (co\_code, co\_name) values ("PM", "Saint Pierre and Miquelon");

insert into country (co\_code, co\_name) values ("VC", "Saint Vincent and the Grenadines");

insert into country (co\_code, co\_name) values ("WS", "Samoa");

insert into country (co\_code, co\_name) values ("SM", "San Marino");

insert into country (co\_code, co\_name) values ("ST", "Sao Tome and Principe");

insert into country (co\_code, co\_name) values ("SA", "Saudi Arabia");

insert into country (co\_code, co\_name) values ("SN", "Senegal");

insert into country (co\_code, co\_name) values ("RS", "Serbia");

insert into country (co\_code, co\_name) values ("SC", "Seychelles");

insert into country (co\_code, co\_name) values ("SL", "Sierra Leone");

insert into country (co\_code, co\_name) values ("SG", "Singapore");

insert into country (co\_code, co\_name) values ("SX", "Sint Maarten (Dutch part)");

insert into country (co\_code, co\_name) values ("SK", "Slovakia");

insert into country (co\_code, co\_name) values ("SI", "Slovenia");

insert into country (co\_code, co\_name) values ("SB", "Solomon Islands");

insert into country (co\_code, co\_name) values ("SO", "Somalia");

insert into country (co\_code, co\_name) values ("ZA", "South Africa");

insert into country (co\_code, co\_name) values ("GS", "South Georgia and the South Sandwich Islands");

insert into country (co\_code, co\_name) values ("SS", "South Sudan");

insert into country (co\_code, co\_name) values ("ES", "Spain");

insert into country (co\_code, co\_name) values ("LK", "Sri Lanka");

insert into country (co\_code, co\_name) values ("SD", "Sudan");

insert into country (co\_code, co\_name) values ("SR", "Suriname");

insert into country (co\_code, co\_name) values ("SJ", "Svalbard and Jan Mayen");

insert into country (co\_code, co\_name) values ("SZ", "Swaziland");

insert into country (co\_code, co\_name) values ("SE", "Sweden");

insert into country (co\_code, co\_name) values ("CH", "Switzerland");

insert into country (co\_code, co\_name) values ("SY", "Syrian Arab Republic");

insert into country (co\_code, co\_name) values ("TW", "Taiwan, Province of China");

insert into country (co\_code, co\_name) values ("TJ", "Tajikistan");

insert into country (co\_code, co\_name) values ("TZ", "Tanzania, United Republic of");

insert into country (co\_code, co\_name) values ("TH", "Thailand");

insert into country (co\_code, co\_name) values ("TL", "Timor-Leste");

insert into country (co\_code, co\_name) values ("TG", "Togo");

insert into country (co\_code, co\_name) values ("TK", "Tokelau");

insert into country (co\_code, co\_name) values ("TO", "Tonga");

insert into country (co\_code, co\_name) values ("TT", "Trinidad and Tobago");

insert into country (co\_code, co\_name) values ("TN", "Tunisia");

insert into country (co\_code, co\_name) values ("TR", "Turkey");

insert into country (co\_code, co\_name) values ("TM", "Turkmenistan");

insert into country (co\_code, co\_name) values ("TC", "Turks and Caicos Islands");

insert into country (co\_code, co\_name) values ("TV", "Tuvalu");

insert into country (co\_code, co\_name) values ("UG", "Uganda");

insert into country (co\_code, co\_name) values ("UA", "Ukraine");

insert into country (co\_code, co\_name) values ("AE", "United Arab Emirates");

insert into country (co\_code, co\_name) values ("GB", "United Kingdom");

insert into country (co\_code, co\_name) values ("US", "United States");

insert into country (co\_code, co\_name) values ("UM", "United States Minor Outlying Islands");

insert into country (co\_code, co\_name) values ("UY", "Uruguay");

insert into country (co\_code, co\_name) values ("UZ", "Uzbekistan");

insert into country (co\_code, co\_name) values ("VU", "Vanuatu");

insert into country (co\_code, co\_name) values ("VE", "Venezuela, Bolivarian Republic of");

insert into country (co\_code, co\_name) values ("VN", "Viet Nam");

insert into country (co\_code, co\_name) values ("VG", "Virgin Islands, British");

insert into country (co\_code, co\_name) values ("VI", "Virgin Islands, U.S.");

insert into country (co\_code, co\_name) values ("WF", "Wallis and Futuna");

insert into country (co\_code, co\_name) values ("EH", "Western Sahara");

insert into country (co\_code, co\_name) values ("YE", "Yemen");

insert into country (co\_code, co\_name) values ("ZM", "Zambia");

insert into country (co\_code, co\_name) values ("ZW", "Zimbabwe");

insert into country (co\_code, co\_name) values ("AX", "Åland Islands");

Refer subsequent hands on exercises to implement the features related to country.

**Solution**

**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;  
  
import java.util.List;  
  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
 List<Country> findByNameContaining(String name);  
  
}

**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import jakarta.transaction.Transactional;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
 @Transactional  
 public Country getCountry(String code) {  
 return countryRepository.findById(code).orElse(null);  
 }  
  
 @Transactiona

public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
  
 @Transactional  
 public void updateCountry(String code, Country updatedCountry) {  
 Country country = countryRepository.findById(code).orElse(null);  
 if (country != null) {  
 country.setName(updatedCountry.getName());  
 countryRepository.save(country);  
 }  
 }  
  
 @Transactional  
 public void deleteCountry(String code) {  
 countryRepository.deleteById(code);  
 }  
  
 @Transactional  
 public List<Country> findCountriesByName(String name) {  
 return countryRepository.findByNameContaining(name);  
 }  
}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
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;  
  
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);  
 countryService = context.getBean(CountryService.class);  
  
 testGetAllCountries();  
 testFindCountry();  
 testAddCountry();  
 testUpdateCountry();  
 testDeleteCountry();  
 testFindCountriesByName();  
 }  
  
 private static void testGetAllCountries() {  
 LOGGER.info("Start");  
 List<Country> countries = countryService.getAllCountries();  
 LOGGER.debug("countries={}", countries);  
 LOGGER.info("End");  
 }  
 private static void testFindCountry() {  
 LOGGER.info("Start: Find Country");  
 Country country = countryService.getCountry("IN");  
 LOGGER.debug("Country: {}", country);  
 LOGGER.info("End");  
 }  
  
 private static void testAddCountry() {  
 LOGGER.info("Start: Add Country");  
 Country country = new Country();  
 country.setCode("XX");  
 country.setName("TestLand");  
 countryService.addCountry(country);  
 LOGGER.info("End");  
 }  
  
 private static void testUpdateCountry() {  
 LOGGER.info("Start: Update Country");  
 Country updated = new Country();  
 updated.setName("UpdatedLand");  
 countryService.updateCountry("XX", updated);  
 LOGGER.info("End");  
 }  
  
 private static void testDeleteCountry() {  
 LOGGER.info("Start: Delete Country");  
 countryService.deleteCountry("XX");  
 LOGGER.info("End");  
 }  
  
 private static void testFindCountriesByName() {  
 LOGGER.info("Start: Find Countries by Partial Name");  
 List<Country> countries = countryService.findCountriesByName("land");  
 LOGGER.debug("Matching Countries: {}", countries);  
 LOGGER.info("End");  
 }  
}

**Output**

OrmLearnApplication testGetAllCountries 34 **countries=[Country [code=AD, name=Andorra], Country [code=AE, name=United Arab Emirates], Country [code=AF, name=Afghanistan], Country [code=AG, name=Antigua and Barbuda], Country [code=AI, name=Anguilla], Country [code=AL, name=Albania], Country [code=AM, name=Armenia], Country [code=AO, name=Angola], Country [code=AQ, name=Antarctica], Country [code=AR, name=Argentina], Country [code=AS, name=American Samoa], Country [code=AT, name=Austria], Country [code=AU, name=Australia], Country [code=AW, name=Aruba], Country [code=AX, name=Åland Islands], Country [code=AZ, name=Azerbaijan], Country [code=BA, name=Bosnia and Herzegovina], Country [code=BB, name=Barbados], Country [code=BD, name=Bangladesh], Country [code=BE, name=Belgium], Country [code=BF, name=Burkina Faso], Country [code=BG, name=Bulgaria], Country [code=BH, name=Bahrain], Country [code=BI, name=Burundi], Country [code=BJ, name=Benin], Country [code=BL, name=Saint Barthélemy], Country [code=BM, name=Bermuda], Country [code=BN, name=Brunei Darussalam], Country [code=BO, name=Bolivia, Plurinational State of], Country [code=BQ, name=Bonaire, Sint Eustatius and Saba], Country [code=BR, name=Brazil], Country [code=BS, name=Bahamas], Country [code=BT, name=Bhutan], Country [code=BV, name=Bouvet Island], Country [code=BW, name=Botswana], Country [code=BY, name=Belarus], Country [code=BZ, name=Belize], Country [code=CA, name=Canada], Country [code=CC, name=Cocos (Keeling) Islands], Country [code=CD, name=Congo, the Democratic Republic of the], Country [code=CF, name=Central African Republic], Country [code=CG, name=Congo], Country [code=CH, name=Switzerland], Country [code=CI, name=Côte d'Ivoire], Country [code=CK, name=Cook Islands], Country [code=CL, name=Chile], Country [code=CM, name=Cameroon], Country [code=CN, name=China], Country [code=CO, name=Colombia], Country [code=CR, name=Costa Rica], Country [code=CU, name=Cuba], Country [code=CV, name=Cape Verde], Country [code=CW, name=Curaçao], Country [code=CX, name=Christmas Island], Country [code=CY, name=Cyprus], Country [code=CZ, name=Czech Republic], Country [code=DE, name=Germany], Country [code=DJ, name=Djibouti], Country [code=DK, name=Denmark], Country [code=DM, name=Dominica], Country [code=DO, name=Dominican Republic], Country [code=DZ, name=Algeria], Country [code=EC, name=Ecuador], Country [code=EE, name=Estonia], Country [code=EG, name=Egypt], Country [code=EH, name=Western Sahara], Country [code=ER, name=Eritrea], Country [code=ES, name=Spain], Country [code=ET, name=Ethiopia], Country [code=FI, name=Finland], Country [code=FJ, name=Fiji], Country [code=FK, name=Falkland Islands (Malvinas)], Country [code=FM, name=Micronesia, Federated States of], Country [code=FO, name=Faroe Islands], Country [code=FR, name=France], Country [code=GA, name=Gabon], Country [code=GB, name=United Kingdom], Country [code=GD, name=Grenada], Country [code=GE, name=Georgia], Country [code=GF, name=French Guiana], Country [code=GG, name=Guernsey], Country [code=GH, name=Ghana], Country [code=GI, name=Gibraltar], Country [code=GL, name=Greenland], Country [code=GM, name=Gambia], Country [code=GN, name=Guinea], Country [code=GP, name=Guadeloupe], Country [code=GQ, name=Equatorial Guinea], Country [code=GR, name=Greece], Country [code=GS, name=South Georgia and the South Sandwich Islands], Country [code=GT, name=Guatemala], Country [code=GU, name=Guam], Country [code=GW, name=Guinea-Bissau], Country [code=GY, name=Guyana], Country [code=HK, name=Hong Kong], Country [code=HM, name=Heard Island and McDonald Islands], Country [code=HN, name=Honduras], Country [code=HR, name=Croatia], Country [code=HT, name=Haiti], Country [code=HU, name=Hungary], Country [code=ID, name=Indonesia], Country [code=IE, name=Ireland], Country [code=IL, name=Israel], Country [code=IM, name=Isle of Man], Country [code=IN, name=India], Country [code=IO, name=British Indian Ocean Territory], Country [code=IQ, name=Iraq], Country [code=IR, name=Iran, Islamic Republic of], Country [code=IS, name=Iceland], Country [code=IT, name=Italy], Country [code=JE, name=Jersey], Country [code=JM, name=Jamaica], Country [code=JO, name=Jordan], Country [code=JP, name=Japan], Country [code=KE, name=Kenya], Country [code=KG, name=Kyrgyzstan], Country [code=KH, name=Cambodia], Country [code=KI, name=Kiribati], Country [code=KM, name=Comoros], Country [code=KN, name=Saint Kitts and Nevis], Country [code=KP, name=Democratic People's Republic of Korea], Country [code=KR, name=Republic of Korea], Country [code=KW, name=Kuwait], Country [code=KY, name=Cayman Islands], Country [code=KZ, name=Kazakhstan], Country [code=LA, name=Lao People's Democratic Republic], Country [code=LB, name=Lebanon], Country [code=LC, name=Saint Lucia], Country [code=LI, name=Liechtenstein], Country [code=LK, name=Sri Lanka], Country [code=LR, name=Liberia], Country [code=LS, name=Lesotho], Country [code=LT, name=Lithuania], Country [code=LU, name=Luxembourg], Country [code=LV, name=Latvia], Country [code=LY, name=Libya], Country [code=MA, name=Morocco], Country [code=MC, name=Monaco], Country [code=MD, name=Moldova, Republic of], Country [code=ME, name=Montenegro], Country [code=MF, name=Saint Martin (French part)], Country [code=MG, name=Madagascar], Country [code=MH, name=Marshall Islands], Country [code=MK, name=Macedonia, the Former Yugoslav Republic of], Country [code=ML, name=Mali], Country [code=MM, name=Myanmar], Country [code=MN, name=Mongolia], Country [code=MO, name=Macao], Country [code=MP, name=Northern Mariana Islands], Country [code=MQ, name=Martinique], Country [code=MR, name=Mauritania], Country [code=MS, name=Montserrat], Country [code=MT, name=Malta], Country [code=MU, name=Mauritius], Country [code=MV, name=Maldives], Country [code=MW, name=Malawi], Country [code=MX, name=Mexico], Country [code=MY, name=Malaysia], Country [code=MZ, name=Mozambique], Country [code=NA, name=Namibia], Country [code=NC, name=New Caledonia], Country [code=NE, name=Niger], Country [code=NF, name=Norfolk Island], Country [code=NG, name=Nigeria], Country [code=NI, name=Nicaragua], Country [code=NL, name=Netherlands], Country [code=NO, name=Norway], Country [code=NP, name=Nepal], Country [code=NR, name=Nauru], Country [code=NU, name=Niue], Country [code=NZ, name=New Zealand], Country [code=OM, name=Oman], Country [code=PA, name=Panama], Country [code=PE, name=Peru], Country [code=PF, name=French Polynesia], Country [code=PG, name=Papua New Guinea], Country [code=PH, name=Philippines], Country [code=PK, name=Pakistan], Country [code=PL, name=Poland], Country [code=PM, name=Saint Pierre and Miquelon], Country [code=PN, name=Pitcairn], Country [code=PR, name=Puerto Rico], Country [code=PS, name=Palestine, State of], Country [code=PT, name=Portugal], Country [code=PW, name=Palau], Country [code=PY, name=Paraguay], Country [code=QA, name=Qatar], Country [code=RE, name=Réunion], Country [code=RO, name=Romania], Country [code=RS, name=Serbia], Country [code=RU, name=Russian Federation], Country [code=RW, name=Rwanda], Country [code=SA, name=Saudi Arabia], Country [code=SB, name=Solomon Islands], Country [code=SC, name=Seychelles], Country [code=SD, name=Sudan], Country [code=SE, name=Sweden], Country [code=SG, name=Singapore], Country [code=SH, name=Saint Helena, Ascension and Tristan da Cunha], Country [code=SI, name=Slovenia], Country [code=SJ, name=Svalbard and Jan Mayen], Country [code=SK, name=Slovakia], Country [code=SL, name=Sierra Leone], Country [code=SM, name=San Marino], Country [code=SN, name=Senegal], Country [code=SO, name=Somalia], Country [code=SR, name=Suriname], Country [code=SS, name=South Sudan], Country [code=ST, name=Sao Tome and Principe], Country [code=SV, name=El Salvador], Country [code=SX, name=Sint Maarten (Dutch part)], Country [code=SY, name=Syrian Arab Republic], Country [code=SZ, name=Swaziland], Country [code=TC, name=Turks and Caicos Islands], Country [code=TD, name=Chad], Country [code=TF, name=French Southern Territories], Country [code=TG, name=Togo], Country [code=TH, name=Thailand], Country [code=TJ, name=Tajikistan], Country [code=TK, name=Tokelau], Country [code=TL, name=Timor-Leste], Country [code=TM, name=Turkmenistan], Country [code=TN, name=Tunisia], Country [code=TO, name=Tonga], Country [code=TR, name=Turkey], Country [code=TT, name=Trinidad and Tobago], Country [code=TV, name=Tuvalu], Country [code=TW, name=Taiwan, Province of China], Country [code=TZ, name=Tanzania, United Republic of], Country [code=UA, name=Ukraine], Country [code=UG, name=Uganda], Country [code=UM, name=United States Minor Outlying Islands], Country [code=US, name=United States], Country [code=UY, name=Uruguay], Country [code=UZ, name=Uzbekistan], Country [code=VA, name=Holy See (Vatican City State)], Country [code=VC, name=Saint Vincent and the Grenadines], Country [code=VE, name=Venezuela, Bolivarian Republic of], Country [code=VG, name=Virgin Islands, British], Country [code=VI, name=Virgin Islands, U.S.], Country [code=VN, name=Viet Nam], Country [code=VU, name=Vanuatu], Country [code=WF, name=Wallis and Futuna], Country [code=WS, name=Samoa], Country [code=YE, name=Yemen], Country [code=YT, name=Mayotte], Country [code=ZA, name=South Africa], Country [code=ZM, name=Zambia], Country [code=ZW, name=Zimbabwe]]**

10-07-25 16:03:22.311 restartedMain INFO c.c.o.OrmLearnApplication testGetAllCountries 35 End 10-07-25 16:03:22.311 restartedMain INFO c.c.o.OrmLearnApplication testFindCountry 38 Start:

Find Country 10-07-25 16:03:22.330 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 16:03:22.340 restartedMain DEBUG c.c.o.OrmLearnApplication testFindCountry 40 Country: Country [code=IN, name=India] 10-07-25 16:03:22.340 restartedMain INFO c.c.o.OrmLearnApplication testFindCountry 41 End 10-07-25 16:03:22.340 restartedMain INFO c.c.o.OrmLearnApplication testAddCountry 45 Start: Add Country 10-07-25 16:03:22.355 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 16:03:22.379 restartedMain DEBUG org.hibernate.SQL logStatement 135 insert into country (co\_name,co\_code) values (?,?) 10-07-25 16:03:22.394 restartedMain INFO c.c.o.OrmLearnApplication testAddCountry 50 End 10-07-25 16:03:22.394 restartedMain INFO c.c.o.OrmLearnApplication testUpdateCountry 54 Start: Update Country 10-07-25 16:03:22.403 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 16:03:22.435 restartedMain DEBUG org.hibernate.SQL logStatement 135 update country set co\_name=? where co\_code=? 10-07-25 16:03:22.445 restartedMain INFO c.c.o.OrmLearnApplication testUpdateCountry 58 End 10-07-25 16:03:22.445 restartedMain INFO c.c.o.OrmLearnApplication testDeleteCountry 62 Start: Delete Country 10-07-25 16:03:22.453 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 16:03:22.469 restartedMain DEBUG org.hibernate.SQL logStatement 135 delete from country where co\_code=? 10-07-25 16:03:22.477 restartedMain INFO c.c.o.OrmLearnApplication testDeleteCountry 64 End 10-07-25 16:03:22.478 restartedMain INFO c.c.o.OrmLearnApplication testFindCountriesByName 68 Start: Find Countries by Partial Name 10-07-25 16:03:22.524 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_name like ? escape '\' 10-07-25 16:03:22.538 restartedMain DEBUG c.c.o.OrmLearnApplication testFindCountriesByName 70 **Matching Countries: [Country [code=AX, name=Åland Islands], Country [code=BV, name=Bouvet Island], Country [code=CC, name=Cocos (Keeling) Islands], Country [code=CH, name=Switzerland], Country [code=CK, name=Cook Islands], Country [code=CX, name=Christmas Island], Country [code=FI, name=Finland], Country [code=FK, name=Falkland Islands (Malvinas)], Country [code=FO, name=Faroe Islands], Country [code=GL, name=Greenland], Country [code=GS, name=South Georgia and the South Sandwich Islands], Country [code=HM, name=Heard Island and McDonald Islands], Country [code=IE, name=Ireland], Country [code=IS, name=Iceland], Country [code=KY, name=Cayman Islands], Country [code=MH, name=Marshall Islands], Country [code=MP, name=Northern Mariana Islands], Country [code=NF, name=Norfolk Island], Country [code=NL, name=Netherlands], Country [code=NZ, name=New Zealand], Country [code=PL, name=Poland], Country [code=SB, name=Solomon Islands], Country [code=SZ, name=Swaziland], Country [code=TC, name=Turks and Caicos Islands], Country [code=TH, name=Thailand], Country [code=UM, name=United States Minor Outlying Islands], Country [code=VG, name=Virgin Islands, British], Country [code=VI, name=Virgin Islands, U.S.]]** 10-07-25 16:03:22.538 restartedMain INFO c.c.o.OrmLearnApplication testFindCountriesByName 71 End 10-07-25 16:03:22.544 licationShutdownHook INFO rEntityManagerFactoryBean destroy 660 Closing JPA EntityManagerFactory for persistence unit 'default' 10-07-25 16:03:22.547 licationShutdownHook INFO c.z.h.HikariDataSource close 349 HikariPool-1 - Shutdown initiated... 10-07-25 16:03:22.559 licationShutdownHook INFO c.z.h.HikariDataSource close 351 HikariPool-1 - Shutdown completed.

Process finished with exit code 0

**Hands on 6**

**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.

**NOTE:** SME to explain the importance of @Transactional annotation. Spring takes care of creating the Hibernate session and manages the transactionality when executing the service method.

**Solution**

**CountryNotFoundException.java**

package com.cognizant.orm\_learn.service.exception;  
  
public class CountryNotFoundException extends Exception {  
 public CountryNotFoundException(String message) {  
 super(message);  
 }  
}

**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import jakarta.transaction.Transactional;  
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;  
  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
 @Transactional  
 public Country getCountry(String code) {  
 return countryRepository.findById(code).orElse(null);  
 }  
  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
  
 @Transactional  
 public void updateCountry(String code, Country updatedCountry) {  
 Country country = countryRepository.findById(code).orElse(null);  
 if (country != null) {  
 country.setName(updatedCountry.getName());  
 countryRepository.save(country);  
 }  
 }  
  
 @Transactional  
 public void deleteCountry(String code) {  
 countryRepository.deleteById(code);  
 }  
  
 @Transactional  
 public List<Country> findCountriesByName(String name) {  
 return countryRepository.findByNameContaining(name);  
 }  
 @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();  
 }  
}

**OrmLEarnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.service.CountryService;  
import com.cognizant.orm\_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;  
  
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);  
 countryService = context.getBean(CountryService.class);  
  
 getAllCountriesTest();  
 }  
  
 private static void getAllCountriesTest() {  
 LOGGER.info("Start");  
  
 try {  
 Country country = countryService.findCountryByCode("IN");  
 LOGGER.debug("Country: {}", country);  
 } catch (CountryNotFoundException e) {  
 LOGGER.error("Error: {}", e.getMessage());  
 }  
  
 LOGGER.info("End");  
 }  
}

**Output**

10-07-25 16:44:00.288 restartedMain INFO c.c.o.OrmLearnApplication getAllCountriesTest 28 Start 10-07-25 16:44:00.341 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 16:44:00.394 restartedMain DEBUG c.c.o.OrmLearnApplication getAllCountriesTest 32 Country: **Country [code=IN, name=India]** 10-07-25 16:44:00.394 restartedMain INFO c.c.o.OrmLearnApplication getAllCountriesTest 37 End

**Hands on 7**

**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:
  + Create new instance of country with a new code and name
  + Call countryService.addCountry() passing the country created in the previous step.
  + Invoke countryService.findCountryByCode() passing the same code used when adding a new country
  + Check in the database if the country is added

**Solution**

**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import jakarta.transaction.Transactional;  
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;  
  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
 @Transactional  
 public Country getCountry(String code) {  
 return countryRepository.findById(code).orElse(null);  
 }  
  
 @Transactional public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
  
 @Transactional  
 public void updateCountry(String code, Country updatedCountry) {  
 Country country = countryRepository.findById(code).orElse(null);  
 if (country != null) {  
 country.setName(updatedCountry.getName());  
 countryRepository.save(country);  
 }  
 }  
  
 @Transactional  
 public void deleteCountry(String code) {  
 countryRepository.deleteById(code);  
 }  
  
 @Transactional  
 public List<Country> findCountriesByName(String name) {  
 return countryRepository.findByNameContaining(name);  
 }  
 **@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();**  
 **}**  
  
}

**OrmLearnApplication.java**

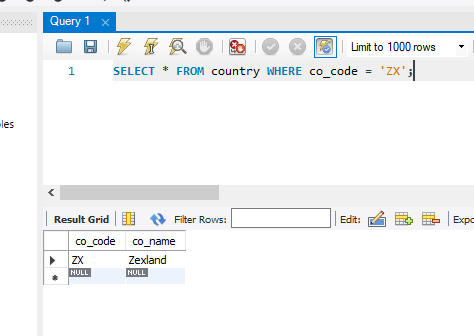
package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.service.CountryService;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
  
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);  
 countryService = context.getBean(CountryService.class);

**testAddCountry();**  
 }  
  
  **private static void testAddCountry() {**  
 **LOGGER.info("Start: testAddCountry");**  
  
 **Country newCountry = new Country();**  
 **newCountry.setCode("ZX");**  
 **newCountry.setName("Zexland");**  
  
 **countryService.addCountry(newCountry);**  
  
 **try {**  
 **Country fetched = countryService.findCountryByCode("ZX");**  
 **LOGGER.debug("Added Country: {}", fetched);**  
 **} catch (CountryNotFoundException e) {**  
 **LOGGER.error("Country not found after insert: {}", e.getMessage());**  
 **}**  
  
 **LOGGER.info("End: testAddCountry");**  
 **}**  
}

**Output**

10-07-25 16:52:40.253 restartedMain INFO c.c.o.OrmLearnApplication logStarted 59 Started OrmLearnApplication in 5.805 seconds (process running for 7.186) 10-07-25 16:52:40.263 restartedMain INFO c.c.o.OrmLearnApplication testAddCountry 28 Start: testAddCountry 10-07-25 16:52:40.312 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 16:52:40.375 restartedMain DEBUG org.hibernate.SQL logStatement 135 insert into country (co\_name,co\_code) values (?,?) 10-07-25 16:52:40.401 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 16:52:40.407 restartedMain DEBUG c.c.o.OrmLearnApplication testAddCountry 38 Added **Country: Country [code=ZX, name=Zexland]** 10-07-25 16:52:40.407 restartedMain INFO c.c.o.OrmLearnApplication testAddCountry 43 End: testAddCountry 10-07-25 16:52:40.422 licationShutdownHook INFO rEntityManagerFactoryBean destroy 660 Closing JPA EntityManagerFactory for persistence unit 'default' 10-07-25 16:52:40.429 licationShutdownHook INFO c.z.h.HikariDataSource close 349 HikariPool-1 - Shutdown initiated... 10-07-25 16:52:40.465 licationShutdownHook INFO c.z.h.HikariDataSource close 351 HikariPool-1 - Shutdown completed.

Process finished with exit code 0



**Hands on 8**

**Update a country based on code**

* Create a new method updateCountry() in CountryService with parameters code and name. Annotate this method with @Transactional. Implement following steps in this method.
  + Get the reference of the country using findById() method in repository
  + In the country reference obtained, update the name of country using setter method
  + Call countryRepository.save() method to update the name
* Include new test method in OrmLearnApplication, which invokes updateCountry() method in CountryService passing a country's code and different name for the country.
* Check in database table if name is modified.

**CountryService.java**

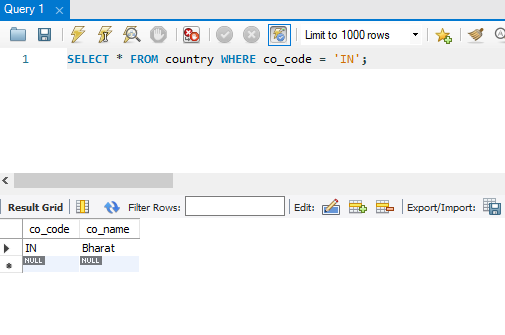
package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import jakarta.transaction.Transactional;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
 @Transactional  
 public Country getCountry(String code) {  
 return countryRepository.findById(code).orElse(null);  
 }  
  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
  
 @Transactional  
 public void updateCountry(String code, Country updatedCountry) {  
 Country country = countryRepository.findById(code).orElse(null);  
 if (country != null) {  
 country.setName(updatedCountry.getName());  
 countryRepository.save(country);  
 }  
 }  
  
 @Transactional  
 public void deleteCountry(String code) {  
 countryRepository.deleteById(code);  
 }  
  
 @Transactional  
 public List<Country> findCountriesByName(String name) {  
 return countryRepository.findByNameContaining(name);  
 }  
 @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 updateCountry(String code, String name) {**  
 **Optional<Country> result = countryRepository.findById(code);**  
 **if (result.isPresent()) {**  
 **Country country = result.get();**  
 **country.setName(name);**  
 **countryRepository.save(country);**  
 **}**  
 **}**  
  
}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.service.CountryService;  
import com.cognizant.orm\_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;  
  
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);  
 countryService = context.getBean(CountryService.class);  
  
  **testUpdateCountry();**  
 }  
  
  **private static void testUpdateCountry() {**  
 **LOGGER.info("Start: testUpdateCountry");**  
  
 **String countryCode = "IN";**  
 **String newName = "Bharat";**  
  
 **countryService.updateCountry(countryCode, newName);**  
  
 **try {**  
 **Country updated = countryService.findCountryByCode(countryCode);**  
 **LOGGER.debug("Updated Country: {}", updated);**  
 **} catch (CountryNotFoundException e) {**  
 **LOGGER.error("Country not found: {}", e.getMessage());**  
 **}**  
  
 **LOGGER.info("End: testUpdateCountry");**  
 **}**  
}

**Output**

10-07-25 17:00:03.489 restartedMain INFO c.c.o.OrmLearnApplication testUpdateCountry 28 Start: testUpdateCountry 10-07-25 17:00:03.537 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 17:00:03.597 restartedMain DEBUG org.hibernate.SQL logStatement 135 update country set co\_name=? where co\_code=? 10-07-25 17:00:03.615 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 17:00:03.630 restartedMain DEBUG c.c.o.OrmLearnApplication testUpdateCountry 37 Updated Country: **Country [code=IN, name=Bharat]** 10-07-25 17:00:03.631 restartedMain INFO c.c.o.OrmLearnApplication testUpdateCountry 42 End: testUpdateCountry



**Hands on 9**

**Delete a country based on code**

* Create new method deleteCountry() in CountryService. Annotate this method with @Transactional.
* In deleteCountry() method call deleteById() method of repository.
* Include new test method in OrmLearnApplication with following steps
  + Call the delete method based on the country code during the add country hands on
* Check in database if the country is deleted

**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import jakarta.transaction.Transactional;  
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;  
  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
 @Transactional  
 public Country getCountry(String code) {  
 return countryRepository.findById(code).orElse(null);  
 }  
  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
  
 @Transactional  
 public void updateCountry(String code, Country updatedCountry) {  
 Country country = countryRepository.findById(code).orElse(null);  
 if (country != null) {  
 country.setName(updatedCountry.getName());  
 countryRepository.save(country);  
 }  
 }  
  
 **@Transactional**  
 **public void deleteCountry(String code) {**  
 **countryRepository.deleteById(code);**  
}  
  
 @Transactional  
 public List<Country> findCountriesByName(String name) {  
 return countryRepository.findByNameContaining(name);  
 }  
 @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 updateCountry(String code, String name) {  
 Optional<Country> result = countryRepository.findById(code);  
 if (result.isPresent()) {  
 Country country = result.get();  
 country.setName(name);  
 countryRepository.save(country);  
 }  
 }  
  
}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.service.CountryService;  
import com.cognizant.orm\_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;  
  
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);  
 countryService = context.getBean(CountryService.class);  
  
 **testDeleteCountry();**  
 }  
  
 **private static void testDeleteCountry() {**  
 **LOGGER.info("Start: testDeleteCountry");**  
  
 **String codeToDelete = "ZX"; // Assuming this was added in testAddCountry()**  
  
 **countryService.deleteCountry(codeToDelete);**  
  
 **try {**  
 **Country country = countryService.findCountryByCode(codeToDelete);**  
 **LOGGER.debug("Country still exists: {}", country);**  
 **} catch (CountryNotFoundException e) {**  
 **LOGGER.info("Country deleted successfully: {}", codeToDelete);**  
 **}**  
  
 **LOGGER.info("End: testDeleteCountry");**  
 **}**  
}

**Output**

10-07-25 17:10:21.875 restartedMain INFO c.c.o.OrmLearnApplication testDeleteCountry 28 Start: testDeleteCountry 10-07-25 17:10:21.921 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 17:10:21.993 restartedMain DEBUG org.hibernate.SQL logStatement 135 delete from country where co\_code=? 10-07-25 17:10:22.008 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1\_0.co\_code,c1\_0.co\_name from country c1\_0 where c1\_0.co\_code=? 10-07-25 17:10:22.011 restartedMain INFO c.c.o.OrmLearnApplication testDeleteCountry 38 **Country deleted successfully: ZX** 10-07-25 17:10:22.011 restartedMain INFO c.c.o.OrmLearnApplication testDeleteCountry 41 End: testDeleteCountry

