**Spring Data JPA with Spring Boot, Hibernate**

**Spring Data JPA - Quick Example**   
**Spring Boot Project: OrmLearnApplication – Spring Data JPA + MySQL**

**Software Requirements**

* **MySQL Server:** 8.0+
* **MySQL Workbench:** 8.0+
* **Eclipse IDE:** Eclipse IDE for Enterprise Java Developers 2019-03 R or later
* **Maven:** 3.6.2 or later
* **JDK:** Java 8 or above

**Step 1: Create a Spring Boot Project using Spring Initializr**

1. Go to <https://start.spring.io>
2. Fill out:
   * Group: com.cognizant
   * Artifact: orm-learn
   * Description: Demo project for Spring Data JPA and Hibernate
3. Select dependencies:
   * Spring Boot DevTools
   * Spring Data JPA
   * MySQL Driver
4. Click **Generate**, download the zip, extract it, and move it to your Eclipse workspace.
5. In Eclipse: File > Import > Maven > Existing Maven Project > Select extracted folder > Finish

**Step 2: Set Up MySQL**

1. Open MySQL Command Line Client
2. Create a schema:

CREATE SCHEMA ormlearn;

USE ormlearn;

1. Create and insert values in the country table:

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(50)

);

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

INSERT INTO country VALUES ('US', 'United States of America');

**Step 3: Configure application.properties**

Path: src/main/resources/application.properties

# 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

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

# MySQL 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

**Step 4: Create Package Structure and Java Classes**

**1. com.cognizant.ormlearn.model.Country.java**

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "name")

private String name;

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; }

@Override

public String toString() {

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

}

}

**2. com.cognizant.ormlearn.repository.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> {

}

**3. com.cognizant.ormlearn.service.CountryService.java**

package com.cognizant.ormlearn.service;

import java.util.List;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

1. **com.cognizant.ormlearn.OrmLearnApplication.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");

}

}

**Step 5: Add Dependencies to pom.xml**

Ensure the following dependencies are inside <dependencies>:

<!-- Spring Boot Starter -->

<dependency>

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

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

</dependency>

<!-- Spring Data JPA -->

<dependency>

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

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

</dependency>

<!-- MySQL Connector -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.0.33</version>

<scope>runtime</scope>

</dependency>

<!-- DevTools -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

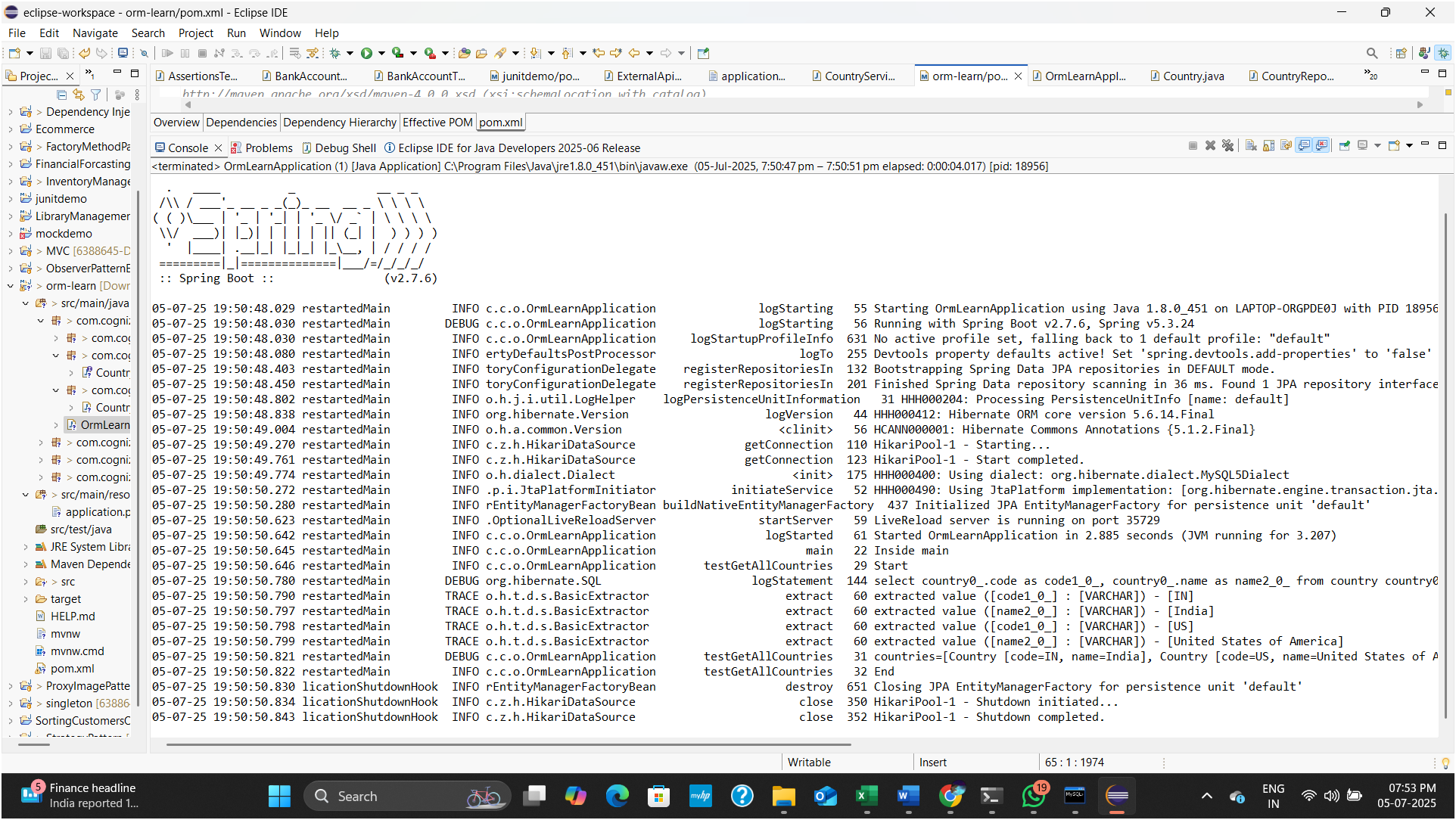
**Final Step: Run and Verify**

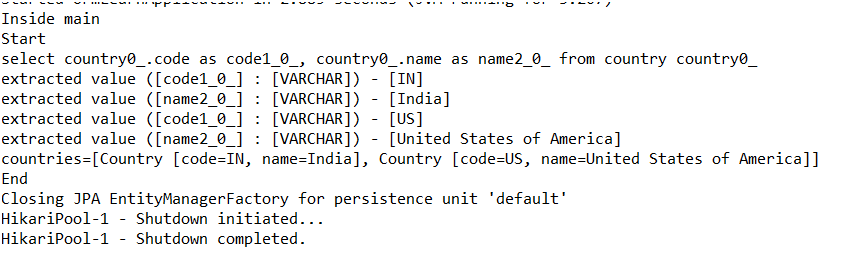
* Run the OrmLearnApplication.java class
* Check the Eclipse Console:
  + You should see Inside main printed.
  + You should see the list of countries printed from MySQL:

Country [code=IN, name=India]

Country [code=US, name=United States of America]

**Output:**





**Difference between JPA, Hibernate and Spring Data JPA**

**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

**Objective:**

To create a Spring Boot project with:

* Maven as the build tool
* Spring Web, Spring Data JPA, and MySQL Driver dependencies
* A working JPA configuration connecting to a MySQL database

**Step 1: Create the Maven Project**

* Open **Eclipse IDE**.
* Go to: File → New → Spring Starter Project.
* Fill in the project details:
  + **Name:** country-demo-01
  + **Type:** Maven
  + **Packaging:** Jar
  + **Java Version:** 21
  + **Group:** com.example
  + **Artifact:** country-demo-01
  + **Description:** Demo project for Spring Boot and JPA
  + **Package:** com.example.countrydemo01
* Click **Next**.

**Step 2: Select Required Dependencies**

Check the following:

* Spring Web
* Spring Data JPA
* MySQL Driver
* Spring Boot DevTools (optional, for auto restart during development)

Click **Finish**.

**Step 3: Maven pom.xml Configuration**

Ensure your pom.xml has the following dependencies:

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

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

<dependency>

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

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

</dependency>

<!-- MySQL Driver -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot DevTools (Optional) -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

**Step 4: Configure application.properties**

In src/main/resources/application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/countrydb

spring.datasource.username=root

spring.datasource.password=root

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

**Step5: Create Entity Class**

In com.example.countrydemo01.entity package, create Country.java:

package com.example.countrydemo01.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Country {

@Id

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**Step6: Create Repository Interface**

In com.example.countrydemo01.repository, create CountryRepository.java:

package com.example.countrydemo01.repository;

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

import com.example.countrydemo01.entity.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

**Step7: Application Class**

In com.example.countrydemo01, modify CountryDemo01Application.java to:

package com.example.countrydemo01;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountryDemo01Application {

public static void main(String[] args) {

SpringApplication.run(CountryDemo01Application.class, args);

}

}

**Step8: Verify Database Connection**

Make sure you have:

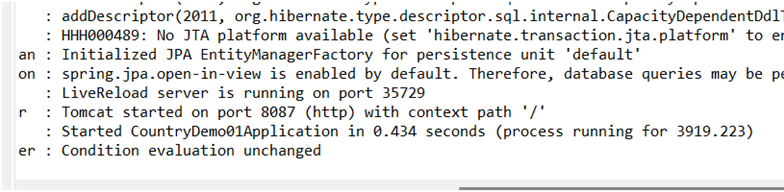
* MySQL service running
* A database named countrydb created:

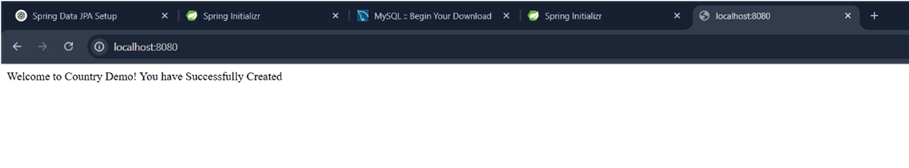
CREATE DATABASE countrydb;

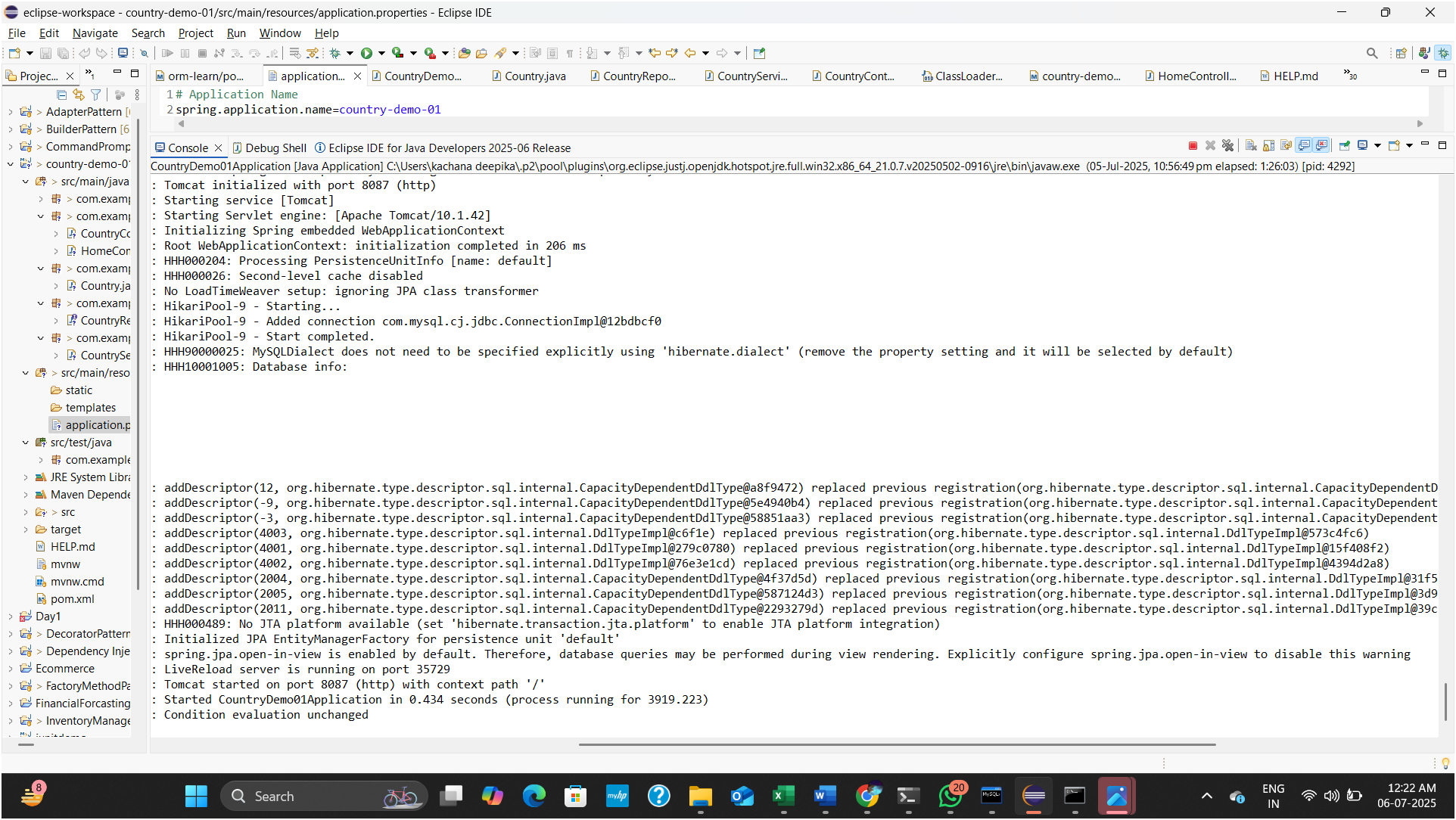
**Run the Application**

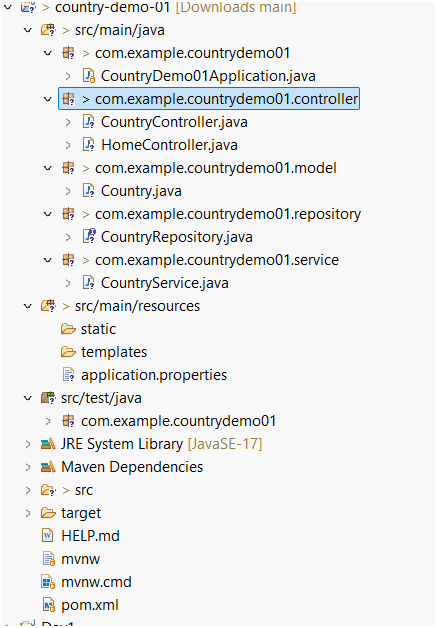
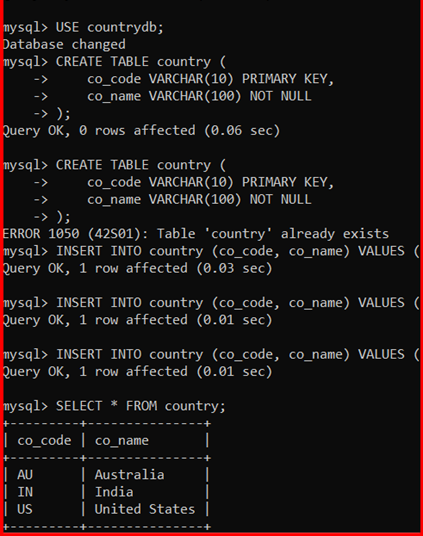
* Right-click on the project → Run As → Spring Boot App
* Check the console:
  + Tomcat started on port 8080
  + Hibernate is connected
  + HikariCP shows a successful connection to MySQL

**Output:**







**Hands on 5(Additional important hands-on)**

**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

**Step 1: Update application.properties**

To change the default port and configure Hibernate:

# src/main/resources/application.properties

server.port=8087

spring.datasource.url=jdbc:mysql://localhost:3306/countrydb

spring.datasource.username=root

spring.datasource.password=your\_password

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

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.show-sql=true

**Step 2: Create the Country Entity**

// src/main/java/com/example/countrydemo/entity/Country.java

package com.example.countrydemo.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String coCode;

private String coName;

// Constructors

public Country() {}

public Country(String coCode, String coName) {

this.coCode = coCode;

this.coName = 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;

}

}

**Step 3: Create the Repository Interface**

// src/main/java/com/example/countrydemo/repository/CountryRepository.java

package com.example.countrydemo.repository;

import com.example.countrydemo.entity.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByCoNameContainingIgnoreCase(String name);

}

**Step 4: Create the Service**

// src/main/java/com/example/countrydemo/service/CountryService.java

package com.example.countrydemo.service;

import com.example.countrydemo.entity.Country;

import com.example.countrydemo.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 getCountryByCode(String code) {

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

return result.orElse(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> searchCountriesByName(String name) {

return countryRepository.findByCoNameContainingIgnoreCase(name);

}

}

**Step 5: Create the REST Controller**

// src/main/java/com/example/countrydemo/controller/CountryController.java

package com.example.countrydemo.controller;

import com.example.countrydemo.entity.Country;

import com.example.countrydemo.service.CountryService;

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

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/{code}")

public Country getByCode(@PathVariable String code) {

return countryService.getCountryByCode(code);

}

@PostMapping

public Country add(@RequestBody Country country) {

return countryService.addCountry(country);

}

@PutMapping

public Country update(@RequestBody Country country) {

return countryService.updateCountry(country);

}

@DeleteMapping("/{code}")

public String delete(@PathVariable String code) {

countryService.deleteCountry(code);

return "Country deleted with code: " + code;

}

@GetMapping("/search")

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

return countryService.searchCountriesByName(name);

}

}

**Step 6: Application Main Class**

// src/main/java/com/example/countrydemo/CountryDemo01Application.java

package com.example.countrydemo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountryDemo01Application {

public static void main(String[] args) {

SpringApplication.run(CountryDemo01Application.class, args);

}

}

**Step7: Create the country table**

DROP TABLE IF EXISTS country;

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(100) NOT NULL

);

**Step8: Insert all countries**

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");

OUTPUT:

