Hands on 1

Spring Data JPA - Quick Example

CREATE SCHEMA ormlearn;

USE ormlearn;

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');

# 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 %-25.25logger**{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.MySQLDialect

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

*@Entity*

*@Table*(name = "country")

public class Country {

*@Id*

*@Column*(name = "code")

private String code;

*@Column*(name = "name")

private String name;

// Getters & 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; }

*@Override*

public String toString() {

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

}

}

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> {

}

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.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

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);

*countryService* = context.getBean(CountryService.class);

***LOGGER***.info("Inside main");

*testGetAllCountries*();

}

private static void testGetAllCountries() {

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

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

***LOGGER***.debug("countries={}", countries);

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

}

}

**Hands on 2**

**Hibernate XML Config implementation walk through**   
  
Hibernate uses mapping files (.hbm.xml) to map Java objects to database tables. This is called **Object-Relational Mapping (ORM)**.

**SessionFactory**

It is a heavyweight object created once per application.

Built using the hibernate.cfg.xml.

Responsible for creating Session objects.

**Session**

A lightweight, single-threaded object used to interact with the DB.

Represents a unit of work.

**Transaction**

Used to group multiple DB operations into one unit.

Starts with beginTransaction() and ends with commit() or rollback().

**beginTransaction()**

Starts a new transaction.

**commit()**

Saves all operations done in the session to the DB permanently.

**rollback()**

Used when an exception occurs to revert all changes.

**session.save(Object)**

Saves a new object (insert into table).

**session.get(Class, id)**

Fetches a record by primary key.

**session.createQuery("FROM ClassName").list()**

Runs an HQL (Hibernate Query Language) query and returns a list.

**session.delete(Object)**

Deletes a record.

**Hands on 3**

**Hibernate Annotation Config implementation walk through**   
  
Hibernate lets you **map Java classes to database tables using annotations**, instead of separate .hbm.xml mapping files.

**@Entity**

Declares this class as a Hibernate entity.

Maps it to a table in the DB.

**@Table(name = "EMPLOYEE")**

Specifies the table name this entity is mapped to.

**@Id**

Marks the primary key of the entity.

**@GeneratedValue**

Specifies how the primary key is auto-generated.

**@Column(name = "first\_name")**

Maps the Java field to a specific column in the DB.

Optional: can specify length, nullable, unique, etc.

| **XML Tag** | **Purpose** |
| --- | --- |
| hibernate.dialect | Tells Hibernate how to generate SQL for your DB (e.g., MySQL, PostgreSQL, etc.). |
| hibernate.connection.driver\_class | JDBC driver for your DB (e.g., com.mysql.cj.jdbc.Driver). |
| hibernate.connection.url | JDBC URL to connect to DB. Format: jdbc:mysql://host:port/database. |
| hibernate.connection.username | Username for DB login. |
| hibernate.connection.password | Password for DB login. |

**Hands on 4**

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

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (JSR 338) | ORM framework & JPA implementation | Abstraction layer over JPA (usually with Hibernate) |
| Implementation Provided? | ❌ No | ✅ Yes | ❌ No (relies on JPA providers like Hibernate) |
| Purpose | Define standard ORM behavior | Persist Java objects to relational DB | Simplify CRUD and boilerplate using Spring |
| Dependency on JPA Provider | N/A | Implements JPA | Uses Hibernate or others as backend |
| Boilerplate Code | Moderate | Moderate to High | Minimal – just extend interfaces |
| Example Config File | persistence.xml | hibernate.cfg.xml | application.properties / annotations |
| Transaction Management | Manual or via Spring | Manual or via Spring | Handled by Spring (@Transactional) |

**Hands on 5**

**Implement services for managing Country**

package com.cognizant.ormlearn.repository;

import java.util.List;

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> {

List<Country> findByNameContainingIgnoreCase(String partialName);

}

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.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

*@Transactional*

public Country findCountryByCode(String code) {

return countryRepository.findById(code).orElse(null);

}

*@Transactional*

public void addCountry(Country country) {

countryRepository.save(country);

}

*@Transactional*

public void updateCountry(Country country) {

countryRepository.save(country);

}

*@Transactional*

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

*@Transactional*

public List<Country> findCountriesByPartialName(String partialName) {

return countryRepository.findByNameContainingIgnoreCase(partialName);

}

}

package com.cognizant.ormlearn.service;

import java.util.List;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import com.cognizant.ormlearn.model.Country;

*@Component*

public class CountryServiceRunner implements CommandLineRunner {

*@Autowired*

private CountryService countryService;

*@Override*

public void run(String... args) throws Exception {

System.***out***.println("All countries: " + countryService.getAllCountries());

System.***out***.println("Find IN: " + countryService.findCountryByCode("IN"));

Country newCountry = new Country();

newCountry.setCode("XX");

newCountry.setName("Newland");

countryService.addCountry(newCountry);

newCountry.setName("Newlandia");

countryService.updateCountry(newCountry);

System.***out***.println("Partial search 'land': " + countryService.findCountriesByPartialName("land"));

countryService.deleteCountry("XX");

}

}

DROP TABLE IF EXISTS country;

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(50)

);

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);

*countryService* = context.getBean(CountryService.class);

***LOGGER***.info("Inside main");

*testGetAllCountries*();

*testFindCountryByCode*();

*testAddCountry*();

*testUpdateCountry*();

*testDeleteCountry*();

*testFindByPartialName*();

}

private static void testGetAllCountries() {

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

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

***LOGGER***.debug("Countries = {}", countries);

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

}

private static void testFindCountryByCode() {

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

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

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

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

}

private static void testAddCountry() {

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

Country country = new Country();

country.setCode("XX");

country.setName("Testland");

*countryService*.addCountry(country);

***LOGGER***.debug("Added Country = {}", *countryService*.findCountryByCode("XX"));

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

}

private static void testUpdateCountry() {

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

Country country = *countryService*.findCountryByCode("XX");

country.setName("UpdatedLand");

*countryService*.updateCountry(country);

***LOGGER***.debug("Updated Country = {}", *countryService*.findCountryByCode("XX"));

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

}

private static void testDeleteCountry() {

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

*countryService*.deleteCountry("XX");

Country country = *countryService*.findCountryByCode("XX");

***LOGGER***.debug("Country after delete = {}", country);

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

}

private static void testFindByPartialName() {

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

List<Country> countries = *countryService*.findCountriesByPartialName("land");

***LOGGER***.debug("Countries with 'land' = {}", countries);

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

}

}

**Hands on 6**

**Find a country based on country code**

package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

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.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

import java.util.Optional;

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

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

*@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();

}

*@Transactional*

public void addCountry(Country country) {

countryRepository.save(country);

}

*@Transactional*

public void updateCountry(Country country) {

countryRepository.save(country);

}

*@Transactional*

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

*@Transactional*

public List<Country> findCountriesByPartialName(String partialName) {

return countryRepository.findByNameContainingIgnoreCase(partialName);

}

}

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;

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

*@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);

***LOGGER***.info("Inside main");

*testGetAllCountries*();

*testFindCountryByCode*();

*testAddCountry*();

*testUpdateCountry*();

*testDeleteCountry*();

*testFindByPartialName*();

*getCountryByCodeTest*();

}

private static void testGetAllCountries() {

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

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

***LOGGER***.debug("Countries = {}", countries);

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

}

private static void testFindCountryByCode() {

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

try {

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

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

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testAddCountry() {

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

Country country = new Country();

country.setCode("XX");

country.setName("Testland");

*countryService*.addCountry(country);

try {

***LOGGER***.debug("Added Country = {}", *countryService*.findCountryByCode("XX"));

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testUpdateCountry() {

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

try {

Country country = *countryService*.findCountryByCode("XX");

country.setName("UpdatedLand");

*countryService*.updateCountry(country);

***LOGGER***.debug("Updated Country = {}", *countryService*.findCountryByCode("XX"));

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testDeleteCountry() {

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

*countryService*.deleteCountry("XX");

try {

Country country = *countryService*.findCountryByCode("XX");

***LOGGER***.debug("Country after delete = {}", country);

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testFindByPartialName() {

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

List<Country> countries = *countryService*.findCountriesByPartialName("land");

***LOGGER***.debug("Countries with 'land' = {}", countries);

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

}

private static void getCountryByCodeTest() {

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

try {

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

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

} catch (CountryNotFoundException e) {

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

}

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

}

}

**Hands on 7**

**Add a new country**

package com.cognizant.ormlearn.service;

import java.util.List;

import com.cognizant.ormlearn.model.Country;

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

public interface CountryService {

List<Country> getAllCountries();

Country findCountryByCode(String code) throws CountryNotFoundException;

void addCountry(Country country);

void updateCountry(Country country) throws CountryNotFoundException;

void deleteCountry(String code) throws CountryNotFoundException;

List<Country> findCountriesByPartialName(String partialName);

}

package com.cognizant.ormlearn.service;

import java.util.List;

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.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

*@Service*

public class CountryServiceImpl implements CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Override*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

*@Override*

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (result.isEmpty()) {

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

}

return result.get();

}

*@Override*

*@Transactional*

public void addCountry(Country country) {

countryRepository.save(country);

}

*@Override*

*@Transactional*

public void updateCountry(Country country) throws CountryNotFoundException {

Country existing = findCountryByCode(country.getCode());

existing.setName(country.getName());

countryRepository.save(existing);

}

*@Override*

*@Transactional*

public void deleteCountry(String code) throws CountryNotFoundException {

Country country = findCountryByCode(code);

countryRepository.delete(country);

}

*@Override*

public List<Country> findCountriesByPartialName(String partialName) {

return countryRepository.findByNameContainingIgnoreCase(partialName);

}

}

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;

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

*@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);

***LOGGER***.info("Inside main");

*testGetAllCountries*();

*testFindCountryByCode*();

*testAddCountry*();

*testUpdateCountry*();

*testDeleteCountry*();

*testFindByPartialName*();

*getCountryByCodeTest*();

}

private static void testGetAllCountries() {

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

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

***LOGGER***.debug("Countries = {}", countries);

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

}

private static void testFindCountryByCode() {

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

try {

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

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

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testAddCountry() {

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

Country country = new Country();

country.setCode("XX");

country.setName("Testland");

*countryService*.addCountry(country);

try {

Country added = *countryService*.findCountryByCode("XX");

***LOGGER***.debug("Added Country = {}", added);

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testUpdateCountry() {

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

try {

Country country = *countryService*.findCountryByCode("XX");

country.setName("UpdatedLand");

*countryService*.updateCountry(country);

***LOGGER***.debug("Updated Country = {}", *countryService*.findCountryByCode("XX"));

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testDeleteCountry() {

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

try {

*countryService*.deleteCountry("XX");

try {

Country country = *countryService*.findCountryByCode("XX");

***LOGGER***.debug("Country after delete = {}", country);

} catch (CountryNotFoundException e) {

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

}

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testFindByPartialName() {

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

List<Country> countries = *countryService*.findCountriesByPartialName("land");

***LOGGER***.debug("Countries with 'land' = {}", countries);

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

}

private static void getCountryByCodeTest() {

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

try {

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

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

} catch (CountryNotFoundException e) {

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

}

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

}

}

**Hands on 8**

**Update a country based on code**

package com.cognizant.ormlearn.service;

import java.util.List;

import com.cognizant.ormlearn.model.Country;

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

public interface CountryService {

List<Country> getAllCountries();

Country findCountryByCode(String code) throws CountryNotFoundException;

void addCountry(Country country);

void updateCountry(Country country) throws CountryNotFoundException;

void updateCountry(String code, String name) throws CountryNotFoundException;

void deleteCountry(String code) throws CountryNotFoundException;

List<Country> findCountriesByPartialName(String partialName);

}

package com.cognizant.ormlearn.service;

import java.util.List;

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.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

*@Service*

public class CountryServiceImpl implements CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Override*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

*@Override*

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (result.isEmpty()) {

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

}

return result.get();

}

*@Override*

*@Transactional*

public void addCountry(Country country) {

countryRepository.save(country);

}

*@Override*

*@Transactional*

public void updateCountry(Country country) throws CountryNotFoundException {

Country existing = findCountryByCode(country.getCode());

existing.setName(country.getName());

countryRepository.save(existing);

}

*@Override*

*@Transactional*

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

Country country = findCountryByCode(code);

country.setName(name);

countryRepository.save(country);

}

*@Override*

*@Transactional*

public void deleteCountry(String code) throws CountryNotFoundException {

Country country = findCountryByCode(code);

countryRepository.delete(country);

}

*@Override*

public List<Country> findCountriesByPartialName(String partialName) {

return countryRepository.findByNameContainingIgnoreCase(partialName);

}

}

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;

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

*@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);

***LOGGER***.info("Inside main");

*testGetAllCountries*();

*testFindCountryByCode*();

*testAddCountry*();

*testUpdateCountry*();

*testDeleteCountry*();

*testFindByPartialName*();

*getCountryByCodeTest*();

*testUpdateCountryName*();

}

private static void testGetAllCountries() {

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

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

***LOGGER***.debug("Countries = {}", countries);

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

}

private static void testFindCountryByCode() {

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

try {

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

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

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testAddCountry() {

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

Country country = new Country();

country.setCode("XX");

country.setName("Testland");

*countryService*.addCountry(country);

try {

Country added = *countryService*.findCountryByCode("XX");

***LOGGER***.debug("Added Country = {}", added);

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testUpdateCountry() {

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

try {

*countryService*.updateCountry("XX", "UpdatedLand");

Country updated = *countryService*.findCountryByCode("XX");

***LOGGER***.debug("Updated Country = {}", updated);

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testDeleteCountry() {

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

try {

*countryService*.deleteCountry("XX");

try {

Country country = *countryService*.findCountryByCode("XX");

***LOGGER***.debug("Country after delete = {}", country);

} catch (CountryNotFoundException e) {

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

}

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testFindByPartialName() {

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

List<Country> countries = *countryService*.findCountriesByPartialName("land");

***LOGGER***.debug("Countries with 'land' = {}", countries);

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

}

private static void getCountryByCodeTest() {

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

try {

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

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

} catch (CountryNotFoundException e) {

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

}

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

}

private static void testUpdateCountryName() {

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

try {

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

Country updated = *countryService*.findCountryByCode("IN");

***LOGGER***.debug("Updated Country = {}", updated);

} catch (CountryNotFoundException e) {

***LOGGER***.error("Country not found during update by code: {}", e.getMessage());

}

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

}

}

**Hands on 9**

**Delete a country based on code**

package com.cognizant.ormlearn.service;

import java.util.List;

import com.cognizant.ormlearn.model.Country;

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

public interface CountryService {

List<Country> getAllCountries();

Country findCountryByCode(String code) throws CountryNotFoundException;

void addCountry(Country country);

void updateCountry(Country country) throws CountryNotFoundException;

void updateCountry(String code, String name) throws CountryNotFoundException;

void deleteCountry(String code) throws CountryNotFoundException;

List<Country> findCountriesByPartialName(String partialName);

}

package com.cognizant.ormlearn.service;

import java.util.List;

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.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

*@Service*

public class CountryServiceImpl implements CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Override*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

*@Override*

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (result.isEmpty()) {

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

}

return result.get();

}

*@Override*

*@Transactional*

public void addCountry(Country country) {

countryRepository.save(country);

}

*@Override*

*@Transactional*

public void updateCountry(Country country) throws CountryNotFoundException {

Country existing = findCountryByCode(country.getCode());

existing.setName(country.getName());

countryRepository.save(existing);

}

*@Override*

*@Transactional*

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

Country country = findCountryByCode(code);

country.setName(name);

countryRepository.save(country);

}

*@Override*

*@Transactional*

public void deleteCountry(String code) throws CountryNotFoundException {

if (!countryRepository.existsById(code)) {

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

}

countryRepository.deleteById(code);

}

*@Override*

public List<Country> findCountriesByPartialName(String partialName) {

return countryRepository.findByNameContainingIgnoreCase(partialName);

}

}