# SPRING\_DATA\_JPA

## Handson1-Spring Data JPA - Quick Example

#### Country.java

package com.cognizant.ormlearn.model; 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 = "co\_code") private String code;

@Column(name = "co\_name") private String name;

// Getters and Setters public String getCode() {

return code;

}

public void setCode(String code) { this.code = code;

}

public String getName() { return name;

}

public void setName(String name) { this.name = name;

}

// toString() @Override

public String toString() {

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

}

}

CountryRepository.java

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Country; @Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

## CountryService.java

package com.cognizant.ormlearn.service; import java.util.List;

import 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();

}

}

### 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");

}

}

MySQL

create database ormlearn; use ormlearn;

create table country (

co\_code varchar(2) primary key, co\_name varchar(50)

);

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

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

## Handson2-Hibernate XML Conflg implementation walk through

Employee.java

package com.example.hibernate; public class Employee {

private int id;

private String firstName; private String lastName; private int salary;

// Getters and Setters public int getId() {

return id;

}

public void setId(int id) { this.id = id;

}

public String getFirstName() { return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() { return lastName;

}

public void setLastName(String lastName) { this.lastName = lastName;

}

public int getSalary() { return salary;

}

public void setSalary(int salary) { this.salary = salary;

}

}

Employee.hbm.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN" "<http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd>">

<hibernate-mapping>

<class name="com.example.hibernate.Employee" table="EMPLOYEE">

<id name="id" column="ID">

<generator class="increment"/>

</id>

<property name="firstName" column="FIRST\_NAME"/>

<property name="lastName" column="LAST\_NAME"/>

<property name="salary" column="SALARY"/>

</class>

</hibernate-mapping>

hibernate.cfg.xml

<?xml version="1.0" encoding="UTF-8"?>

<!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 connection settings -->

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</ property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/your\_d atabase</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">your\_password</property>

<!-- SQL dialect -->

<property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</prop erty>

<!-- Show SQL in console -->

<property name="hibernate.show\_sql">true</property>

<!-- Mapping file -->

<mapping resource="Employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

MainApp.java

package com.example.hibernate;

import org.hibernate.Session;

import org.hibernate.SessionFactory; import org.hibernate.Transaction; import org.hibernate.cfg.Configuration;

import java.util.List;

public class MainApp {

public static void main(String[] args) {

// Load configuration and mappings Configuration cfg = new

Configuration().configure("hibernate.cfg.xml"); SessionFactory factory = cfg.buildSessionFactory();

Session session = factory.openSession(); Transaction tx = null;

try {

tx = session.beginTransaction();

// CREATE: Insert new employee Employee emp = new Employee(); emp.setFirstName("Alice"); emp.setLastName("Smith"); emp.setSalary(60000); session.save(emp);

System.out.println("Inserted: " + emp.getFirstName());

// READ: List all employees

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

for (Employee e : employees) {

System.out.println(e.getId() + ": " + e.getFirstName() + " " + e.getLastName());

}

// READ: Get employee by ID

Employee one = session.get(Employee.class, emp.getId()); System.out.println("Fetched by ID: " + one.getFirstName());

// DELETE: Remove the employee session.delete(one);

System.out.println("Deleted employee with ID: " + one.getId());

tx.commit();

} catch (Exception e) {

if (tx != null) tx.rollback(); e.printStackTrace();

} finally { session.close(); factory.close();

}

}

}

# Output

Inserted: Alice 101: Alice Smith

Fetched by ID: Alice

Deleted employee with ID: 101

###### Handson 3-Hibernate Annotation Conflg implementation walk through

Employee.java

package com.example.hibernate; import javax.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

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getFirstName() { return firstName; }

public void setFirstName(String firstName) { this.firstName = firstName; }

public String getLastName() { return lastName; }

public void setLastName(String lastName) { this.lastName = lastName; }

public int getSalary() { return salary; }

public void setSalary(int salary) { this.salary = salary; }

}

### hibernate.cfg.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN" "<http://hibernate.sourceforge.net/hibernate-configuration->

3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Dialect: Tells Hibernate which SQL dialect to use -->

<property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</prop erty>

<!-- Database connection settings -->

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</ property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/your\_d atabase</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">your\_password</property>

<!-- Show SQL -->

<property name="hibernate.show\_sql">true</property>

<!-- Scan annotated classes -->

<mapping class="com.example.hibernate.Employee"/>

</session-factory>

</hibernate-configuration>

MainApp.java

package com.example.hibernate;

import org.hibernate.Session;

import org.hibernate.SessionFactory; import org.hibernate.Transaction; import org.hibernate.cfg.Configuration;

import java.util.List;

public class MainApp {

public static void main(String[] args) { Configuration cfg = new

Configuration().configure("hibernate.cfg.xml"); SessionFactory factory = cfg.buildSessionFactory(); Session session = factory.openSession(); Transaction tx = null;

try {

tx = session.beginTransaction();

// Create

Employee emp = new Employee();

emp.setFirstName("Bob"); emp.setLastName("Taylor"); emp.setSalary(45000); session.save(emp);

System.out.println("Inserted: " + emp.getFirstName());

// Read all

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

for (Employee e : employees) {

System.out.println(e.getId() + ": " + e.getFirstName() + " " + e.getLastName());

}

// Read by ID

Employee found = session.get(Employee.class, emp.getId()); System.out.println("Found: " + found.getFirstName());

// Delete session.delete(found);

System.out.println("Deleted: " + found.getId());

tx.commit();

} catch (Exception e) {

if (tx != null) tx.rollback(); e.printStackTrace();

} finally { session.close(); factory.close();

}

}

}

HandsOn-4 Difference between JPA, Hibernate and Spring Data JPA

Conceptual Differences

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | Specification (JSR 338) | Implementation of JPA | Abstraction layer over JPA (usually uses  Hibernate internally) |
| Implements JPA? | ❌ (just defines the contract) | ✅ Implements JPA | ❌ Uses a JPA provider like Hibernate |
| Boilerplate Code | Requires manual code | Slightly reduced boilerplate | ✅ Drastically reduced via interfaces and  annotations |
| Transaction Mgmt | Manual or using Spring | Manual or using Spring | ✅ Integrated with Spring’s @Transactional  support |
| Repositories | No built-in repo support | Uses DAO pattern manually | ✅ Uses JpaRepository, CrudRepository, etc.  for automatic query methods |

1. Hibernate Implementation

### Employee.java

package com.example.hibernate.model; public class Employee {

private int id;

private String firstName; private String lastName; private int salary;

public Employee() {}

public Employee(String fname, String lname, int salary) { this.firstName = fname;

this.lastName = lname;

this.salary = salary;

}

public int getId() { return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() { return firstName;

}

public void setFirstName(String fname) { this.firstName = fname;

}

public String getLastName() { return lastName;

}

public void setLastName(String lname) { this.lastName = lname;

}

public int getSalary() { return salary;

}

public void setSalary(int salary) { this.salary = salary;

}

}

EmployeeDao.java

package com.example.hibernate.dao;

import com.example.hibernate.model.Employee; import org.hibernate.HibernateException; import org.hibernate.Session;

import org.hibernate.SessionFactory; import org.hibernate.Transaction;

public class EmployeeDao { private SessionFactory factory;

public EmployeeDao(SessionFactory factory) { this.factory = factory;

}

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 Implementation

Employee.java

package com.example.springdatajpa.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee") public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;

@Column(name = "first\_name") private String firstName;

@Column(name = "last\_name") private String lastName;

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

public Employee() {}

public Employee(String firstName, String lastName, int salary) { this.firstName = firstName;

this.lastName = lastName;

this.salary = salary;

}

// Getters and setters omitted for brevity

}

EmployeeRepository.java

package com.example.springdatajpa.repository; import com.example.springdatajpa.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// Custom methods can be added like findByFirstName, etc.

}

### EmployeeService.java

package com.example.springdatajpa.service;

import com.example.springdatajpa.model.Employee;

import com.example.springdatajpa.repository.EmployeeRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) { employeeRepository.save(employee);

}

}

Application.java

package com.example.springdatajpa;

import com.example.springdatajpa.model.Employee;

import com.example.springdatajpa.service.EmployeeService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import javax.annotation.PostConstruct;

@SpringBootApplication public class Application {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) { SpringApplication.run(Application.class, args);

}

@PostConstruct

public void init() {

employeeService.addEmployee(new Employee("John", "Doe", 30000));

}

}

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);

}

}

Country.java

package com.example.country.entity;

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") private String name;

// Constructors 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;

}

}

CountryRepository.java

package com.example.country.repository; import com.example.country.entity.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String partialName);

}

### CountryService.java

package com.example.country.service;

import com.example.country.entity.Country;

import com.example.country.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 Optional<Country> getCountryByCode(String code) { return countryRepository.findById(code.toUpperCase());

}

public Country addCountry(Country country) { return countryRepository.save(country);

}

public Country updateCountry(String code, Country updatedCountry) {

return countryRepository.findById(code.toUpperCase())

.map(existing -> { existing.setName(updatedCountry.getName()); return countryRepository.save(existing);

})

.orElse(null);

}

public boolean deleteCountry(String code) {

if (countryRepository.existsById(code.toUpperCase())) { countryRepository.deleteById(code.toUpperCase()); return true;

}

return false;

}

public List<Country> searchCountries(String partialName) {

return countryRepository.findByNameContainingIgnoreCase(partialName);

}

public List<Country> getAllCountries() { return countryRepository.findAll();

}

}

###### CountryController.java

package com.example.country.controller;

import com.example.country.entity.Country;

import com.example.country.service.CountryService;

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

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

import java.util.List;

@RestController @RequestMapping("/api/countries") public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/{code}")

public ResponseEntity<Country> getCountry(@PathVariable String code) {

return countryService.getCountryByCode(code)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PostMapping

public ResponseEntity<Country> createCountry(@RequestBody Country country) {

return ResponseEntity.ok(countryService.addCountry(country));

}

@PutMapping("/{code}")

public ResponseEntity<Country> updateCountry(@PathVariable String code, @RequestBody Country country) {

Country updated = countryService.updateCountry(code, country);

return updated != null ? ResponseEntity.ok(updated) : ResponseEntity.notFound().build();

}

@DeleteMapping("/{code}")

public ResponseEntity<Void> deleteCountry(@PathVariable String code) {

return countryService.deleteCountry(code) ?

ResponseEntity.noContent().build() : ResponseEntity.notFound().build();

}

@GetMapping("/search")

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

return ResponseEntity.ok(countryService.searchCountries(name));

}

@GetMapping

public ResponseEntity<List<Country>> getAllCountries() { return ResponseEntity.ok(countryService.getAllCountries());

}

}

##### Hands on 6-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);

}

}

###### CountryService.java

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.entity.Country;

import com.cognizant.springlearn.repository.CountryRepository;

import com.cognizant.springlearn.service.exception.CountryNotFoundExcept ion;

import jakarta.transaction.Transactional;

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

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

}

###### OrmLearnApplication.java

package com.cognizant.springlearn;

import com.cognizant.springlearn.entity.Country;

import com.cognizant.springlearn.service.CountryService; import

com.cognizant.springlearn.service.exception.CountryNotFoundExcept ion;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication; import

org.springframework.boot.autoconfigure.SpringBootApplication; 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) {

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

countryService = context.getBean(CountryService.class);

getAllCountriesTest(); // Invoke test

}

private static void getAllCountriesTest() { LOGGER.info("Start");

try {

Country country = countryService.findCountryByCode("IN"); LOGGER.debug("Country: {}", country);

} catch (CountryNotFoundException e) { LOGGER.error("Exception: {}", e.getMessage());

}

LOGGER.info("End");

}

}

###### Hands on 7-Add a new country

Country.java

package com.cognizant.springlearn.entity; import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Country {

@Id

private String code;

private String name;

public String getCode() { return code;

}

public void setCode(String code) {

this.code = code.toUpperCase(); // ensure uppercase

}

public String getName() { return name;

}

public void setName(String name) { this.name = name;

}

@Override

public String toString() {

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

}

}

###### CountryRepository.java

package com.cognizant.springlearn.repository; import com.cognizant.springlearn.entity.Country;

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

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

##### CountryNotFoundException.java

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) { super(message);

}

}

###### CountryService.java

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.entity.Country;

import com.cognizant.springlearn.repository.CountryRepository;

import com.cognizant.springlearn.service.exception.CountryNotFoundExcept ion;

import jakarta.transaction.Transactional;

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

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

@Transactional

public void addCountry(Country country) { countryRepository.save(country);

}

}

###### OrmLearnApplication.java

package com.cognizant.springlearn;

import com.cognizant.springlearn.entity.Country;

import com.cognizant.springlearn.service.CountryService; import

com.cognizant.springlearn.service.exception.CountryNotFoundExcept ion;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication; import

org.springframework.boot.autoconfigure.SpringBootApplication; 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) {

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

countryService = context.getBean(CountryService.class);

testAddCountry();

}

private static void testAddCountry() { LOGGER.info("Start");

Country newCountry = new Country(); newCountry.setCode("JP"); newCountry.setName("Japan");

countryService.addCountry(newCountry);

try {

Country country = countryService.findCountryByCode("JP"); LOGGER.debug("Added Country: {}", country);

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

}

# output

Start

Added Country: Country [code=JP, name=Japan] End

Hands on 8-Update a country based on code

Country.java

package com.cognizant.springlearn.model;

import jakarta.persistence.Entity; import jakarta.persistence.Id;

@Entity

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.springlearn.repository;

import org.springframework.data.jpa.repository.JpaRepository; import com.cognizant.springlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

### CountryNotFoundException.java

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception { public CountryNotFoundException(String message) {

super(message);

}

}

##### CountryService.java

package com.cognizant.springlearn.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.springlearn.model.Country;

import com.cognizant.springlearn.repository.CountryRepository;

import com.cognizant.springlearn.service.exception.CountryNotFoundExcept ion;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

@Transactional

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

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

if (!result.isPresent()) {

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

}

Country country = result.get(); country.setName(newName); countryRepository.save(country);

}

}

###### OrmLearnApplication.java

package com.cognizant.springlearn;

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.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService; import

com.cognizant.springlearn.service.exception.CountryNotFoundExcept ion;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

testUpdateCountry(); // Call update test

}

private static void testUpdateCountry() { LOGGER.info("Start");

try {

// Updating country JP to "Nippon" countryService.updateCountry("JP", "Nippon");

// Retrieve and display updated country Country updatedCountry =

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

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

}

LOGGER.info("End");

}

}

# output

INFO Start

DEBUG Updated Country: Country [code=JP, name=Nippon] INFO End

###### Hands on 9-Delete a country based on code

Country.java

package com.cognizant.springlearn.model;

import javax.persistence.Entity; import javax.persistence.Id;

@Entity

public class Country {

@Id

private String code; private String name;

// Constructors public Country() {}

public Country(String code, String name) { this.code = code;

this.name = name;

}

// Getters & Setters public String getCode() {

return code;

}

public void setCode(String code) { this.code = code.toUpperCase();

}

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.springlearn.repository;

import org.springframework.data.jpa.repository.JpaRepository; import com.cognizant.springlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

CountryNotFoundException.java

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) { super(message);

}

}

###### CountryService.java

package com.cognizant.springlearn.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.springlearn.model.Country;

import com.cognizant.springlearn.repository.CountryRepository;

import com.cognizant.springlearn.service.exception.CountryNotFoundExcept ion;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

@Transactional

public void addCountry(Country country) { countryRepository.save(country);

}

@Transactional

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

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

if (!result.isPresent()) {

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

}

Country country = result.get();

country.setName(newName); countryRepository.save(country);

}

@Transactional

public void deleteCountry(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

countryRepository.deleteById(code.toUpperCase());

}

}

###### OrmLearnApplication.java

package com.cognizant.springlearn;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService; import

com.cognizant.springlearn.service.exception.CountryNotFoundExcept ion;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication; import

org.springframework.boot.autoconfigure.SpringBootApplication; 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) {

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

countryService = context.getBean(CountryService.class);

// Uncomment to test the features

// testAddCountry();

// testUpdateCountry(); testDeleteCountry();

}

private static void testAddCountry() { LOGGER.info("Start Add");

Country country = new Country("ZZ", "Zetland"); countryService.addCountry(country); LOGGER.debug("Country added: {}", country); LOGGER.info("End Add");

}

private static void testUpdateCountry() { LOGGER.info("Start Update");

try {

countryService.updateCountry("ZZ", "New Zetland"); LOGGER.info("Country updated successfully");

} catch (CountryNotFoundException e) { LOGGER.error("Exception: {}", e.getMessage());

}

LOGGER.info("End Update");

}

private static void testDeleteCountry() { LOGGER.info("Start Delete");

try {

countryService.deleteCountry("ZZ"); LOGGER.info("Country ZZ deleted successfully");

// Check if deleted countryService.findCountryByCode("ZZ");

} catch (CountryNotFoundException e) { LOGGER.error("Expected Exception: {}", e.getMessage());

}

LOGGER.info("End Delete");

}

}

# output

INFO Start Delete

INFO Country ZZ deleted successfully

ERROR Expected Exception: Country with code ZZ not found. INFO End Delete