# Spring Data JPA with Spring Boot, Hibernate

# spring-data-jpa-handson

# Additional hands-on

**Hands on 5**

**Implement services for managing Country** 

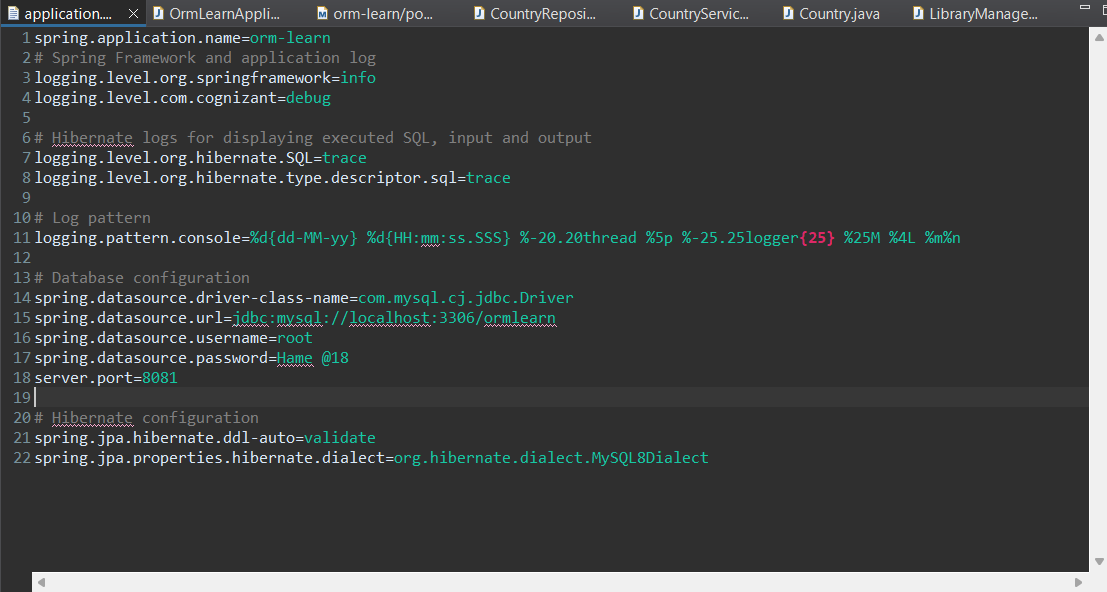
**Hibernate table creation configuration:**

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

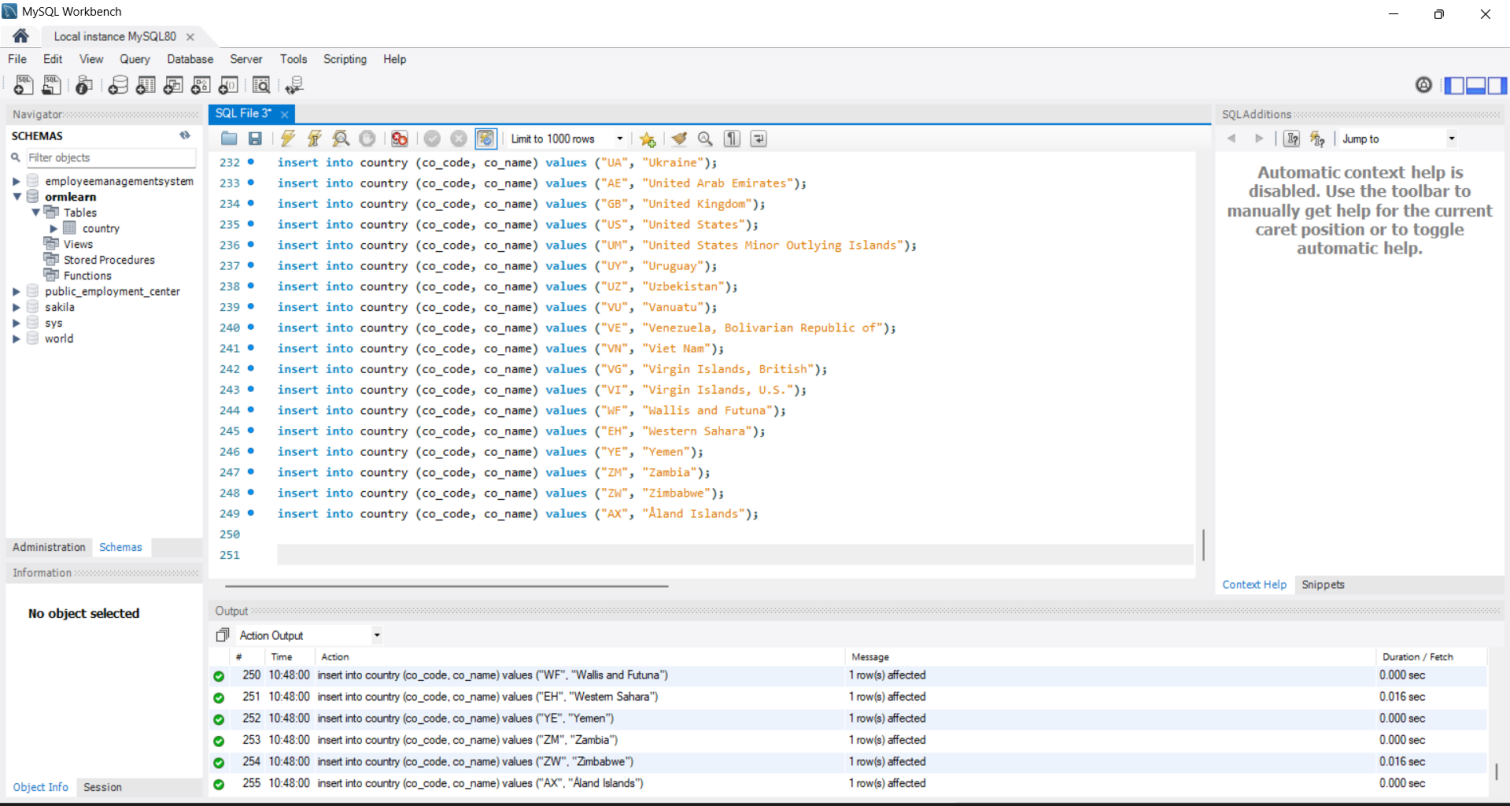
spring.jpa.hibernate.ddl-auto=validate

Moreover the ddl-auto defines how hibernate behaves if a specific table or column is not present in the database.

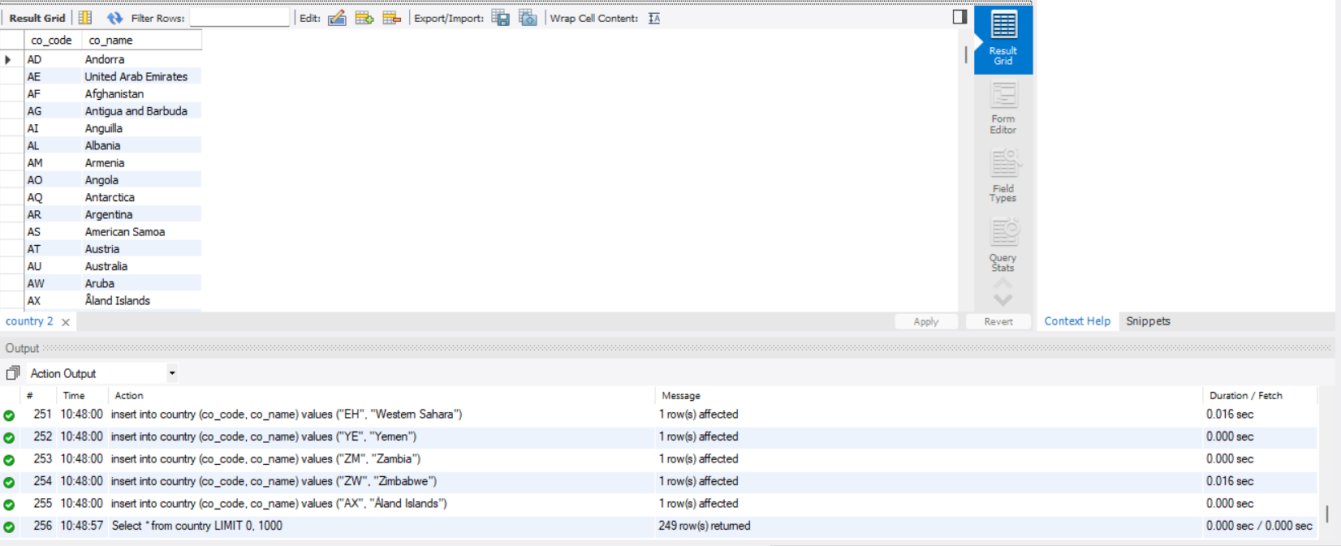
* + create - drops existing tables data and structure, then creates new tables
  + validate - check if the table and columns exist or not, throws an exception if a matching table or column is not found
  + update - if a table does not exists, it creates a new table; if a column does not exists, it creates a new column
  + create-drop - creates the table, once all operations are completed, the table is dropped



**Populate country table:**



*Select \* from country;*



**Hands on 6**

**Find a country based on country code**

1. **Create the CountryNotFoundException Class**

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

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

1. **Add findCountryByCode() Method in CountryService**

package com.cognizant.orm\_learn.service;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

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

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

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

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

}

1. **Repository class:**

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;

@Repository

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

1. **Add the Test Method in OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

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

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) throws CountryNotFoundException {

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

countryService = context.getBean(CountryService.class);

getCountryTest(); // Test invocation

}

private static void getCountryTest() throws CountryNotFoundException {

LOGGER.info("Start");

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

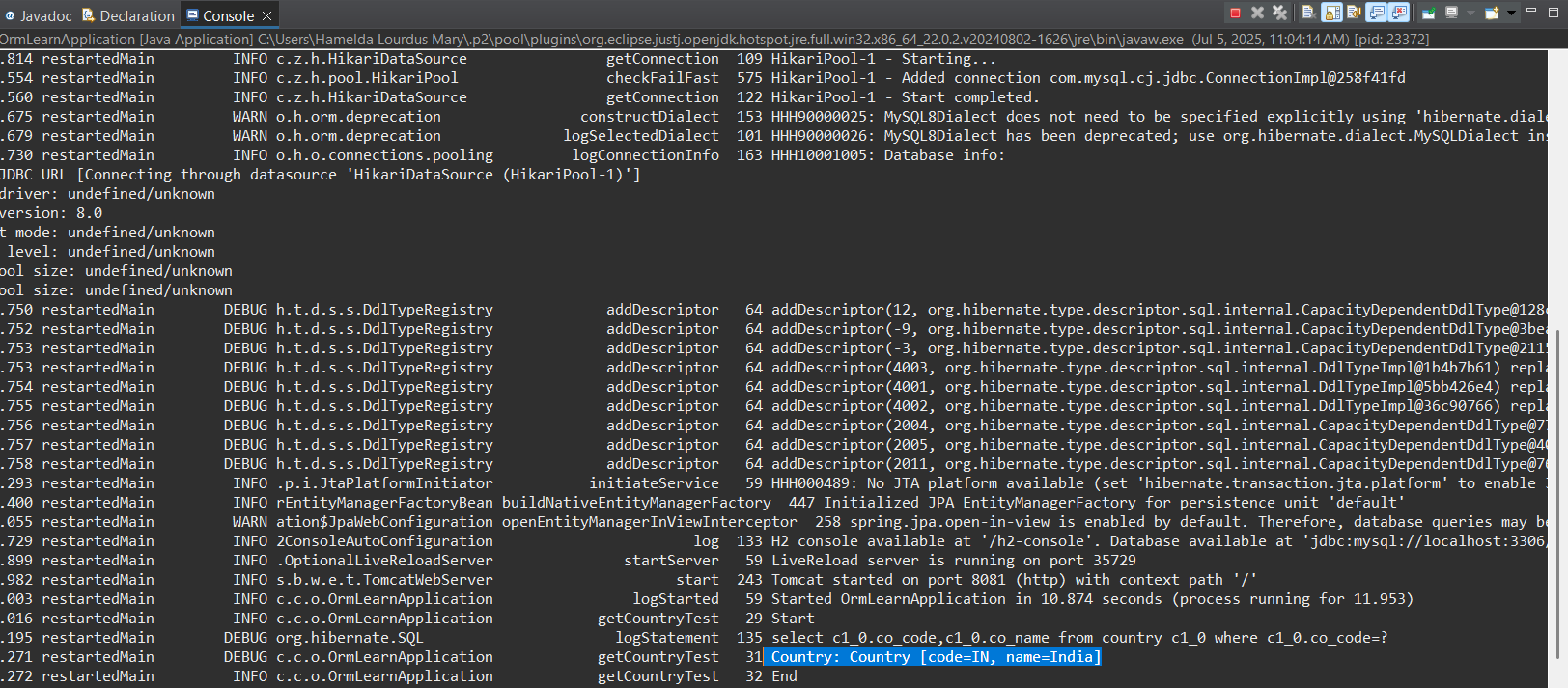
LOGGER.debug("Country: {}", country);

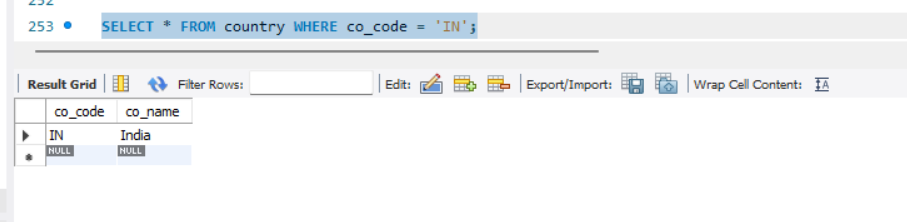
LOGGER.info("End");

}

}

**Output:**





**Notes on @Transactional:**

* To indicate that the annotated code should be executed within a transaction
* The @Transactional annotation tells Spring to **open a transaction** at the beginning of the method and **commit** it at the end.
* Even for **read-only operations**, it's a best practice to annotate service methods that interact with the database to ensure session management.
* Without @Transactional, Spring might not properly open a Hibernate session, especially if lazy-loaded associations exist.

**Hands on 7**

**Add a new country**

1. **Create new method in CountryService.**

package com.cognizant.orm\_learn.service;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

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

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

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

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

// Already existing method

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

return countryRepository.findById(code)

.orElseThrow(() -> new CountryNotFoundException("Country not found with code: " + code));

}

}

1. **Add TestAddCountry() in OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

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

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) throws CountryNotFoundException {

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

countryService = context.getBean(CountryService.class);

testAddCountry(); // Test invocation

}

private static void testAddCountry() {

LOGGER.info("Start");

Country newCountry = new Country();

newCountry.setCode("XY");

newCountry.setName("Xyland");

countryService.addCountry(newCountry);

try {

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

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

} catch (CountryNotFoundException e) {

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

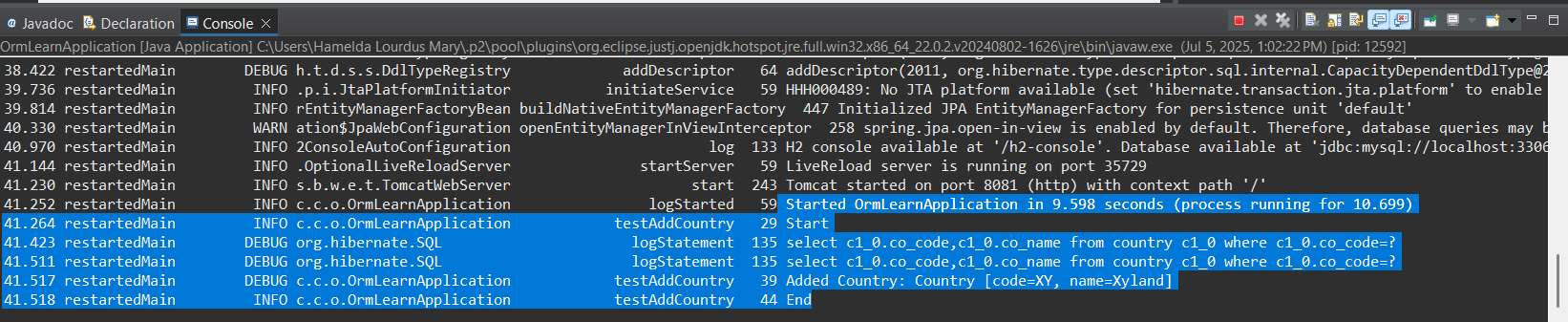
}

LOGGER.info("End");

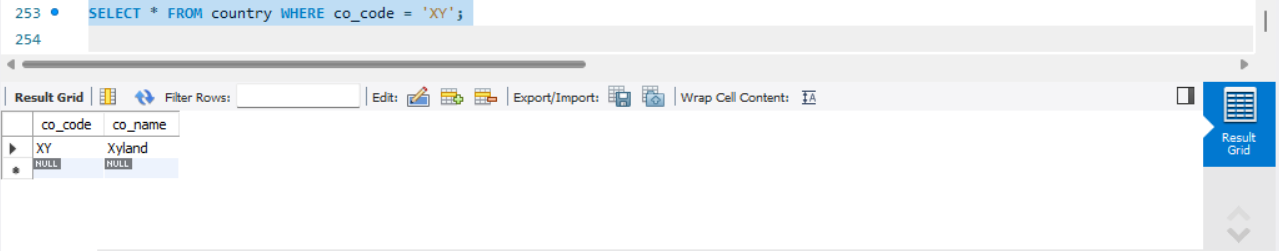
}

}

**Output:**



Check in the database if the country is added



**Hands on 8**

**Update a country based on code**

1. **Create a new method updateCountry() in CountryService**

@Transactional

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

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

if (!optional.isPresent()) {

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

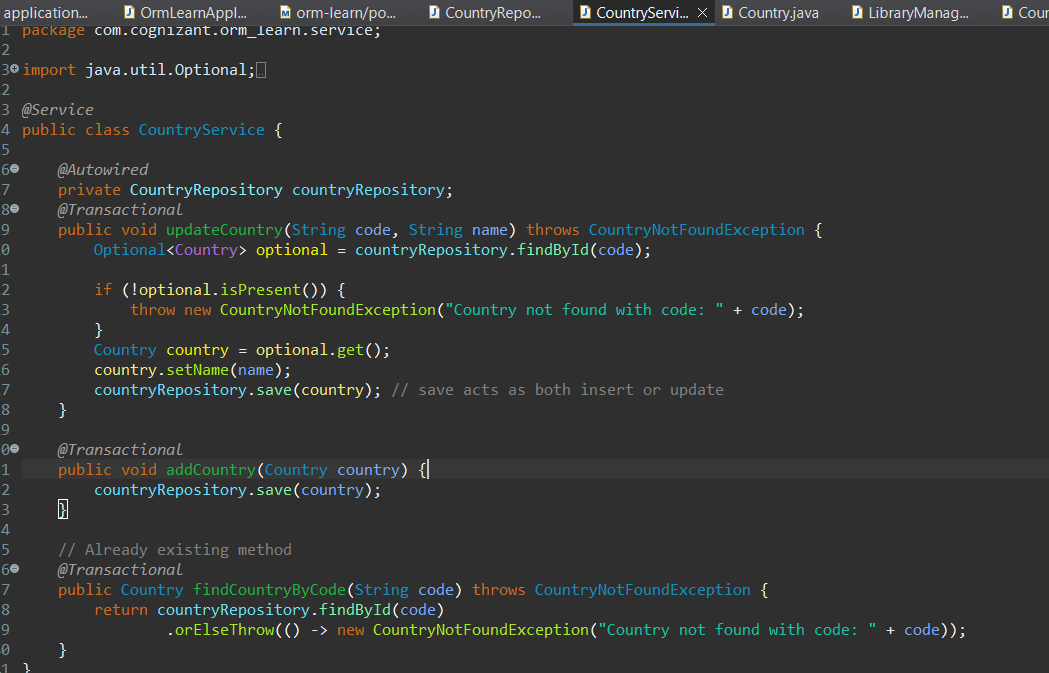
}

Country country = optional.get();

country.setName(name);

countryRepository.save(country); // save acts as both insert or update

}



1. **Include new test method in OrmLearnApplication**

private static void testUpdateCountry() {

LOGGER.info("Start");

try {

countryService.updateCountry("IN", "Bharat");

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

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

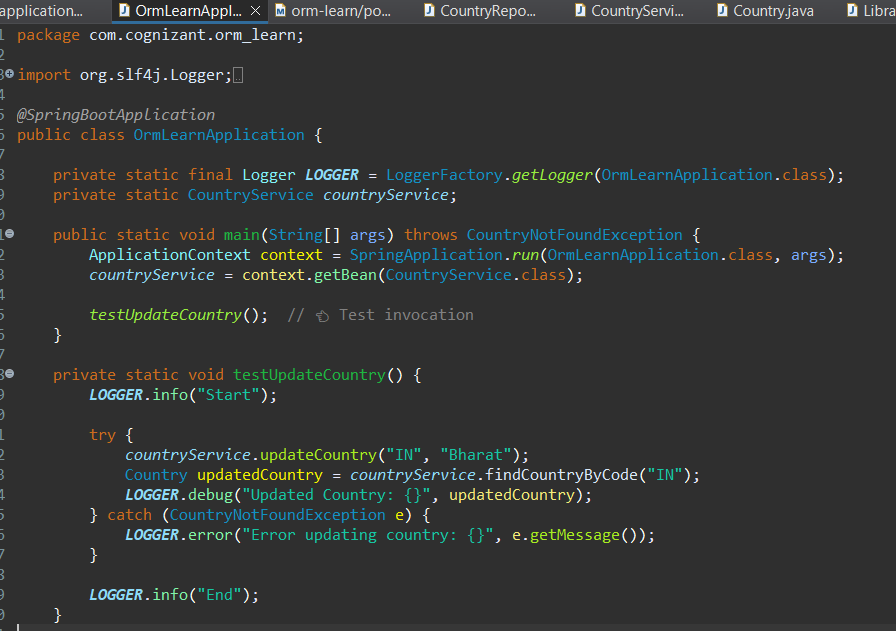
} catch (CountryNotFoundException e) {

LOGGER.error("Error updating country: {}", e.getMessage());

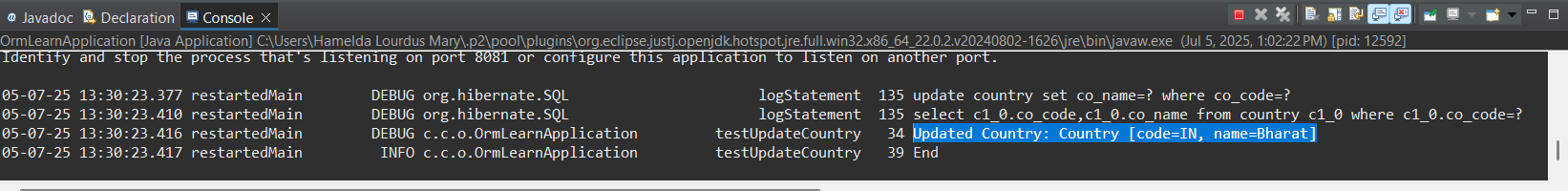
}

LOGGER.info("End");

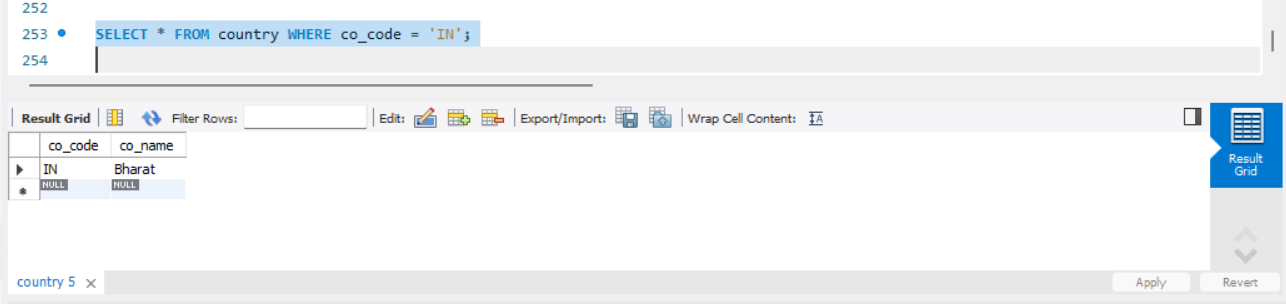
}



**Output:**



Check in database table if name is modified:



**Hands on 9**

**Delete a country based on code**

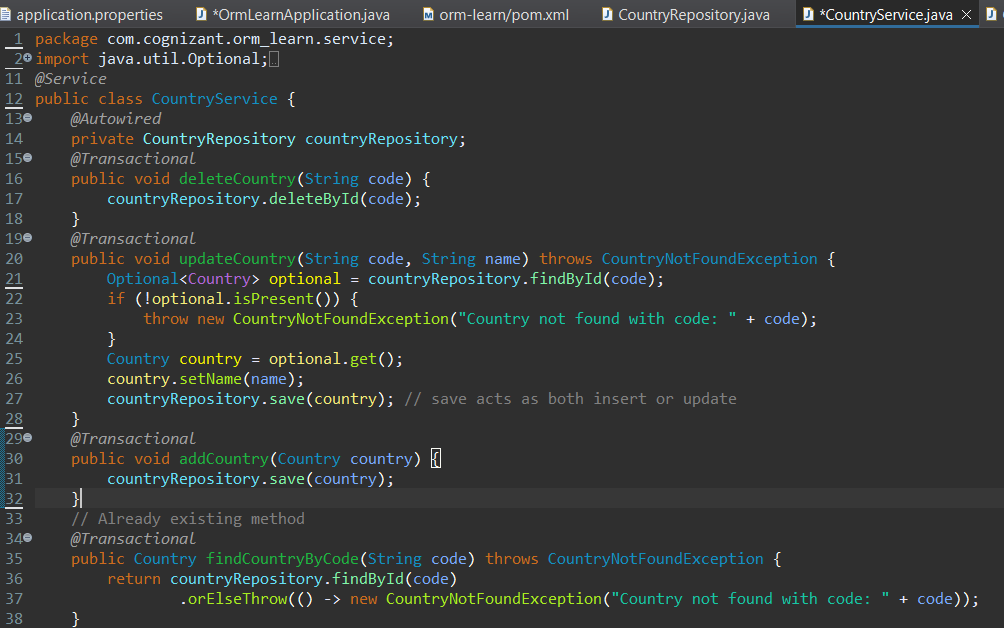
1. **Create new method deleteCountry() in CountryService:**

@Transactional

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}



1. **Include new test method in OrmLearnApplication:**

private static void testDeleteCountry() {

LOGGER.info("Start");

// Delete the country added in addCountry() test

countryService.deleteCountry("XY");

// Try fetching to confirm deletion

try {

Country deletedCountry = countryService.findCountryByCode("ZZ");

LOGGER.debug("Country Found (Should have been deleted): {}", deletedCountry);

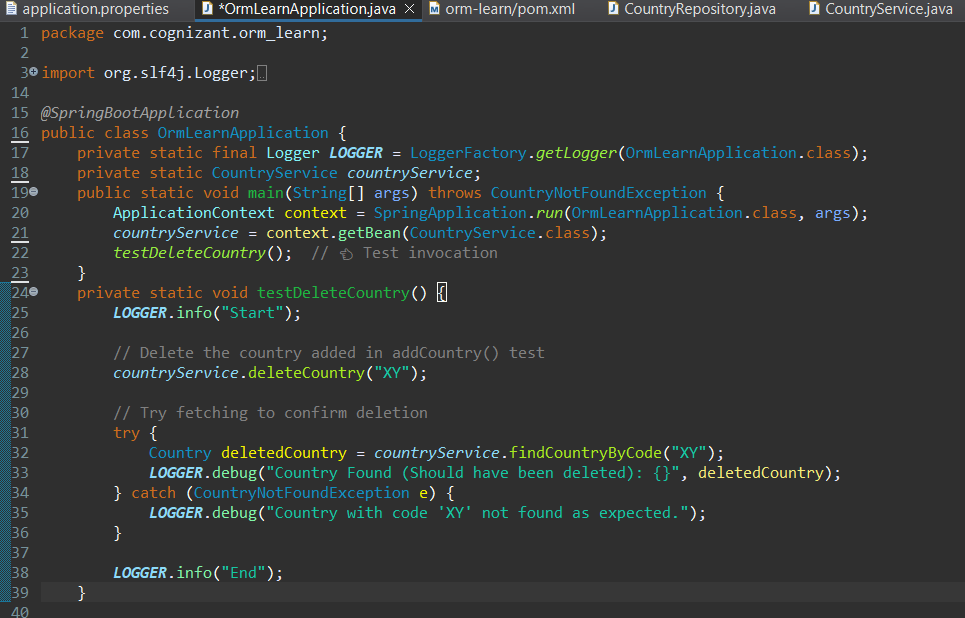
} catch (CountryNotFoundException e) {

LOGGER.debug("Country with code 'ZZ' not found as expected.");

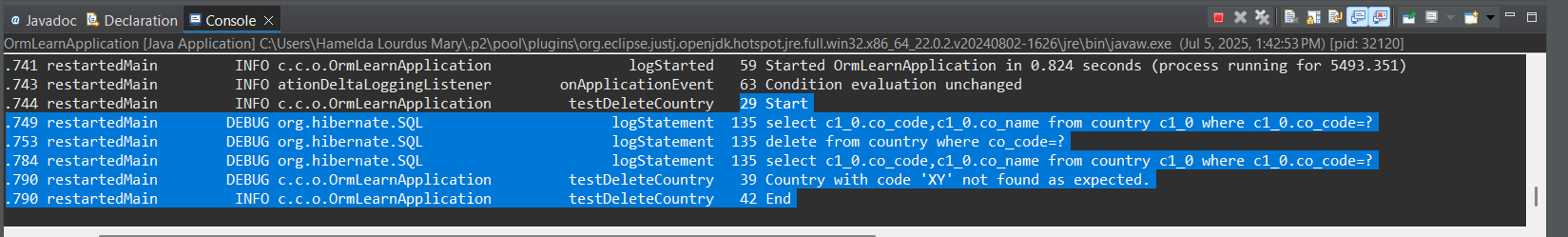
}

LOGGER.info("End");

}



**Output:**



Check in database if the country is deleted

