**WEEK 3 – ASSIGNMENT**

**Spring Core and Maven**

**Exercise 1: Configuring a Basic Spring Application**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Core Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

</dependencies>

</project>

**applicationContext.xml**

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

<beans xmlns=<http://www.springframework.org/schema/beans>

Xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance>

Xsi:schemaLocation=”

<http://www.springframework.org/schema/beans>

<http://www.springframework.org/schema/beans/spring-beans.xsd>”>

<!—Repository Bean 🡪

<bean id=”bookRepository” class=”com.library.repository.BookRepository”/>

<!—Service Bean with dependency injection 🡪

<bean id=”bookService” class=”com.library.service.BookService”>

<property name=”bookRepository” ref=”bookRepository”/>

</bean>

</beans>

**Com.library.repository.BookRepository.java**

package com.library.repository;

public class BookRepository {

public void fetchBooks() {

System.out.println("Fetching books from the repository...");

}

}

**com.library.service.BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBooks() {

System.out.println("Inside BookService: Calling repository...");

bookRepository.fetchBooks();

}

}

**com.library.MainApp.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext(“applicationContext.xml”);

BookService bookService = (BookService) context.getBean(“bookService”);

bookService.displayBooks();

}

}

**Output:**

Inside BookService: Calling repository...

Fetching books from the repository...

**Exercise 2: Implementing Dependency Injection**

**applicationContext.xml**

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

<beans xmlns=<http://www.springframework.org/schema/beans>

Xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance>

Xsi:schemaLocation=”

<http://www.springframework.org/schema/beans>

<http://www.springframework.org/schema/beans/spring-beans.xsd>”>

<!—BookRepository Bean 🡪

<bean id=”bookRepository” class=”com.library.repository.BookRepository”/>

<!—BookService Bean with DI using setter method 🡪

<bean id=”bookService” class=”com.library.service.BookService”>

<property name=”bookRepository” ref=”bookRepository”/>

</bean>

</beans>

**com.library.service.BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

Public void setBookRepository(BookRepository bookRepository) {

This.bookRepository = bookRepository;

}

Public void displayBooks() {

System.out.println(“Inside BookService: Displaying books using repository...”);

bookRepository.fetchBooks();

}

}

**Com.library.repository.BookRepository.java**

package com.library.repository;

public class BookRepository {

public void fetchBooks() {

System.out.println("Fetching list of books from the database...");

}

}

**Com.library.LibraryManagementApplication.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.displayBooks();

}

}

**Output:**

Inside BookService: Displaying books using repository...

Fetching list of books from the database...

**Exercise 3: Creating and Configuring a Maven Project**

**Pom.xml**

<project xmlns=<http://maven.apache.org/POM/4.0.0>

Xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance>

Xsi:schemaLocation=”http://maven.apache.org/POM/4.0.0

<http://maven.apache.org/xsd/maven-4.0.0.xsd>”>

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!—Spring Context 🡪

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!—Spring AOP 🡪

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!—Spring Web MVC 🡪

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

<!—Servlet API (for Spring MVC to work) 🡪

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

**XML**

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Output:**

[INFO] BUILD SUCCESS

[INFO] Total time: 3.502 s

[INFO] Finished at: ...

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

**Exercise 1: Implement services for managing Country**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.CommandLineRunner;

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

import org.springframework.stereotype.Service;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.persistence.Entity;

import javax.persistence.Id;

import java.util.List;

import java.util.Optional;

// ---------- Boot application ----------

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

// Smoke‑tests for every feature

@Override

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

testAddCountry();

testGetCountry();

testUpdateCountry();

testFindByPartialName();

testDeleteCountry();

}

// ---- Add ----

private void testAddCountry() {

LOGGER.info("=== Add Country ===");

Country france = new Country("FR", "France");

countryService.addCountry(france);

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

}

//---- Read ----

private void testGetCountry() {

LOGGER.info("=== Get Country ===");

try {

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

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

} catch (CountryNotFoundException e) {

LOGGER.error(e.getMessage());

}

}

//---- Update ----

private void testUpdateCountry() {

LOGGER.info("=== Update Country ===");

try {

countryService.updateCountry("FR", "French Republic");

Country updated = countryService.findCountryByCode("FR");

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

} catch (CountryNotFoundException e) {

LOGGER.error(e.getMessage());

}

}

//---- Search by partial name ----

private void testFindByPartialName() {

LOGGER.info("=== Search Countries containing 're' ===");

List<Country> list = countryService.getCountriesByName("%re%");

list.forEach(c -> LOGGER.debug("Match: {}", c));

}

//---- Delete ----

private void testDeleteCountry() {

LOGGER.info("=== Delete Country ===");

countryService.deleteCountry("FR");

try {

countryService.findCountryByCode("FR");

} catch (CountryNotFoundException e) {

LOGGER.debug("Deletion confirmed: {}", e.getMessage());

}

}

}

@Entity

class Country {

@Id

private String code;

private String name;

public Country() {} // JPA needs it

public Country(String code, String name) { // convenience ctor

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; }

@Override public String toString() {

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

}

}

// ---------- Repository ----------

@Repository

interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String namePart);

}

// ---------- Service interface ----------

interface CountryService {

Country findCountryByCode(String code) throws CountryNotFoundException;

void addCountry(Country country);

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

void deleteCountry(String code);

List<Country> getCountriesByName(String likeExpression);

}

// ---------- Service implementation ----------

@Service

class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository repo;

//----- READ -----

@Override @Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

----- CREATE -----

@Override @Transactional

public void addCountry(Country country) {

repo.save(country);

}

// ----- UPDATE -----

@Override @Transactional

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

Country country = findCountryByCode(code);

country.setName(newName);

repo.save(country);

}

//---- DELETE -----

@Override @Transactional

public void deleteCountry(String code) {

repo.deleteById(code);

}

//----- SEARCH (partial name) -----

@Override @Transactional(readOnly = true)

public List<Country> getCountriesByName(String likeExpression) {

String clean = likeExpression.replace("%", "");

return repo.findByNameContainingIgnoreCase(clean);

}

}

//---------- Custom Exception ----------

class CountryNotFoundException extends Exception {

public CountryNotFoundException(String msg) { super(msg); }

}

**Output:**

INFO OrmLearnApplication - === Add Country ===

DEBUG OrmLearnApplication – Added: Country[code=FR, name=France]

INFO OrmLearnApplication - === Get Country ===

DEBUG OrmLearnApplication – Fetched: Country[code=FR, name=France]

INFO OrmLearnApplication - === Update Country ===

DEBUG OrmLearnApplication – Updated: Country[code=FR, name=French Republic]

INFO OrmLearnApplication - === Search Countries containing ‘re’ ===

DEBUG OrmLearnApplication – Match: Country[code=FR, name=French Republic]

DEBUG OrmLearnApplication – Match: Country[code=IE, name=Ireland]

... (other matches) ...

INFO OrmLearnApplication - === Delete Country ===

DEBUG OrmLearnApplication – Deletion confirmed: Country ‘FR’ not found

**Exercise 2: Find a country based on country code**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.CommandLineRunner;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

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

import org.springframework.stereotype.Repository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.persistence.Entity;

import javax.persistence.Id;

import java.util.Optional;

// Main application

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryService countryService;

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

}

@Override

public void run(String... args) {

getCountryByCodeTest();

}

private 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");

}

}

// Entity class

@Entity

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; }

@Override

public String toString() {

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

}

}

// Repository

@Repository

interface CountryRepository extends JpaRepository<Country, String> {

}

// Service Interface

interface CountryService {

Country findCountryByCode(String code) throws CountryNotFoundException;

}

// Service Implementation

@Service

class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository countryRepository;

@Override

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

}

// Custom Exception

class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**Output:**

INFO OrmLearnApplication – Start

DEBUG OrmLearnApplication – Country: Country [code=IN, name=India]

INFO OrmLearnApplication – End

**Exercise 3: Add a new country**

// File: OrmLearnApplication.java

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.CommandLineRunner;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

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

import org.springframework.stereotype.Repository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.persistence.Entity;

import javax.persistence.Id;

import java.util.Optional;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryService countryService;

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

}

@Override

public void run(String... args) {

testAddCountry();

}

// Add a new country

private void testAddCountry() {

LOGGER.info("Start testAddCountry");

Country newCountry = new Country();

newCountry.setCode("JP");

newCountry.setName("Japan");

countryService.addCountry(newCountry);

try {

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End testAddCountry");

}

}

// Entity

@Entity

class Country {

@Id

private String code;

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 + "]";

}

}

// Repository

@Repository

interface CountryRepository extends JpaRepository<Country, String> { }

// Service Interface

interface CountryService {

Country findCountryByCode(String code) throws CountryNotFoundException;

void addCountry(Country country);

}

// Service Implementation

@Service

class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository countryRepository;

@Override

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

@Override

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

}

// Custom Exception

class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**Output:**

INFO OrmLearnApplication – Start testAddCountry

DEBUG OrmLearnApplication – Country added: Country [code=JP, name=Japan]

INFO OrmLearnApplication – End testAddCountry

**Exercise 4: Demonstrate implementation of Query Methods feature of Spring Data JPA**

package com.cognizant.spring\_learn;

import org.springframework.boot.\*;

import org.springframework.boot.autoconfigure.\*;

import org.springframework.data.jpa.repository.\*;

import org.springframework.stereotype.\*;

import org.springframework.transaction.annotation.Transactional;

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

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDate;

import java.util.\*;

import org.slf4j.\*;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired private CountryRepository countryRepo;

@Autowired private StockRepository stockRepo;

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

}

@Override

public void run(String... args) {

LOGGER.info("Search countries containing 'ou':"); countryRepo.findByNameContainingIgnoreCase("ou")

.forEach(c -> LOGGER.info(c.toString()));

LOGGER.info("Search countries containing 'ou' sorted by name:"); countryRepo.findByNameContainingIgnoreCaseOrderByNameAsc("ou")

.forEach(c -> LOGGER.info(c.toString()));

LOGGER.info("Countries starting with 'Z':");

countryRepo.findByNameStartingWith("Z")

.forEach(c -> LOGGER.info(c.toString()));

LOGGER.info("Facebook stocks in Sept 2019:");

stockRepo.findByCodeAndDateBetween("FB",

LocalDate.of(2019, 9, 1), LocalDate.of(2019, 9, 30))

.forEach(s -> LOGGER.info(s.toString()));

LOGGER.info("Google stocks where close > 1250:"); stockRepo.findByCodeAndCloseGreaterThan("GOOGL", new BigDecimal("1250"))

.forEach(s -> LOGGER.info(s.toString()));

LOGGER.info("Top 3 stocks by volume:");

stockRepo.findTop3ByOrderByVolumeDesc()

.forEach(s -> LOGGER.info(s.toString()));

LOGGER.info("Lowest 3 Netflix closing prices:");

stockRepo.findTop3ByCodeOrderByCloseAsc("NFLX")

.forEach(s -> LOGGER.info(s.toString()));

}

// ---------- Entity: Country ----------

@Entity

class Country {

@Id

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code; this.name = name;

}

public String toString() {

return String.format("%-4s %s", code, name);

}

}

interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String namePart);

List<Country> findByNameContainingIgnoreCaseOrderByNameAsc(String namePart);

List<Country> findByNameStartingWith(String prefix);

}

@Entity

class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String code;

private LocalDate date;

private BigDecimal open;

private BigDecimal close;

private long volume;

public String toString() {

return String.format("%-6s %s open:%.2f close:%.2f vol:%d",

code, date, open, close, volume);

}

}

// ---------- Repository ------------

interface StockRepository extends JpaRepository<Stock, Integer> {

List<Stock> findByCodeAndDateBetween(String code, LocalDate start, LocalDate end);

List<Stock> findByCodeAndCloseGreaterThan(String code, BigDecimal price);

List<Stock> findTop3ByOrderByVolumeDesc();

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

**Output:**

Search countries containing ‘ou’:

BV Bouvet Island

DJ Djibouti

GP Guadeloupe

GS South Georgia and the South Sandwich Islands

LU Luxembourg

SS South Sudan

TF French Southern Territories

UM United States Minor Outlying Islands

ZA South Africa

Search countries containing ‘ou’ sorted by name:

BV Bouvet Island

DJ Djibouti

TF French Southern Territories

GP Guadeloupe

LU Luxembourg

ZA South Africa

GS South Georgia and the South Sandwich Islands

SS South Sudan

UM United States Minor Outlying Islands

Countries starting with ‘Z’:

ZM Zambia

ZW Zimbabwe

Facebook stocks in Sept 2019:

FB 2019-09-03 open:184.00 close:182.39 vol:9779400...

FB 2019-09-27 open:180.49 close:177.10 vol:14656200

Google stocks where close > 1250:

GOOGL 2019-04-22 open:1236.67 close:1253.76 vol:954200...

GOOGL 2019-10-17 open:1251.40 close:1252.80 vol:1047900

Top 3 stocks by volume:

FB 2019-01-31 open:165.60 close:166.69 vol:77233600

FB 2018-10-31 open:155.00 close:151.79 vol:60101300

FB 2018-12-19 open:141.21 close:133.24 vol:57404900

Lowest 3 Netflix closing prices:

NFLX 2018-12-24 open:242.00 close:233.88 vol:9547600

NFLX 2018-12-21 open:263.83 close:246.39 vol:21397600

NFLX 2018-12-26 open:233.92 close:253.67 vol:14402700

**Exercise 5: Demonstrate implementation of O/R Mapping**

package com.cognizant.orm\_demo;

import org.springframework.boot.\*;

import org.springframework.boot.autoconfigure.\*;

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

import org.springframework.stereotype.Repository;

import jakarta.persistence.\*;

import java.util.\*;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

private final EmployeeRepo empRepo;

private final DepartmentRepo deptRepo;

private final SkillRepo skillRepo;

public OrmLearnApplication(EmployeeRepo e, DepartmentRepo d, SkillRepo s) {

this.empRepo = e; this.deptRepo = d; this.skillRepo = s;

}

public static void main(String[] a){SpringApplication.run(OrmLearnApplication.class,a);}

@Override public void run(String... a){

Department it = deptRepo.save(new Department("IT"));

Skill java = skillRepo.save(new Skill("Java"));

Skill sql = skillRepo.save(new Skill("SQL"));

Employee emp = new Employee("Alice", 90000);

emp.setDepartment(it);

// @ManyToOne side

emp.getSkillList().addAll(List.of(java, sql)); // @ManyToMany

empRepo.save(emp);

/\* ---------- fetch ---------- \*/

Employee fetched = empRepo.findById(emp.getId()).get();

System.out.println("EMP : " + fetched.getName());

System.out.println("DEPT : " + fetched.getDepartment().getName()); // eager by default

System.out.println("SKILL: " + fetched.getSkillList().stream().map(Skill::getName).toList()); // LAZY by default

}

}

@Entity

class Department {

@Id @GeneratedValue private int id;

private String name;

@OneToMany(mappedBy = "department", fetch = FetchType.LAZY)

private Set<Employee> employeeList = new HashSet<>();

Department() {}

Department(String name){this.name=name;}

public String getName(){return name;}

}

@Entity

class Employee {

@Id @GeneratedValue private int id;

private String name; private double salary;

/\* many‑to‑one \*/

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "dept\_id") // FK column

private Department department;

/\* many‑to‑many \*/

@ManyToMany(fetch = FetchType.LAZY)

@JoinTable(name = "employee\_skill",

joinColumns = @JoinColumn(name = "emp\_id"),

inverseJoinColumns = @JoinColumn(name = "sk\_id"))

private Set<Skill> skillList = new HashSet<>();

Employee(){}

Employee(String n,double s){name=n;salary=s;}

public int getId(){return id;}

public String getName(){return name;}

public Department getDepartment(){return department;}

public void setDepartment(Department d){this.department=d;}

public Set<Skill> getSkillList(){return skillList;}

}

@Entity

class Skill {

@Id @GeneratedValue private int id;

private String name;

@ManyToMany(mappedBy = "skillList", fetch = FetchType.LAZY)

private Set<Employee> employeeList = new HashSet<>();

Skill(){}

Skill(String n){name=n;}

public String getName(){return name;}

}

@Repository interface EmployeeRepo extends JpaRepository<Employee,Integer>{}

@Repository interface DepartmentRepo extends JpaRepository<Department,Integer>{}

@Repository interface SkillRepo extends JpaRepository<Skill,Integer>{}

**Output:**

EMP : Alice

DEPT : IT

SKILL: [Java, SQL]

**Exercise 6: Demonstrate writing Hibernate Query Language and Native Query**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.CommandLineRunner;

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

import org.springframework.stereotype.Service;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

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

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

import org.springframework.data.repository.query.Param;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.OneToMany;

import jakarta.persistence.FetchType;

import jakarta.persistence.CascadeType;

import java.util.Optional;

import java.util.List;

import java.util.ArrayList;

// ----------- Spring Boot Application -----------

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryService countryService;

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

}

@Override

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

Country c1 = new Country("US", "United States");

Country c2 = new Country("IN", "India");

Country c3 = new Country("FR", "France");

countryService.addCountry(c1);

countryService.addCountry(c2);

countryService.addCountry(c3);

// 1. HQL Query

Country india = countryService.findByCountryName("India");

LOGGER.info("HQL Result: {}", india);

// 2. Native Query

Country france = countryService.findByCodeNative("FR");

LOGGER.info("Native SQL Result: {}", france);

// 3. Partial Name Query

List<Country> matches = countryService.findByPartialName("in");

LOGGER.info("Countries containing 'in':");

matches.forEach(c -> LOGGER.info("Match: {}", c));

long count = countryService.countCountries();

LOGGER.info("Total countries in DB: {}", count);

}

}

// ----------- Entity -----------

@Entity

class Country {

@Id

private String code;

private String name;

public Country() {}

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; }

@Override

public String toString() {

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

}

}

// ----------- Repository -----------

@Repository

interface CountryRepository extends JpaRepository<Country, String> {

@Query("SELECT c FROM Country c WHERE c.name = :name")

Country findByCountryName(@Param("name") String name);

@Query(value = "SELECT \* FROM country WHERE code = :code", nativeQuery = true)

Country findByCodeNative(@Param("code") String code);

@Query("SELECT c FROM Country c WHERE LOWER(c.name) LIKE LOWER(CONCAT('%', :keyword, '%'))")

List<Country> findByPartialName(@Param("keyword") String keyword);

@Query("SELECT COUNT(c) FROM Country c")

long countCountries();

}

// ----------- Service Interface -----------

interface CountryService {

void addCountry(Country country);

Country findByCountryName(String name);

Country findByCodeNative(String code);

List<Country> findByPartialName(String keyword);

long countCountries();

}

// ----------- Service Implementation -----------

@Service

class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository repo;

@Override @Transactional

public void addCountry(Country country) {

repo.save(country);

}

@Override @Transactional(readOnly = true)

public Country findByCountryName(String name) {

return repo.findByCountryName(name);

}

@Override @Transactional(readOnly = true)

public Country findByCodeNative(String code) {

return repo.findByCodeNative(code);

}

@Override @Transactional(readOnly = true)

public List<Country> findByPartialName(String keyword) {

return repo.findByPartialName(keyword);

}

@Override @Transactional(readOnly = true)

public long countCountries() {

return repo.countCountries();

}

}

// ----------- Custom Exception -----------

class CountryNotFoundException extends Exception {

public CountryNotFoundException(String msg) {

super(msg);

}

}

**Output:**

HQL Result: Country[code=IN, name=India]

Native SQL Result: Country[code=FR, name=France]

Countries containing 'in':

Match: Country[code=IN, name=India]

Match: Country[code=FR, name=France]

Total countries in DB: 3