1. Spring Data JPA Handson

Hands on 1 : Spring Data JPA - Quick Example (Mandatory Handson)

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 %-25.25logge
r{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:

1. src/main/java - Folder with application code
2. src/main/resources - Folder for application configuration
3. src/test/java - Folder with code for testing the application
4. OrmLearnApplication.java - Walkthrough the main() method.
5. Purpose of @SpringBootApplication annotation

6. pom.xml
    1. Walkthrough all the configuration defined in XML file
    2. 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.

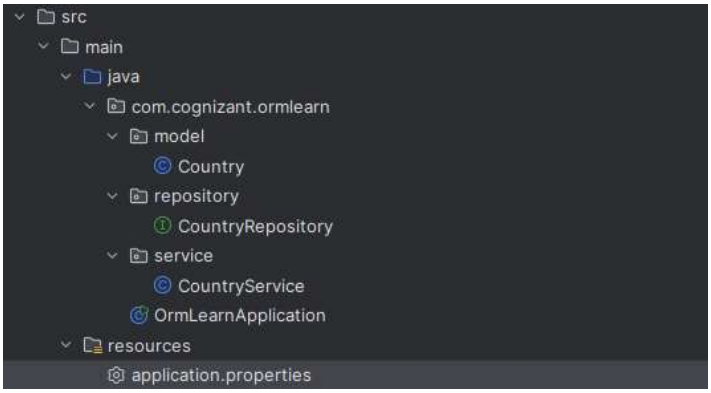
```
    ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
    countryService = context.getBean(CountryService.class);


    testGetAllCountries();
```

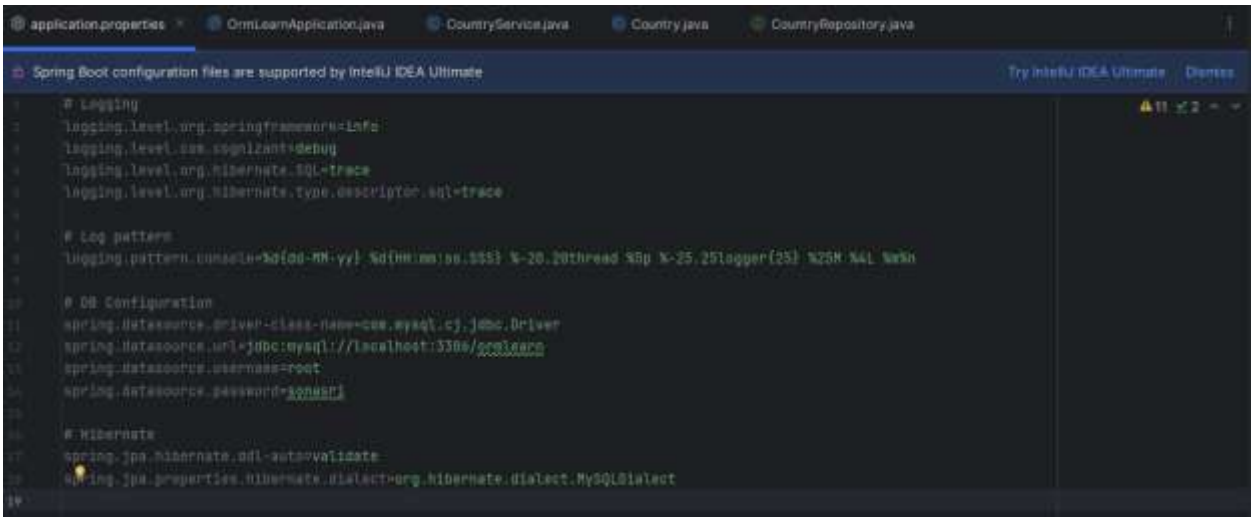
- Execute main method to check if data from ormlearn database is retrieved.

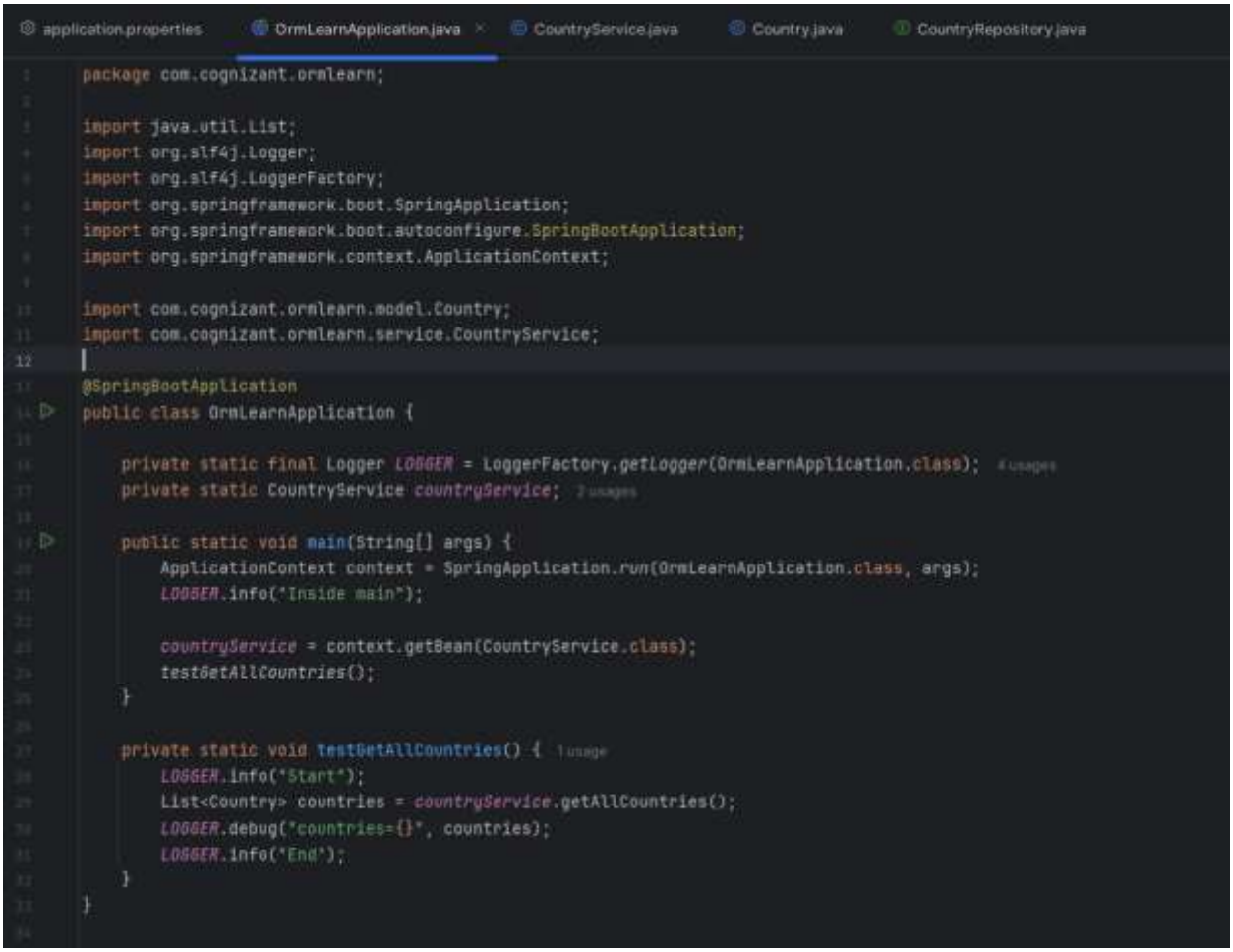
Implementation:

File Structure:



Code:


application.properties
OrmLearnApplication.java
CountryService.java
Country.java
CountryRepository.java
Spring Boot configuration files are supported by IntelliJ IDEA Ultimate
Try IntelliJ IDEA Ultimate
Dismiss
# Logging
logging.level.org.springframework=info
logging.level.com.cognizant=debug
logging.level.org.hibernate.SQL=trace
logging.level.org.hibernate.type.descriptor.sql=trace
# Log pattern
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n
# DB 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=sqluser1
# Hibernate
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect


application.properties
OrmLearnApplication.java
CountryService.java
Country.java
CountryRepository.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");
}
}

application.properties | OrmLearnApplication.java | CountryService.java | Country.java | CountryRepository.java

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

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;

import jakarta.transaction.Transactional;

@Service 3 usages
public class CountryService {

    @Autowired 1 usage
    private CountryRepository countryRepository;

    @Transactional 1 usage
    public List<Country> getAllCountries() { return countryRepository.findAll(); }
}
```

application.properties | OrmLearnApplication.java | CountryService.java | Country.java | CountryRepository.java

```java
package com.cognizant.ormlearn.model;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity 0 usages
@Table(name = "country")
public class Country {

    @Id 3 usages
    @Column(name = "co_code")
    private String code;

    @Column(name = "co_name") 3 usages
    private String name;

    // Getters and Setters
    public String getCode() { return code; } no usages
    public void setCode(String code) { this.code = code; } no usages

    public String getName() { return name; } no usages
    public void setName(String name) { this.name = name; } no usages

    @Override
    public String toString() { return "Country [code=" + code + ", name=" + name + "]"; }
}
```

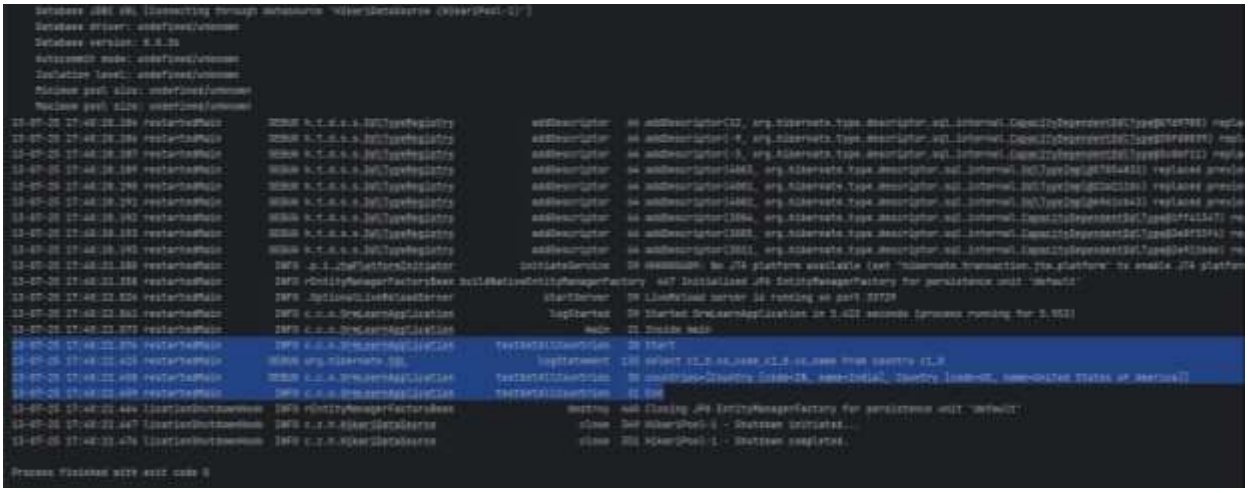
application.properties | OrmLearnApplication.java | CountryService.java | Country.java | CountryRepository.java

```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  2 usages
public interface CountryRepository extends JpaRepository<Country, String> {
}
```

Output:



Hands on 2 - Hibernate XML Config implementation walk through (Neither Mandatory nor Additional)

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()

Answer:

1. Object-Relational Mapping (ORM) in Hibernate XML
Hibernate uses **.hbm.xml files** to map Java classes (objects) to database tables.
Each class is mapped to a table, and each field is mapped to a column.

Example: Employee.hbm.xml

```
<hibernate-mapping>
  <class name="Employee" table="EMPLOYEE">
    <id name="id" type="int" column="ID">
      <generator class="native"/>
    </id>
    <property name="firstName" column="FIRST_NAME" type="string"/>
    <property name="lastName" column="LAST_NAME" type="string"/>
    <property name="salary" column="SALARY" type="float"/>
  </class>
</hibernate-mapping>
```

Explanation:

- <class>: Maps the class to a DB table (Employee → EMPLOYEE)
- <id>: Specifies the primary key column (id → ID)
- <property>: Maps fields to columns (firstName → FIRST_NAME etc.)

2. Hibernate End-to-End Operations

SessionFactory

- Created once at startup.
- Heavyweight object responsible for managing DB connections.
- Built from hibernate.cfg.xml using Configuration class.

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

Session

- A lightweight object used for performing CRUD operations.
- Acts like a short-lived connection to the database.

```
Session session = factory.openSession();
```

Transaction

- Required to group database operations into an atomic unit.

```
Transaction tx = session.beginTransaction();
```

beginTransaction()

- Starts a new transaction.
- All operations after this will be part of the same transaction.

```
tx.begin();  // implicitly done with beginTransaction()
```

commit()

- Finalizes the transaction and saves changes to the database.

```
tx.commit();
```

rollback()

- Used to undo the changes if any exception occurs.

```
tx.rollback();
```

Hibernate CRUD Methods

session.save(object)

- Saves a new object to the DB and returns the generated primary key.

```
Integer empID = (Integer) session.save(emp);
```

session.get(Class, id)

- Retrieves an object based on the primary key.
- Returns null if not found.

```
Employee emp = session.get(Employee.class, 101);
```

session.createQuery().list()

- Executes a query and returns a list of results.

```
List<Employee> employees = session.createQuery("FROM Employee").list();
```

session.delete(object)

- Deletes the given object from the database.

```
session.delete(emp);
```

Full Flow Example:

```
SessionFactory factory = new Configuration().configure().buildSessionFactory();
Session session = factory.openSession();
Transaction tx = null;

try {
   tx = session.beginTransaction();

   Employee emp = new Employee();
   emp.setFirstName("John");
   emp.setLastName("Doe");
   emp.setSalary(50000);
   session.save(emp);  // INSERT

   List<Employee> empList = session.createQuery("FROM Employee").list();  // SELECT

   Employee empToDelete = session.get(Employee.class, 101); // SELECT by ID
   session.delete(empToDelete); // DELETE

   tx.commit();
} catch (Exception e) {
   if (tx != null) tx.rollback();
   e.printStackTrace();
} finally {
   session.close();
}
```

Hands on 3- Hibernate Annotation Config implementation walk through (Neither Mandatory nor Additional)

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

Answer:

1. Object to Relational Mapping using Annotations in Employee class

In Hibernate annotation-based configuration, the **Java class itself** is used to define mappings to the database using **JPA annotations** (from jakarta.persistence or javax.persistence).

Example: Employee.java

```
import jakarta.persistence.*;
@Entity
@Table(name = "EMPLOYEE")
public class Employee {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "ID")
   private int id;

   @Column(name = "FIRST_NAME")
   private String firstName;

   @Column(name = "LAST_NAME")
   private String lastName;

   @Column(name = "SALARY")
   private int salary;

   // Getters and setters
}
```

2. Explanation of Annotations

Annotation	Description
@Entity	Marks this class as a persistent entity mapped to a table. Hibernate will manage this class.
@Table(name = "EMPLOYEE")	Specifies the exact table name in the database that this class maps to.
@Id	Indicates the primary key field.
@GeneratedValue	Auto-generates the primary key value. Strategy can be AUTO, IDENTITY, SEQUENCE, or TABLE.
@Column(name = "...")	Maps the Java field to the specific database column name.

3. Hibernate Configuration (hibernate.cfg.xml)

This file configures the Hibernate setup and DB connection details.

Sample hibernate.cfg.xml

```
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>

    <!-- JDBC Database Configuration -->
    <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/your_database</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">root</property>

    <!-- SQL Dialect -->
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

    <!-- Hibernate Configuration -->
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.hbm2ddl.auto">update</property>

    <!-- Annotated Class -->
    <mapping class="com.example.Employee"/>

  </session-factory>
</hibernate-configuration>
```

4. Key Config Elements Explained

Property	Description
hibernate.connection.driver_class	Specifies JDBC driver for the database (e.g., MySQL).
hibernate.connection.url	Full JDBC connection URL to your database.
hibernate.connection.username	Username used to connect to the database.
hibernate.connection.password	Password used to connect to the database.
hibernate.dialect	Tells Hibernate how to convert HQL into native SQL (e.g., MySQL dialect).
hibernate.hbm2ddl.auto	Schema update option (update, create, validate, etc.).
hibernate.show_sql	Displays SQL commands in the console for debugging.

Hands on 4- Difference between JPA, Hibernate and Spring Data JPA (Mandatory Handson)

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

Answer:

1. JPA (Java Persistence API)

- JPA is a specification (JSR 338) created by Oracle.
- It provides standard guidelines to map Java objects to relational database tables (ORM).
- JPA does not provide any implementation—it only defines interfaces (e.g., EntityManager, @Entity, etc.).
- Hibernate is one of the most popular implementations of JPA.

2. Hibernate

- Hibernate is a **concrete ORM framework** that **implements JPA**.
- Provides additional features over JPA like caching, lazy loading, and HQL (Hibernate Query Language).
- Developers often use Hibernate APIs directly or through JPA interfaces.
- Requires more **boilerplate code** for sessions, transactions, and CRUD operations.

3. Spring Data JPA

- Spring Data JPA is **not a JPA provider**—it's a Spring module that:
  - **Wraps JPA/Hibernate**.
  - **Reduces boilerplate code** by providing repository interfaces.
  - Handles **transactions, sessions, queries**, etc., internally.
- Developers **only define interfaces**, and Spring provides the implementation at runtime.

Comparison with Code:

Aspect	Hibernate	Spring Data JPA
Setup	Requires Session, Transaction, manual rollback	Handled automatically
Code	Manual save, commit, rollback, close	Just call save()
Error handling	Explicit try-catch-finally	Internally handled
Repository	Not available	Built-in via JpaRepository

Hibernate Example:

```
public Integer addEmployee(Employee employee){
   Session session = factory.openSession();
   Transaction tx = null;
   Integer id = null;
   try {
      tx = session.beginTransaction();
      id = (Integer) session.save(employee);
      tx.commit();
   } catch (HibernateException e) {
      if (tx != null) tx.rollback();
      e.printStackTrace();
   } finally {
      session.close();
   }
   return id;
}
```

Spring Data JPA Example:

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

@Service
public class EmployeeService {
   @Autowired
   private EmployeeRepository employeeRepository;

   @Transactional
   public void addEmployee(Employee employee) {
      employeeRepository.save(employee);
   }
}
```

Hands on 5- Implement services for managing Country (Additional Handson)

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");
```

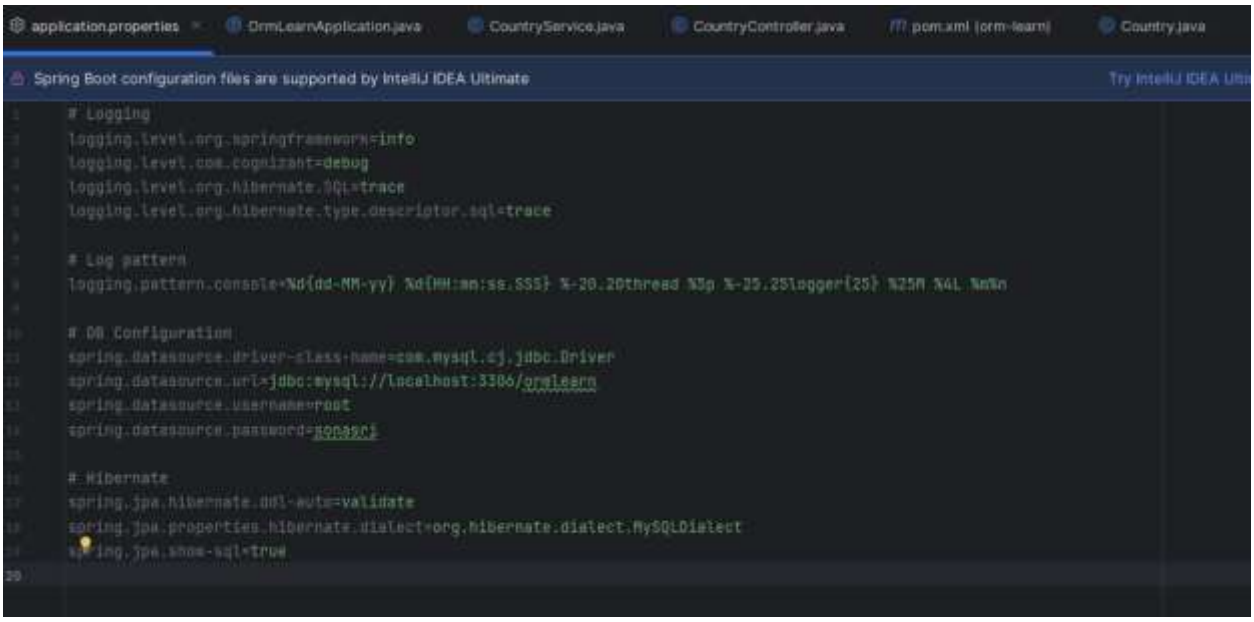
```
insert into country (co_code, co_name) values ("TW", "Taiwan, Province of China");
insert into country (co_code, co_name) values ("TJ", "Tajikistan");
insert into country (co_code, co_name) values ("TZ", "Tanzania, United Republic of");
insert into country (co_code, co_name) values ("TH", "Thailand");
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");
```

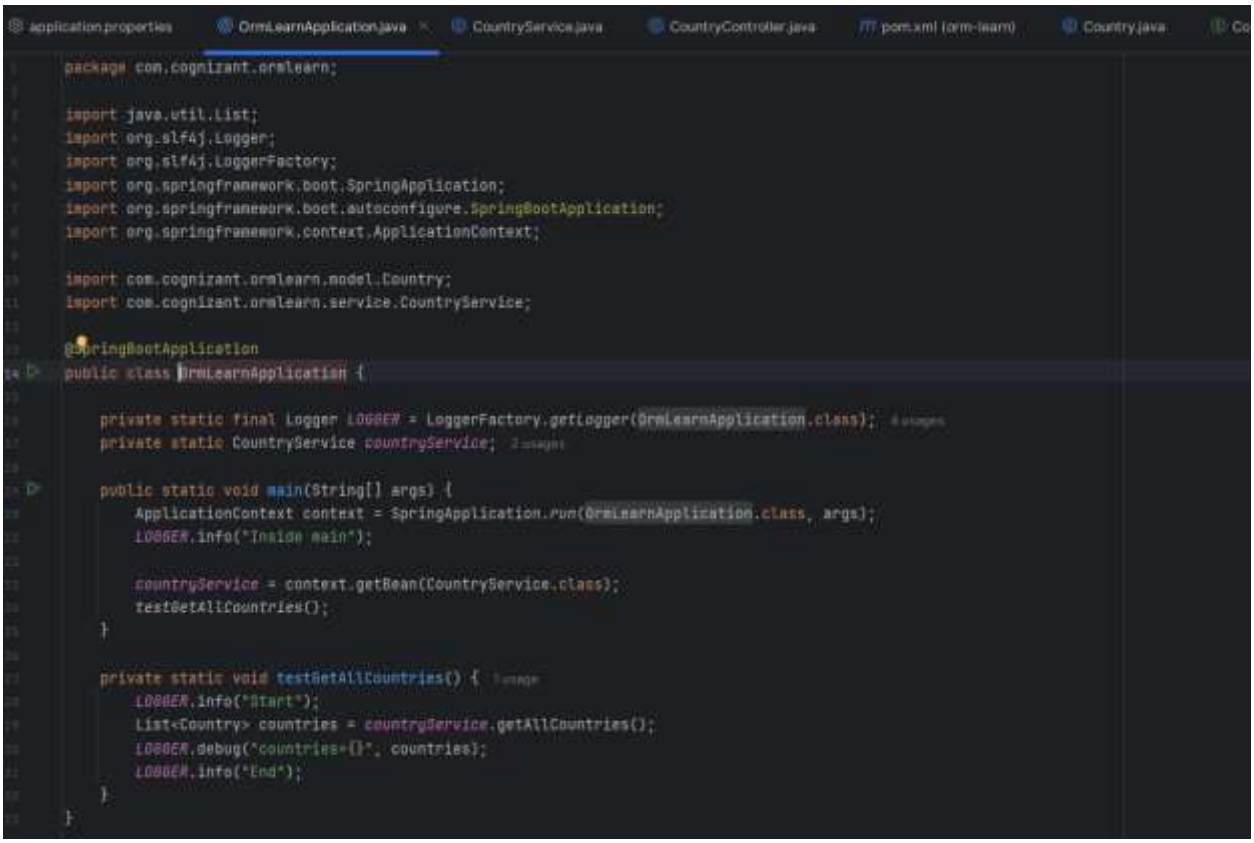
Refer subsequent hands on exercises to implement the features related to country.

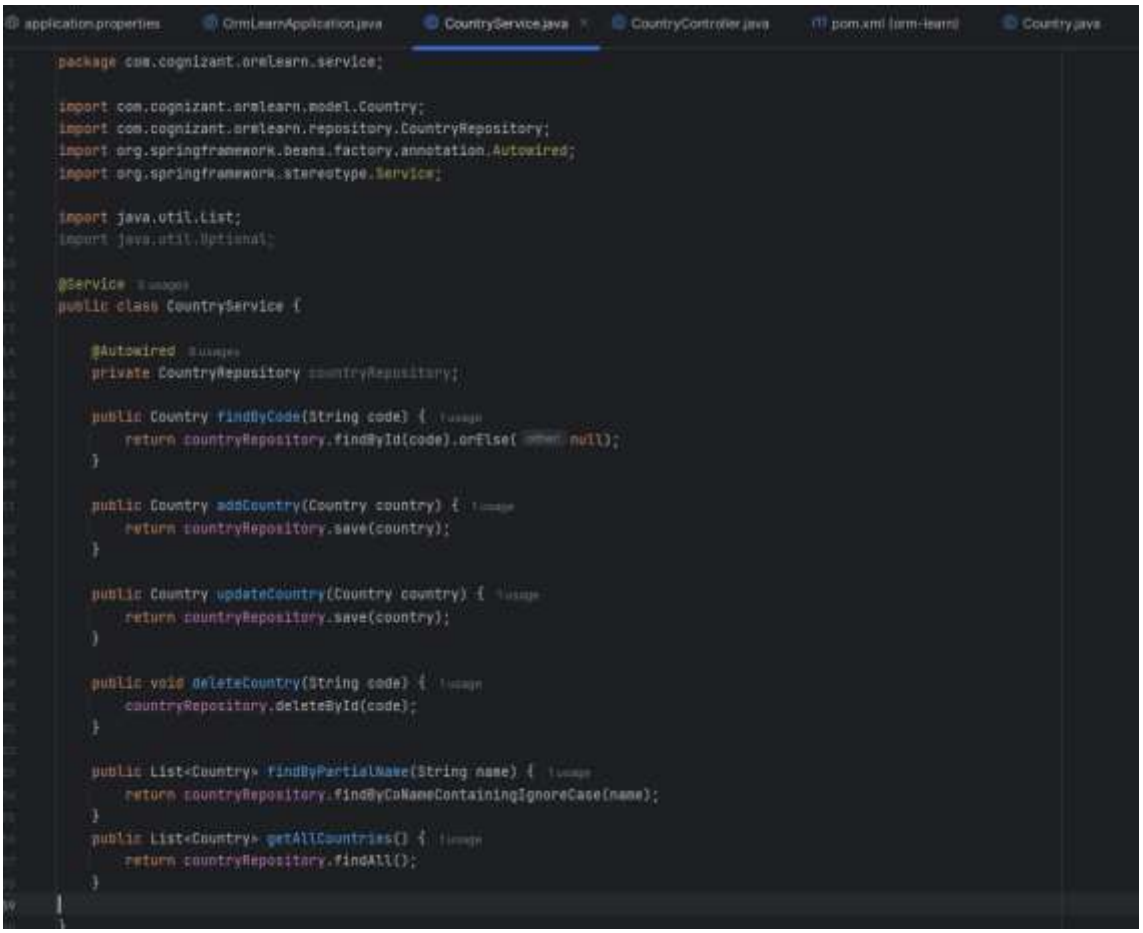
Implementation:

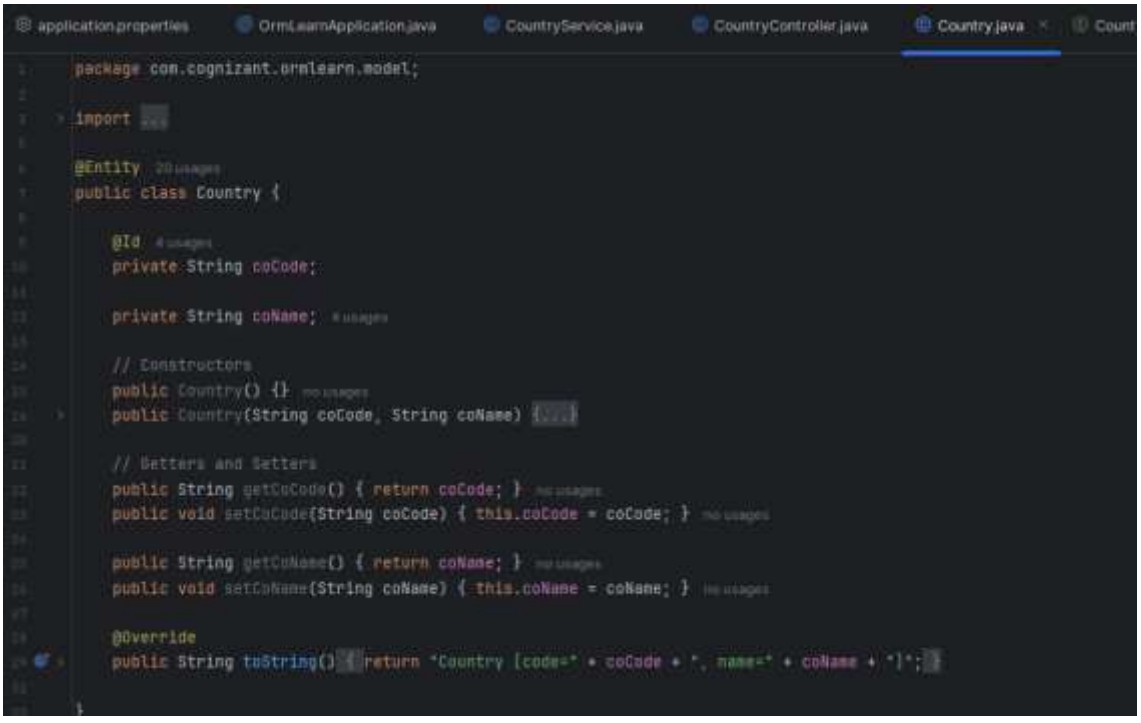
File Structure:

Code:


application.properties
OrmLearnApplication.java
CountryService.java
CountryController.java
pom.xml (orm-learn)
Country.java
Spring Boot configuration files are supported by IntelliJ IDEA Ultimate
Try IntelliJ IDEA Ult
# Logging
logging.level.org.springframework=info
logging.level.com.cognizant=debug
logging.level.org.hibernate.SQL=trace
logging.level.org.hibernate.type.descriptor.sql=trace
# Log pattern
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n
# DB 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=sonasri
# Hibernate
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true


application.properties
OrmLearnApplication.java
CountryService.java
CountryController.java
pom.xml (orm-learn)
Country.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");
}
}


application.properties
OrmLearnApplication.java
CountryService.java
CountryController.java
pom.xml (orm-learn)
Country.java
package com.cognizant.ormlearn.service;
import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Optional;
@Service
public class CountryService {
@Autowired
private CountryRepository countryRepository;
public Country findByCode(String code) {
return countryRepository.findById(code).orElse(other: null);
}
public Country addCountry(Country country) {
return countryRepository.save(country);
}
public Country updateCountry(Country country) {
return countryRepository.save(country);
}
public void deleteCountry(String code) {
countryRepository.deleteById(code);
}
public List<Country> findByPartialName(String name) {
return countryRepository.findByCoNameContainingIgnoreCase(name);
}
public List<Country> getAllCountries() {
return countryRepository.findAll();
}
}


application.properties
OrmLearnApplication.java
CountryService.java
CountryController.java
Country.java
package com.cognizant.ormlearn.model;
import ...
@Entity
public class Country {
@Id
private String coCode;
private String coName;
// Constructors
public Country() {}
public Country(String coCode, String coName) {...}
// Getters and Setters
public String getCoCode() { return coCode; }
public void setCoCode(String coCode) { this.coCode = coCode; }
public String getCoName() { return coName; }
public void setCoName(String coName) { this.coName = coName; }
@Override
public String toString() { return "Country [code=" + coCode + ", name=" + coName + "]"; }
}

application.properties | OrmLearnApplication.java | CountryService.java | CountryController.java | Country.java

```java
package com.cognizant.ormlearn.controller;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.service.CountryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import jakarta.persistence.*;
@RestController  no usages
@RequestMapping("/countries")
public class CountryController {

    @Autowired  5 usages
    private CountryService countryService;

    @GetMapping("/{code}")  no usages
    public Country getCountryByCode(@PathVariable String code) {
        return countryService.findByCode(code);
    }

    @PostMapping  no usages
    public Country addCountry(@RequestBody Country country) {
        return countryService.addCountry(country);
    }

    @PutMapping  no usages
    public Country updateCountry(@RequestBody Country country) {
        return countryService.updateCountry(country);
    }

    @DeleteMapping("/{code}")  no usages
    public void deleteCountry(@PathVariable String code) {
        countryService.deleteCountry(code);
    }

    @GetMapping("/search")  no usages
    public List<Country> searchCountries(@RequestParam String name) {
        return countryService.findByPartialName(name);
    }
}

```

application.properties | OrmLearnApplication.java | CountryService.java | CountryController.java | Country.java

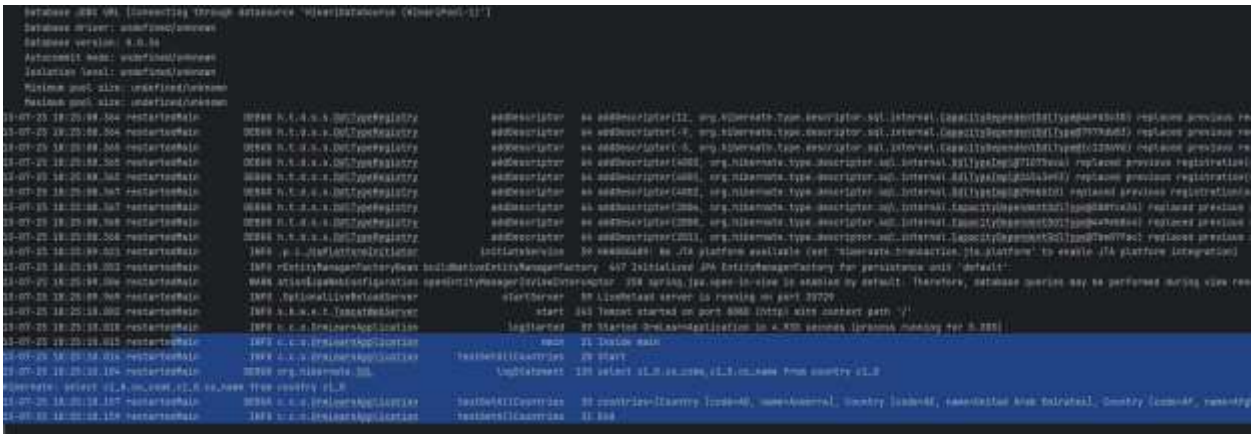
```java
package com.cognizant.ormlearn.repository;

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

public interface CountryRepository extends JpaRepository<Country, String> {  2 usages
    List<Country> findByCoNameContainingIgnoreCase(String name);  no usages
}

```

Output:



Hands on 6 - Find a country based on country code (Additional Handson)

- 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 CountryNotFoundExceptio
n
```

- 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.

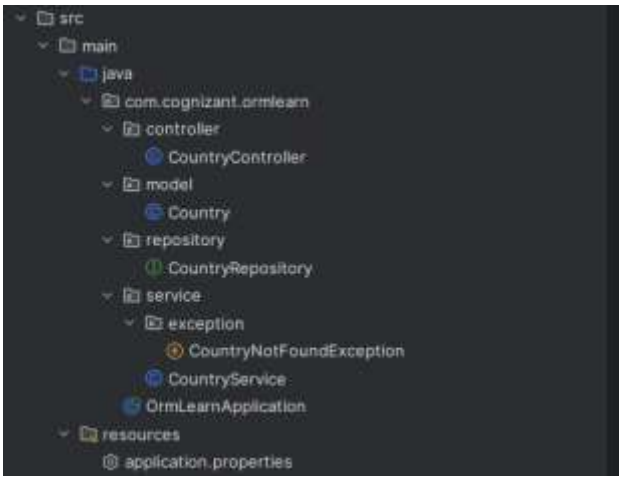
```
private static void getAllCountriesTest() {
    LOGGER.info("Start");
    Country country = countryService.findCountryByCode("IN");
    LOGGER.debug("Country:{}", country);
    LOGGER.info("End");
}
```

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

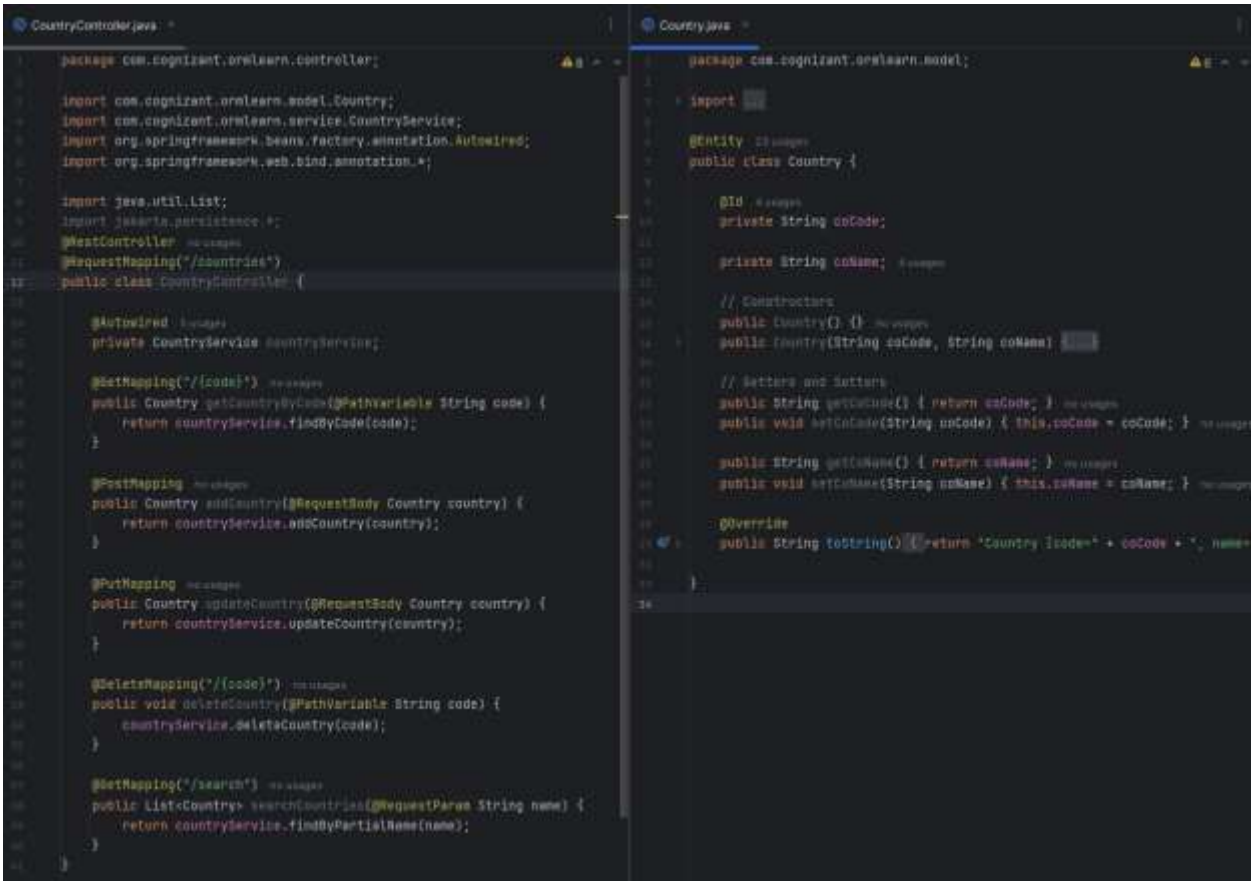
NOTE: SME to explain the importance of @Transactional annotation. Spring takes care of creating the Hibernate session and manages the transactionality when executing the service method.

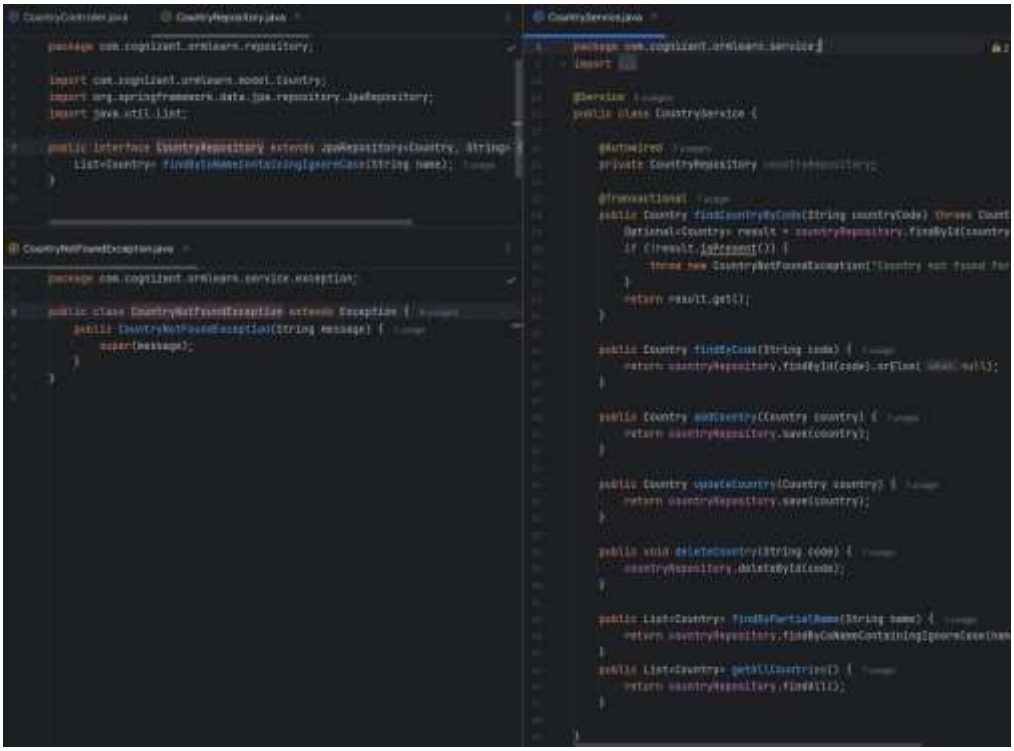
Implementation:

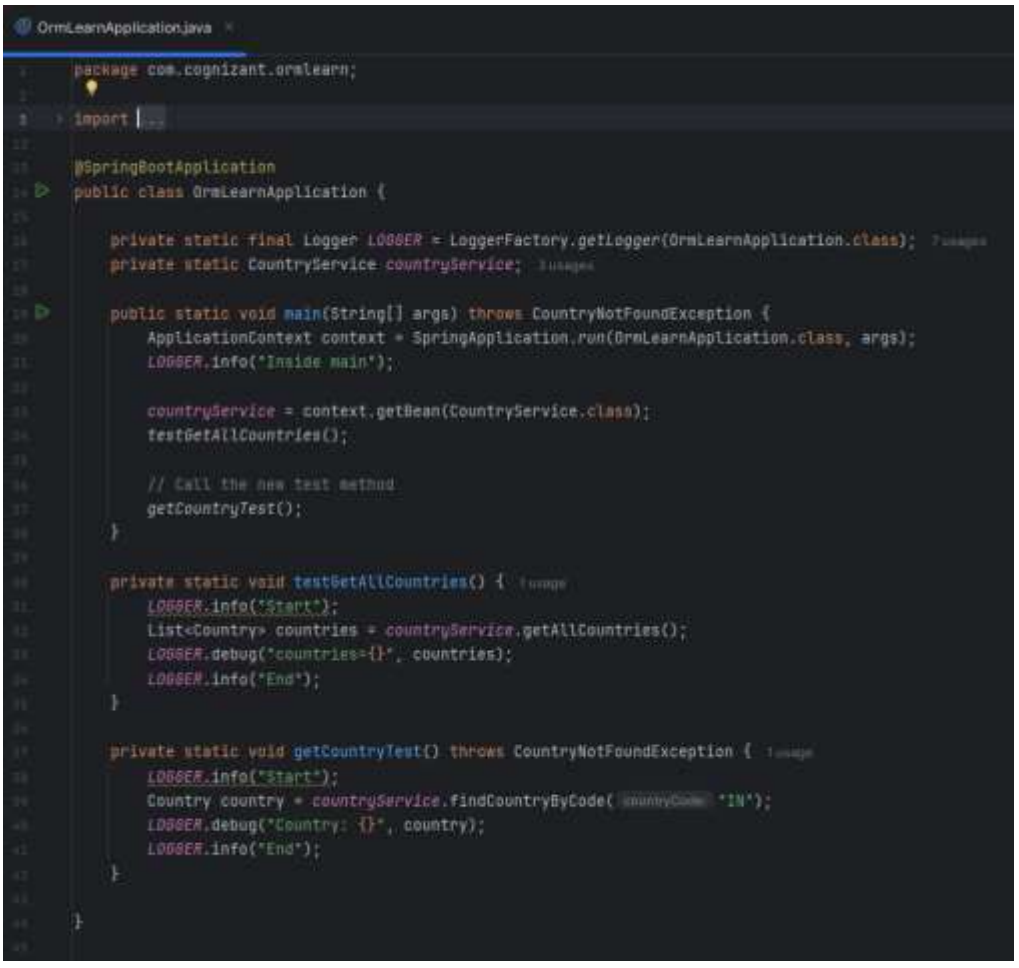
File Structure:



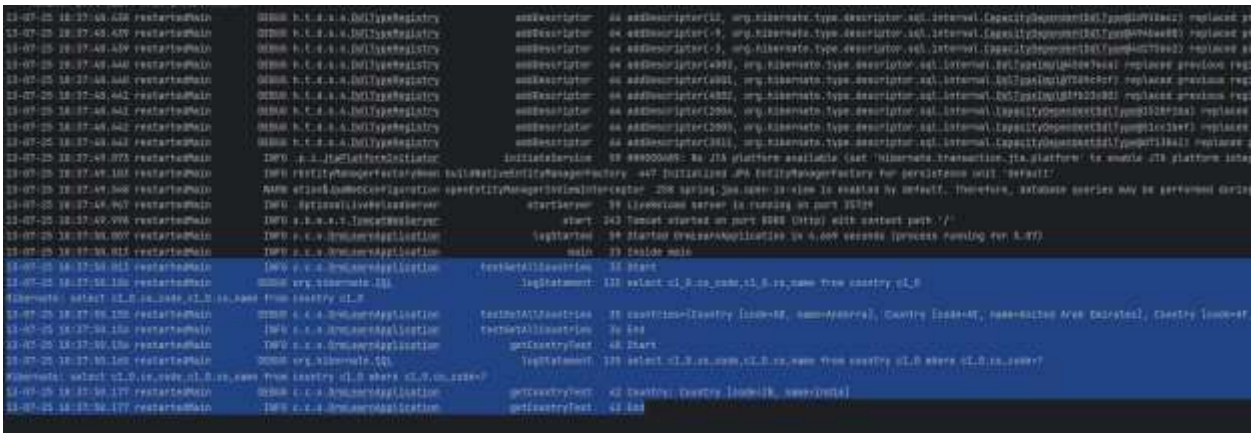
Code:







Output:



Hands on 7- Add a new country (Additional Handson)

- Create new method in CountryService.

```
@Transactional
public void addCountry(Country country)
```

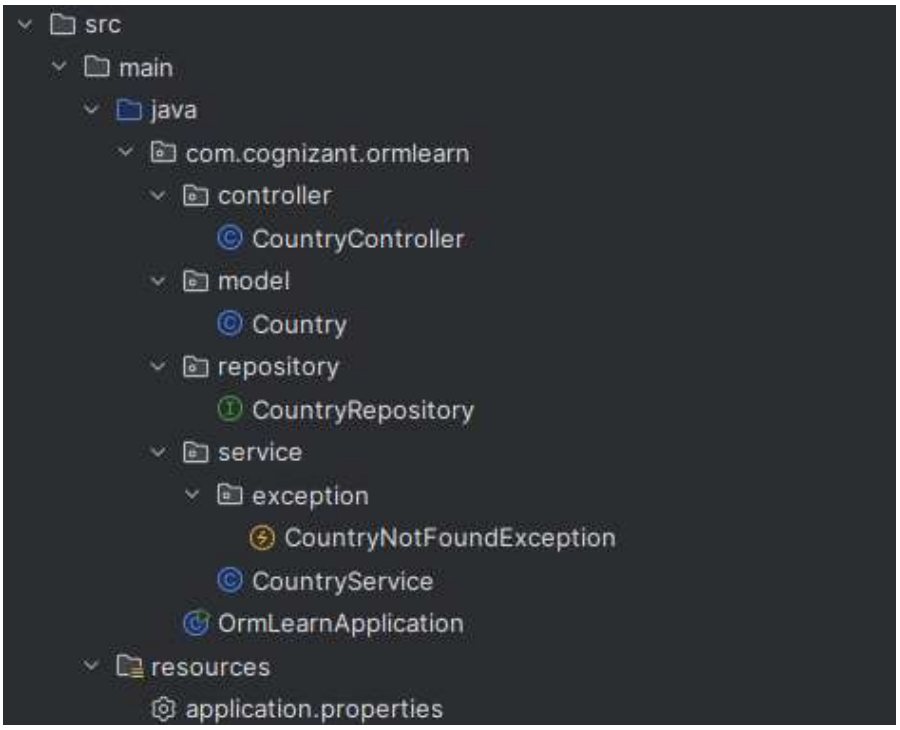
- Invoke save() method of repository to get the country added.

```
countryRepository.save(country)
```

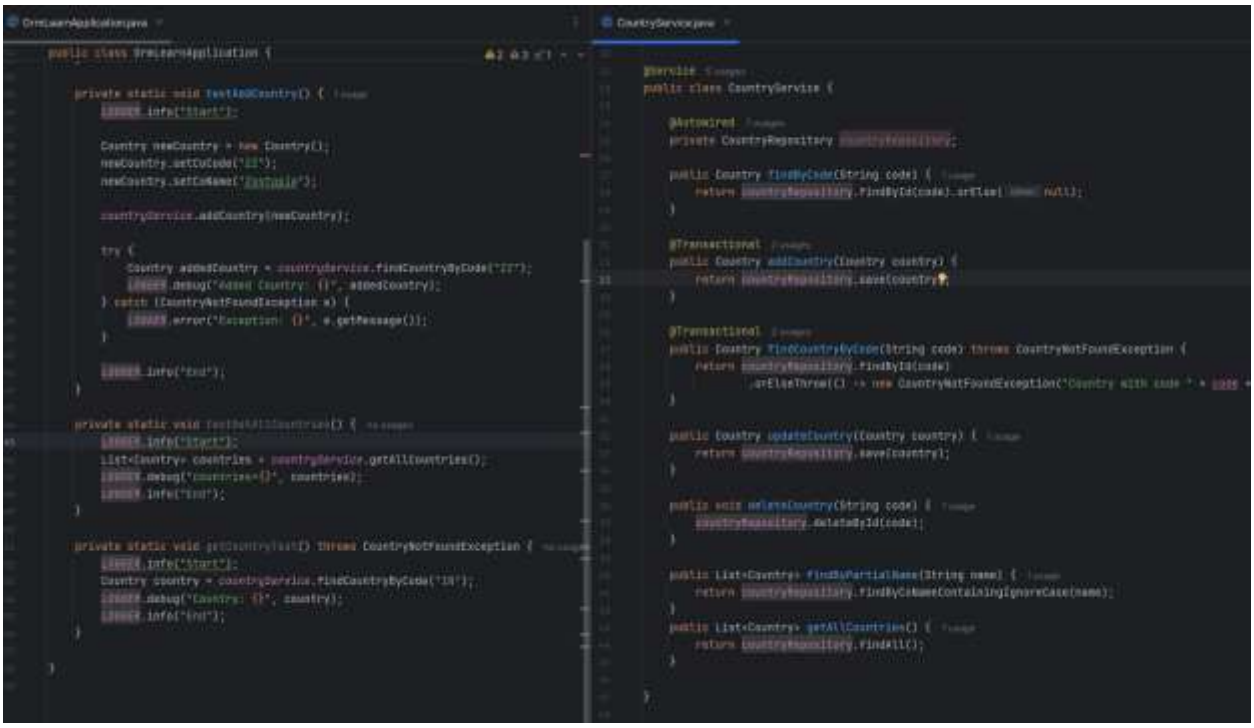
- Include new testAddCountry() method in OrmLearnApplication. Perform steps below:
  - Create new instance of country with a new code and name
  - Call countryService.addCountry() passing the country created in the previous step.
  - Invoke countryService.findCountryByCode() passing the same code used when adding a new country
  - Check in the database if the country is added

Implementation:

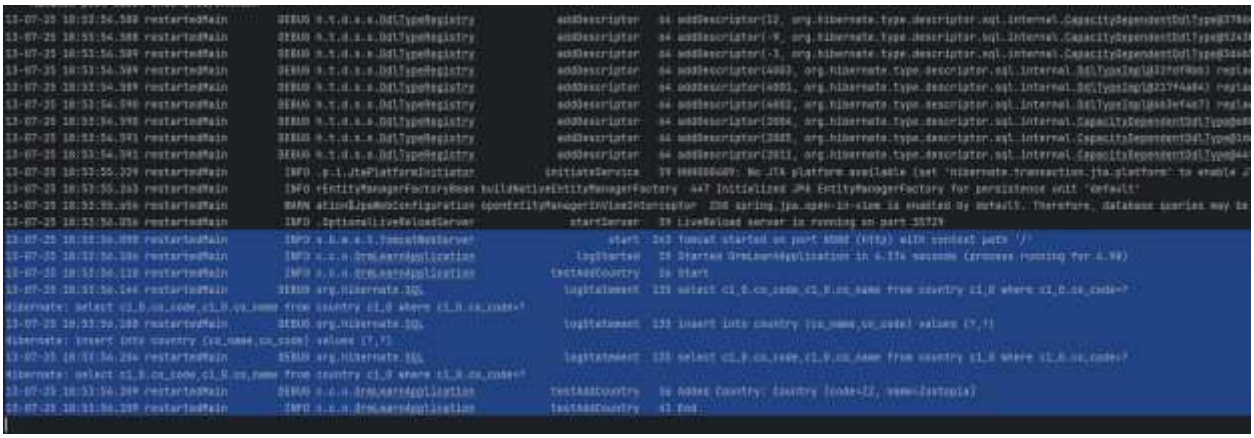
File Structure:



Code:



Output:



```
263 •    SELECT * FROM country WHERE co_code = 'ZZ';
264
265
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

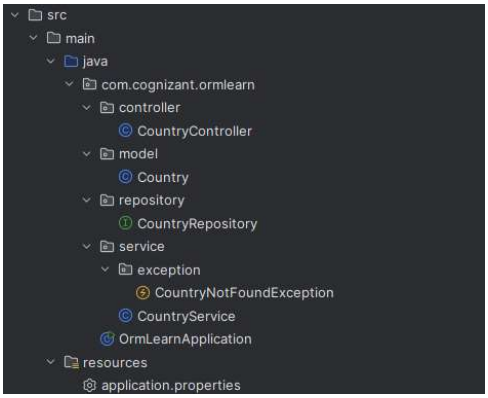
co_code	co_name
ZZ	Zootopia
NULL	NULL

Hands on 8 - Update a country based on code (Neither Mandatory nor Additional)

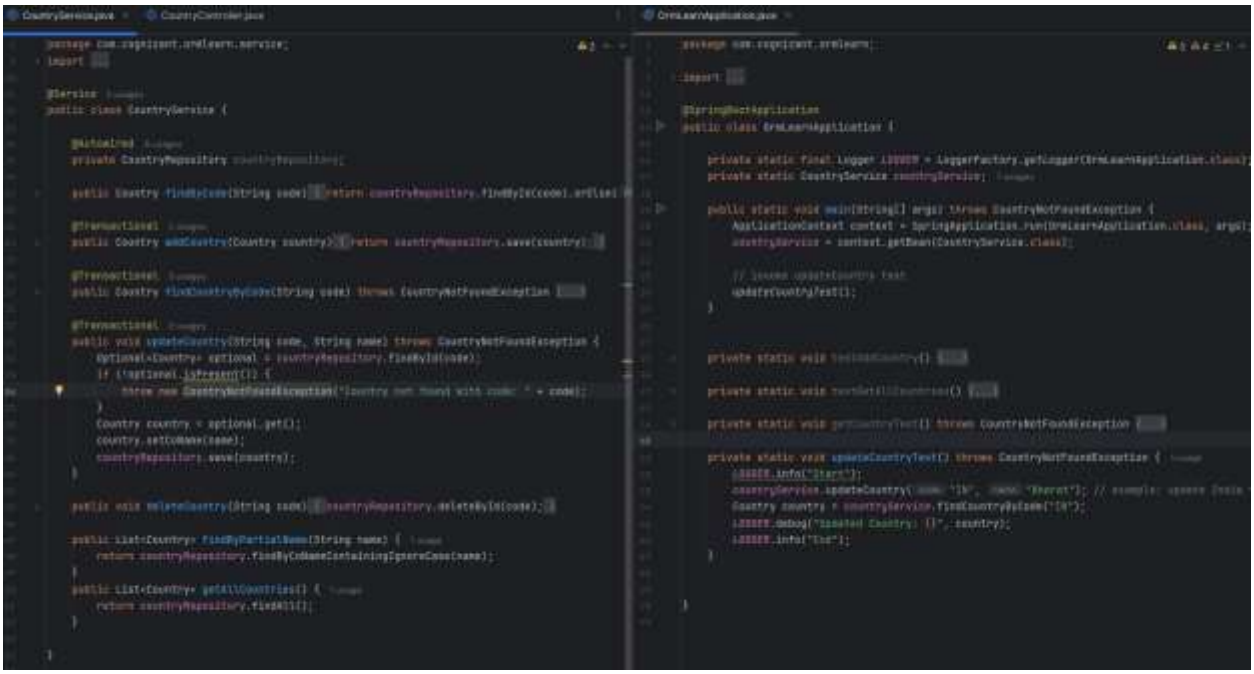
- Create a new method updateCountry() in CountryService with parameters code and name. Annotate this method with @Transactional. Implement following steps in this method.
    - Get the reference of the country using findById() method in repository
    - In the country reference obtained, update the name 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.

Implementation:

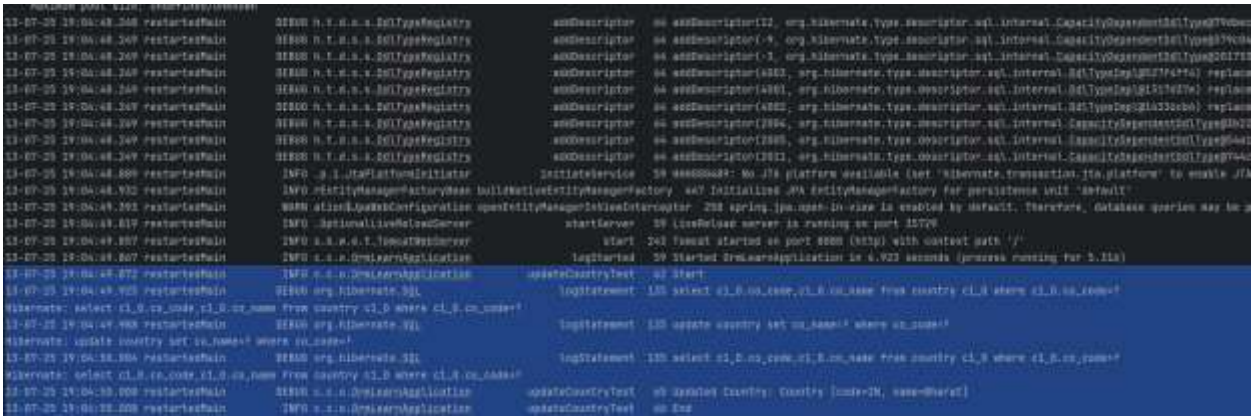
File Structure:



Code:



Output:



```
13-07-25 19:04:48.869 restartedMain       INFO _g_.JtaPlatformInitiator          InitiateService  59 HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA
13-07-25 19:04:48.932 restartedMain       INFO ntityManagerFactoryBean buildNativeEntityManagerFactory  447 Initialized JPA EntityManagerFactory for persistence unit 'default'
13-07-25 19:04:49.393 restartedMain       WARN ationS$JpaWebConfiguration openEntityManagerInViewInterceptor  258 spring.jpa.open-in-view is enabled by default. Therefore, database queries may be p
13-07-25 19:04:49.819 restartedMain       INFO .OptionalLiveReloadServer              startServer  59 LiveReload server is running on port 35729
13-07-25 19:04:49.867 restartedMain       INFO o.s.b.w.e.t.TomcatWebServer                 start  243 Tomcat started on port 8080 (http) with context path '/'
13-07-25 19:04:49.867 restartedMain       INFO c.c.c.OrmLearnApplication           logStarted  59 Started OrmLearnApplication in 6.923 seconds (process running for 9.31s)
13-07-25 19:04:49.872 restartedMain       INFO c.c.c.OrmLearnApplication       updateCountryTest  42 Start
13-07-25 19:04:49.925 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=?
Hibernate: select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
13-07-25 19:04:49.968 restartedMain       DEBUG org.hibernate.SQL                 logStatement  135 update country set co_name=? where co_code=?
Hibernate: update country set co_name=? where co_code=?
13-07-25 19:04:50.004 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=?
Hibernate: select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
13-07-25 19:04:50.008 restartedMain       DEBUG c.c.c.OrmLearnApplication       updateCountryTest  45 Updated Country: Country [code=IN, name=Bharat]
13-07-25 19:04:50.008 restartedMain       INFO c.c.c.OrmLearnApplication       updateCountryTest  46 End
```

```
265 •   SELECT * FROM country WHERE co_code = 'IN';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

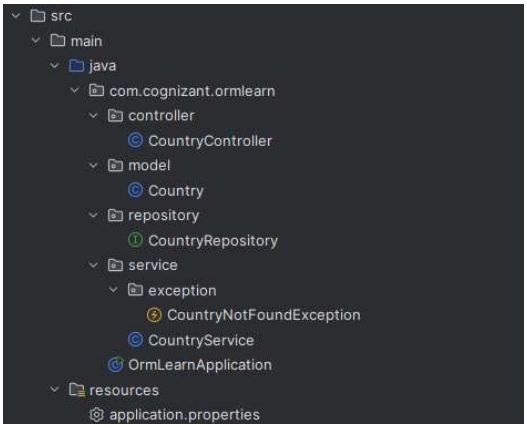
co_code	co_name
IN	Bharat
NULL	NULL

Hands on 9 - Delete a country based on code (Neither Mandatory nor Additional)

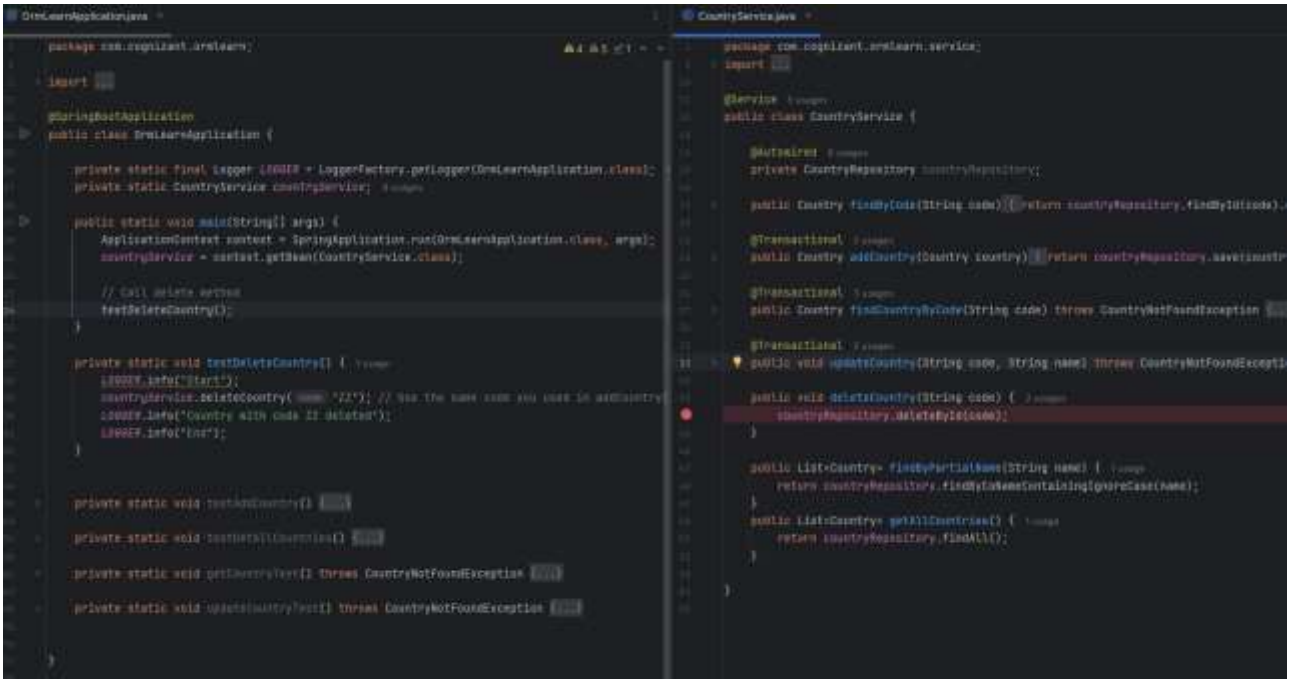
- Create new method deleteCountry() in CountryService. Annotate this method with @Transactional.
- In deleteCountry() method call deleteById() method of repository.
- Include new test method in OrmLearnApplication with following steps
    - Call the delete method based on the country code during the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

Implementation:

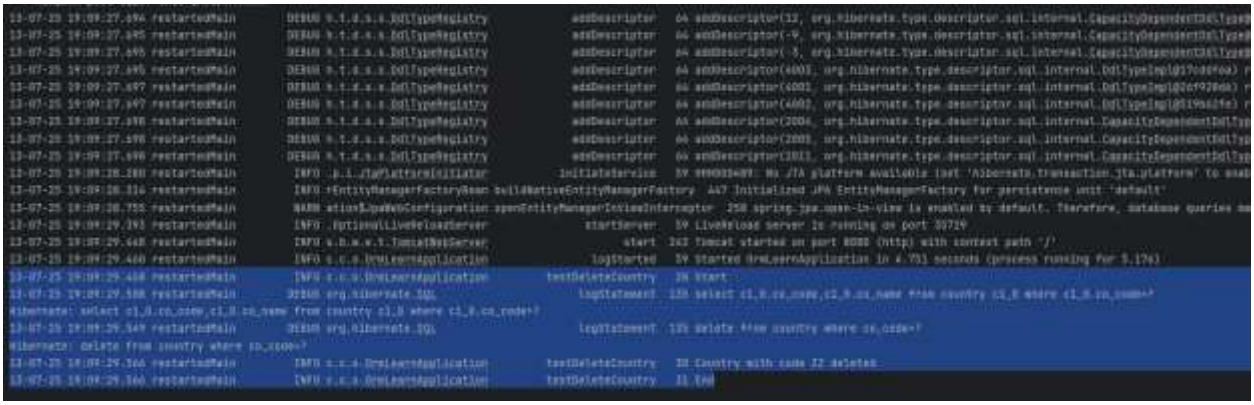
File Structure:



Code:



Output:


Hibernate: select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
Hibernate: delete from country where co_code=?


267 • SELECT * FROM country WHERE co_code = '22';
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
co_code
co_name
NULL
NULL