**Mandotary Handson**

**1.Spring Data JPA - Quick Example**

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import com.cognizant.ormlearn.entity.Country;

import com.cognizant.ormlearn.service.CountryService;

*@SpringBootApplication*

public class OrmLearnApplication {

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

public static void main(String[] args) {

SpringApplication.*run*(OrmLearnApplication.class, args);

}

/\*\*

\* Runs automatically after the Spring context is ready.

\*/

*@Bean*

CommandLineRunner runAtStart(CountryService countryService) {

return args -> {

***LOGGER***.info("Start");

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

***LOGGER***.info("countries = {}", countries); // changed DEBUG → INFO

***LOGGER***.info("End");

};

}

}

**Country.java**

package com.cognizant.ormlearn.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String code;

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.entity.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.repository.CountryRepository;

import com.cognizant.ormlearn.entity.Country;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=orm\_learn

spring.datasource.password=madhu0305

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=orm\_learn

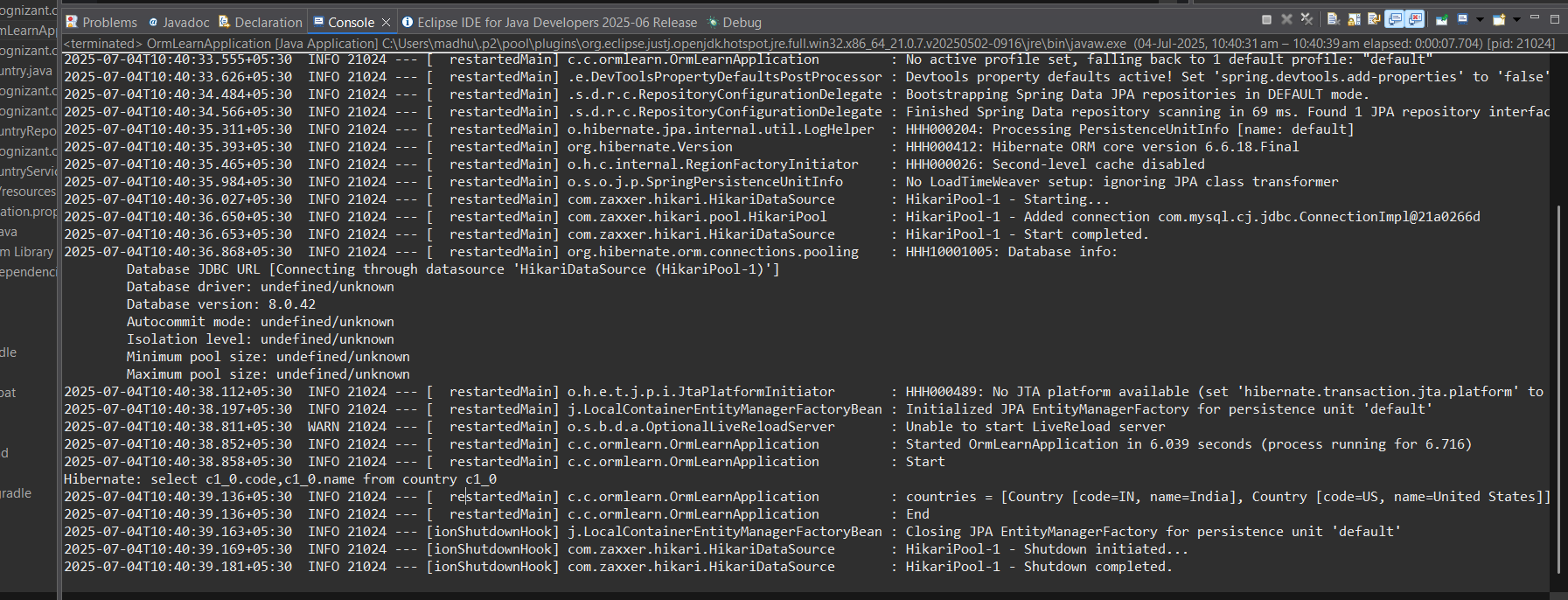
spring.datasource.password=madhu0305

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**Output:**

****

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

**JPA (Java Persistence API):**

* JPA is a **specification** (defined in JSR 338) used for **mapping Java objects to relational databases**.
* It defines **interfaces and annotations** like @Entity, @Id, and @Table.
* JPA **does not provide implementation**. It only defines how ORM should work in Java.
* Example providers that implement JPA are **Hibernate**, **EclipseLink**, etc.

**2. Hibernate:**

* Hibernate is a **popular implementation** of the JPA specification.
* It is an **ORM (Object-Relational Mapping) tool** that provides the actual logic to persist, update, and retrieve Java objects from a database.
* Hibernate can be used **with or without** Spring.
* Requires manual handling of **SessionFactory**, **Transaction**, etc., if used directly.

**3. Spring Data JPA:**

* Spring Data JPA is a **Spring framework abstraction** built on top of JPA.
* It **simplifies database operations** by reducing boilerplate code.
* Provides interfaces like JpaRepository or CrudRepository, which automatically generate standard CRUD operations.
* It uses Hibernate (or any JPA provider) under the hood.
* Offers additional features like **query derivation from method names**, pagination, and auditing.

**Using Hibernate (manual):**

* java
* CopyEdit
* Session session = sessionFactory.openSession();
* Transaction tx = session.beginTransaction();
* session.save(employee);
* tx.commit();

**Using Spring Data JPA:**

* java
* CopyEdit
* @Autowired
* private EmployeeRepository employeeRepository;
* employeeRepository.save(employee);

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification | Implementation of JPA | Abstraction layer on top of JPA |
| **Provides** | API/annotations only | ORM features + native SQL, HQL, caching | Repository interfaces and query methods |
| **Requires Implementation** | Yes – needs a provider like Hibernate | No – it's already a provider | No – uses JPA provider (like Hibernate) underneath |
| **Boilerplate Code** | Requires EntityManager, transactions manually | Slightly better but still requires code for common operations | Reduces boilerplate via JpaRepository, CrudRepository etc. |
| **Integration in Spring** | Manual configuration required | Can be configured easily | Auto-configured in Spring Boot |
| **Query Support** | JPQL | JPQL + HQL + Criteria API | JPQL + method names + @Query support |
| **Ease of Use** | Intermediate | Moderate | Very high |

**Bottom of Form**

**Additional important hands-on**

**Hands on 5 Implement services for managing Country**

**CountryApplication.java**

package com.example.country;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CountryApplication {

public static void main(String[] args) {

SpringApplication.run(CountryApplication.class, args);

}

}

**CountryController.java**

package com.example.country.controller;

import java.util.List;

import org.springframework.http.HttpStatus;

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

import com.example.country.model.Country;

import com.example.country.service.CountryService;

@RestController

@RequestMapping("/api/countries")

public class CountryController {

private final CountryService service;

public CountryController(CountryService service) { this.service = service; }

@GetMapping("/{code}")

public Country get(@PathVariable String code) {

return service.getByCode(code);

}

@PostMapping

@ResponseStatus(HttpStatus.CREATED)

public Country add(@RequestBody Country country) {

return service.add(country);

}

@PutMapping("/{code}")

public Country update(@PathVariable String code,

@RequestBody Country country) {

return service.update(code, country);

}

@DeleteMapping("/{code}")

@ResponseStatus(HttpStatus.NO\_CONTENT)

public void delete(@PathVariable String code) {

service.delete(code);

}

@GetMapping

public List<Country> search(@RequestParam(required = false) String name) {

return (name == null || name.isBlank())

? service.getAll()

: service.searchByName(name);

}

}

**Country.java**

package com.example.country.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", length = 2)

private String code;

@Column(name = "co\_name", nullable = false)

private String name;

public Country() {} // JPA needs this

public Country(String code, String name) {

this.code = code;

this.name = 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; }

}

**CountryService.java**

package com.example.country.service;

import java.util.List;

import com.example.country.model.Country;

public interface CountryService {

Country getByCode(String code);

Country add(Country country);

Country update(String code, Country country);

void delete(String code);

// NEW

List<Country> getAll();

List<Country> searchByName(String namePart);

}

**CountryRepository.java**

package com.example.country.repository;

import java.util.List;

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

import com.example.country.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String namePart);

}

**CountryServuceImpl.java**

package com.example.country.service;

import java.util.List;

import java.util.NoSuchElementException;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.example.country.model.Country;

import com.example.country.repository.CountryRepository;

@Service

@Transactional

public class CountryServiceImpl implements CountryService {

private final CountryRepository repo;

public CountryServiceImpl(CountryRepository repo) { this.repo = repo; }

@Override

public Country getByCode(String code) {

return repo.findById(code.toUpperCase())

.orElseThrow(() -> new NoSuchElementException("Country not found"));

}

@Override

public Country add(Country c) { return repo.save(c); }

@Override

public Country update(String code, Country changes) {

Country existing = getByCode(code);

existing.setName(changes.getName());

return repo.save(existing);

}

@Override

public void delete(String code) { repo.deleteById(code.toUpperCase()); }

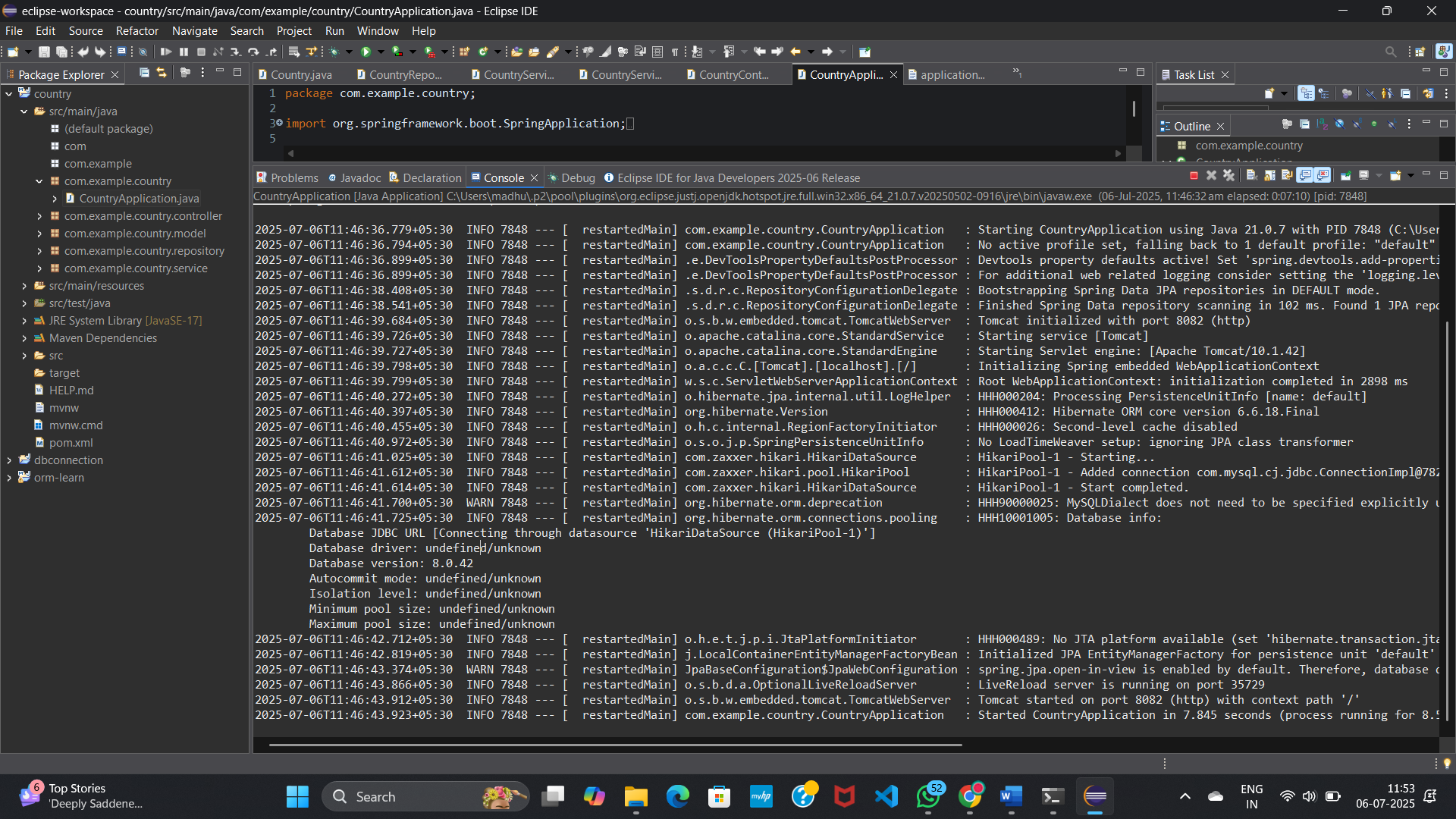
// NEW

@Override public List<Country> getAll() { return repo.findAll(); }

@Override public List<Country> searchByName(String n) { return repo.findByNameContainingIgnoreCase(n); }

}

**OUTPUT:**

****

**Find a country based on country code**

**CountryNotFoundException.java**

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**countryApplication.java**

package com.example.country;

import com.example.country.model.Country;

import com.example.country.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootApplication

public class CountryApplication implements CommandLineRunner {

// If using Lombok @Slf4j, remove the next two lines.

private static final Logger log = LoggerFactory.getLogger(CountryApplication.class);

@Autowired // injected by Spring

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(CountryApplication.class, args);

}

@Override

public void run(String... args) {

log.info("=== getCountryByCodeTest: START ===");

try {

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

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

} catch (CountryNotFoundException e) {

log.error("Exception caught: {}", e.getMessage());

}

log.info("=== getCountryByCodeTest: END ===");

}

}

**COuntryController.java**

package com.example.country.controller;

import java.util.List;

import org.springframework.http.HttpStatus;

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

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import com.example.country.model.Country;

import com.example.country.service.CountryService;

@RestController

@RequestMapping("/api/countries")

public class CountryController {

private final CountryService service;

public CountryController(CountryService service) {

this.service = service;

}

@GetMapping("/{code}")

public Country get(@PathVariable String code) throws CountryNotFoundException {

return service.findCountryByCode(code); // updated method name

}

@ResponseStatus(HttpStatus.CREATED)

public Country add(@RequestBody Country country) {

return service.add(country);

}

@PutMapping("/{code}")

public Country update(@PathVariable String code,

@RequestBody Country country) {

return service.update(code, country);

}

@DeleteMapping("/{code}")

@ResponseStatus(HttpStatus.NO\_CONTENT)

public void delete(@PathVariable String code) {

service.delete(code);

}

@GetMapping

public List<Country> search(@RequestParam(required = false) String name) {

return (name == null || name.isBlank())

? service.getAll()

: service.searchByName(name);

}

}

**Country.java**

package com.example.country.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", length = 2)

private String code;

@Column(name = "co\_name", nullable = false)

private String name;

public Country() {} // JPA needs this

public Country(String code, String name) {

this.code = code;

this.name = 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; }

}

**CountryRepositoary.java**

package com.example.country.repository;

import java.util.List;

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

import com.example.country.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String namePart);

}

**CountryService.java**

package com.example.country.service;

import java.util.List;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import com.example.country.model.Country;

public interface CountryService {

/\*\* Find a country by primary key (code). \*/

Country findCountryByCode(String code) throws CountryNotFoundException;

Country add(Country country);

Country update(String code, Country country);

void delete(String code);

List<Country> getAll();

List<Country> searchByName(String namePart);

}

**CountryServiceImply.java**

package com.example.country.service;

import java.util.List;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import com.example.country.model.Country;

import com.example.country.repository.CountryRepository; // make sure this import matches your package

@Service

public class CountryServiceImpl implements CountryService {

private final CountryRepository countryRepository;

public CountryServiceImpl(CountryRepository countryRepository) {

this.countryRepository = countryRepository;

}

@Override

@Transactional(readOnly = true)

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

return countryRepository.findById(countryCode)

.orElseThrow(() -> new CountryNotFoundException(

"Country with code " + countryCode + " not found"));

}

@Override

@Transactional

public Country add(Country country) {

return countryRepository.save(country);

}

@Override

@Transactional

public Country update(String code, Country updated) {

updated.setCode(code); // keep primary key stable

return countryRepository.save(updated);

}

@Override

@Transactional

public void delete(String code) {

countryRepository.deleteById(code);

}

@Override

@Transactional(readOnly = true)

public List<Country> getAll() {

return countryRepository.findAll();

}

@Override

@Transactional(readOnly = true)

public List<Country> searchByName(String namePart) {

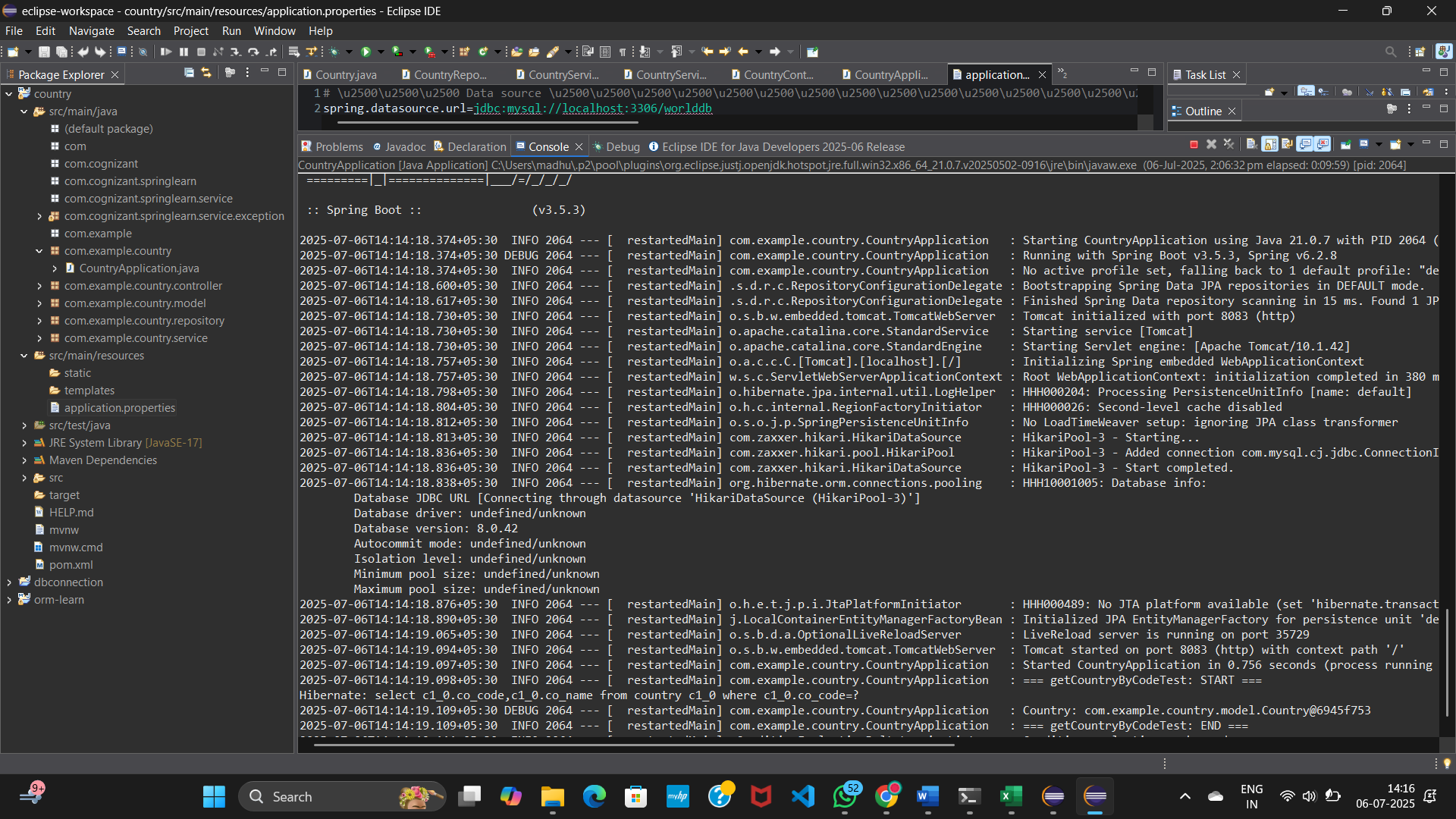
return countryRepository.findByNameContainingIgnoreCase(namePart);

// create a query method in CountryRepository or use @Query if not yet present

}

}

**OUTPUT:**

****

**Add a new country**

**CountryApplication.java**

package com.example.country;

import com.example.country.model.Country;

import com.example.country.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@SpringBootApplication*

public class CountryApplication implements CommandLineRunner {

private static final Logger ***log*** = LoggerFactory.*getLogger*(CountryApplication.class);

*@Autowired*

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.*run*(CountryApplication.class, args);

}

*@Override*

public void run(String... args) {

getCountryByCodeTest();

testAddCountry();

}

private void getCountryByCodeTest() {

***log***.info("=== getCountryByCodeTest: START ===");

try {

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

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

} catch (CountryNotFoundException e) {

***log***.error("Exception caught: {}", e.getMessage());

}

***log***.info("=== getCountryByCodeTest: END ===");

}

private void testAddCountry() {

***log***.info("=== testAddCountry: START ===");

// create a unique country

Country newCountry = new Country();

newCountry.setCode("ZZ"); // pick a code not used yet

newCountry.setName("Zanzibar");

countryService.addCountry(newCountry);

try {

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

***log***.debug("Fetched after insert: {}", fetched);

} catch (CountryNotFoundException e) {

***log***.error("Should not happen – just inserted ZZ", e);

}

***log***.info("=== testAddCountry: END ===");

}

}

**CountryController.java**

package com.example.country.controller;

import java.util.List;

import org.springframework.http.HttpStatus;

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

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import com.example.country.model.Country;

import com.example.country.service.CountryService;

@RestController

@RequestMapping("/api/countries")

public class CountryController {

private final CountryService service;

public CountryController(CountryService service) {

this.service = service;

}

@GetMapping("/{code}")

public Country get(@PathVariable String code) throws CountryNotFoundException {

return service.findCountryByCode(code); // updated method name

}

@PostMapping

@ResponseStatus(HttpStatus.CREATED)

public Country add(@RequestBody Country country) {

return service.add(country);

}

@PutMapping("/{code}")

public Country update(@PathVariable String code,

@RequestBody Country country) {

return service.update(code, country);

}

@DeleteMapping("/{code}")

@ResponseStatus(HttpStatus.NO\_CONTENT)

public void delete(@PathVariable String code) {

service.delete(code);

}

@GetMapping

public List<Country> search(@RequestParam(required = false) String name) {

return (name == null || name.isBlank())

? service.getAll()

: service.searchByName(name);

}

}

**Country.java**

package com.example.country.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", length = 2)

private String code;

@Column(name = "co\_name", nullable = false)

private String name;

public Country() {} // JPA needs this

public Country(String code, String name) {

this.code = code;

this.name = 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; }

}

**CountryService.java**

package com.example.country.repository;

import java.util.List;

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

import com.example.country.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String namePart);

}

**CountryServiceImpl.java**

package com.example.country.service;

import java.util.List;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import com.example.country.model.Country;

import com.example.country.repository.CountryRepository;

*@Service*

public class CountryServiceImpl implements CountryService {

private final CountryRepository countryRepository;

public CountryServiceImpl(CountryRepository countryRepository) {

this.countryRepository = countryRepository;

}

*@Override*

*@Transactional*(readOnly = true)

public Country findCountryByCode(String code) throws CountryNotFoundException {

return countryRepository.findById(code)

.orElseThrow(() -> new CountryNotFoundException(

"Country with code " + code + " not found"));

}

*@Override*

*@Transactional*

public Country add(Country country) {

return countryRepository.save(country);

}

*@Override*

*@Transactional*

public void addCountry(Country country) {

countryRepository.save(country);

}

*@Override*

*@Transactional*

public Country update(String code, Country updated) {

updated.setCode(code); // keep PK stable

return countryRepository.save(updated);

}

*@Override*

*@Transactional*

public void delete(String code) {

countryRepository.deleteById(code);

}

*@Override*

*@Transactional*(readOnly = true)

public List<Country> getAll() {

return countryRepository.findAll();

}

*@Override*

*@Transactional*(readOnly = true)

public List<Country> searchByName(String namePart) {

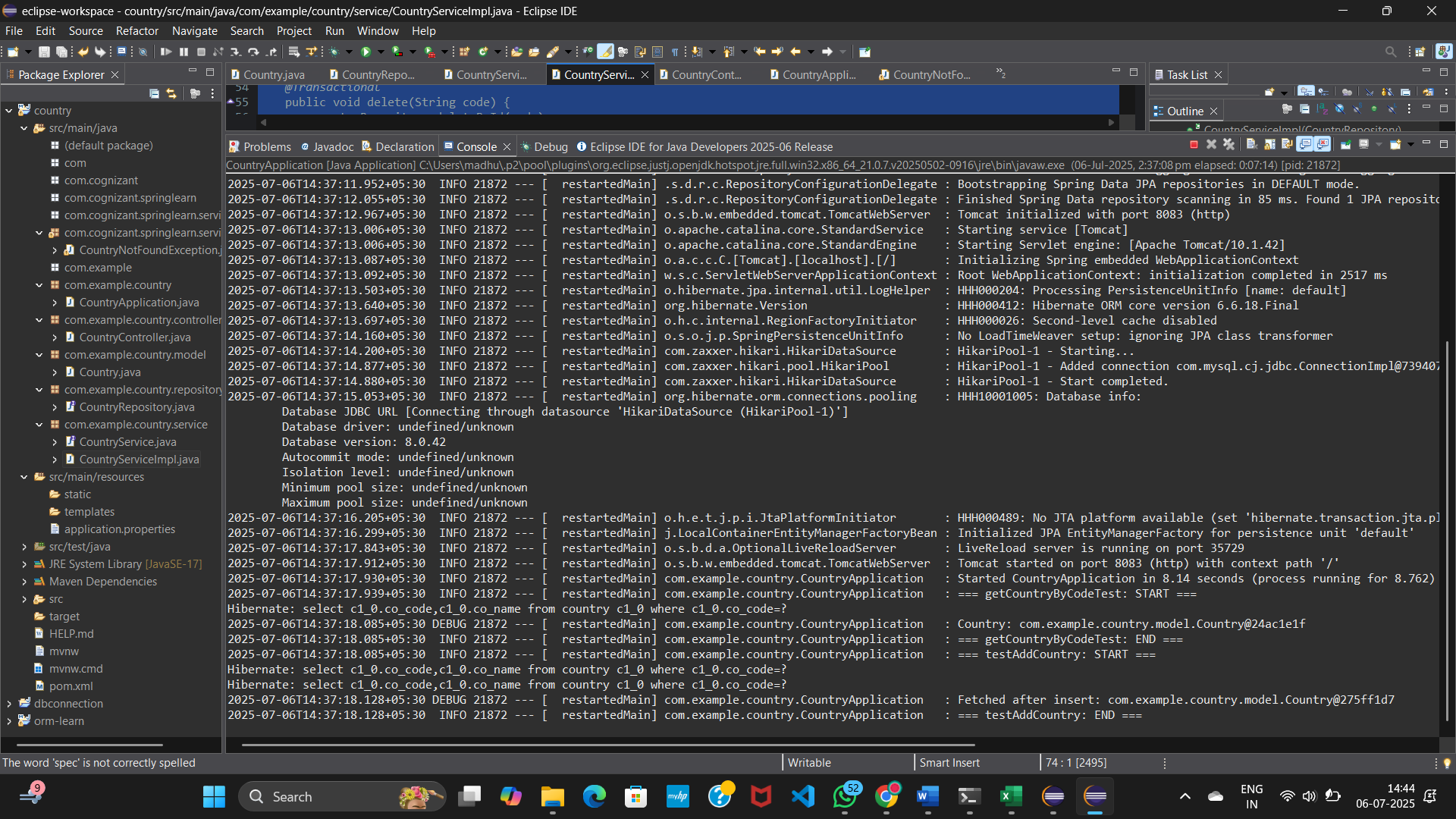
return countryRepository.findByNameContainingIgnoreCase(namePart);

// make sure this repository method exists, or define it with @Query

}

}

**OUTPUT:**

****