**Exercise 1 – Country Table (Query Methods)**

**Country.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

@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; }

@Override

public String toString() {

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

}

}

**CountryRepository.java**

package com.example.ormlearn.repository;

import java.util.List;

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

import com.example.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String keyword);

List<Country> findByNameContainingOrderByNameAsc(String keyword);

List<Country> findByNameStartingWith(String letter);

}

**Exercise 2 – Stock Table (Query Methods)**

**Stock.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.sql.Date;

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "st\_code")

private String code;

@Column(name = "st\_date")

private Date date;

@Column(name = "st\_open")

private BigDecimal open;

@Column(name = "st\_close")

private BigDecimal close;

@Column(name = "st\_volume")

private BigDecimal volume;

// Getters, Setters, toString

public String toString() {

return code + " | " + date + " | " + open + " | " + close + " | " + volume;

}

}

**StockRepository.java**

package com.example.ormlearn.repository;

import java.math.BigDecimal;

import java.sql.Date;

import java.util.List;

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

import com.example.ormlearn.model.Stock;

public interface StockRepository extends JpaRepository<Stock, Integer> {

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

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

List<Stock> findTop3ByOrderByVolumeDesc();

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

**Exercise 4 – ManyToOne (Employee ↔ Department)**

**Employee.java**

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private double salary;

private boolean permanent;

@Column(name = "em\_date\_of\_birth")

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "em\_dp\_id")

private Department department;

// Getters, Setters, toString

}

**Department.java**

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

// Getters, Setters, toString

}

**Repositories**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

public interface DepartmentRepository extends JpaRepository<Department, Integer> {}

**Service Layer**

@Service

public class EmployeeService {

@Autowired private EmployeeRepository repo;

public Employee get(int id) { return repo.findById(id).get(); }

public void save(Employee e) { repo.save(e); }

}

**Exercise 5 – OneToMany (Department → Employee)**

**Department.java**

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

private Set<Employee> employeeList;

**Exercise 6 – ManyToMany (Employee ↔ Skill)**

**Skill.java**

@Entity

@Table(name = "skill")

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

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

private Set<Employee> employeeList;

// Getters, Setters, toString

}

**Employee.java**

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name = "employee\_skill",

joinColumns = @JoinColumn(name = "es\_em\_id"),

inverseJoinColumns = @JoinColumn(name = "es\_sk\_id"))

private Set<Skill> skillList;

**OrmLearnApplication.java**

package com.example.ormlearn;

import java.math.BigDecimal;

import java.sql.Date;

import java.util.List;

import java.util.Set;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.model.Department;

import com.example.ormlearn.model.Employee;

import com.example.ormlearn.model.Skill;

import com.example.ormlearn.model.Stock;

import com.example.ormlearn.repository.CountryRepository;

import com.example.ormlearn.repository.StockRepository;

import com.example.ormlearn.service.DepartmentService;

import com.example.ormlearn.service.EmployeeService;

import com.example.ormlearn.service.SkillService;

@SpringBootApplication

public class OrmLearnApplication {

@Autowired

private CountryRepository countryRepository;

@Autowired

private StockRepository stockRepository;

@Autowired

private EmployeeService employeeService;

@Autowired

private DepartmentService departmentService;

@Autowired

private SkillService skillService;

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

public static void main(String[] args) {

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

OrmLearnApplication app = context.getBean(OrmLearnApplication.class);

}

private void testSearchByContainingText() {

LOGGER.info("Search by containing 'ou':");

List<Country> countries = countryRepository.findByNameContaining("ou");

countries.forEach(country -> LOGGER.debug("Country: {}", country));

}

private void testSearchByContainingSorted() {

LOGGER.info("Search by containing 'ou' sorted:");

List<Country> countries = countryRepository.findByNameContainingOrderByNameAsc("ou");

countries.forEach(country -> LOGGER.debug("Country: {}", country));

}

private void testSearchByStartingLetter() {

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

List<Country> countries = countryRepository.findByNameStartingWith("Z");

countries.forEach(country -> LOGGER.debug("Country: {}", country));

}

private void testFacebookStockInSep2019() {

List<Stock> stocks = stockRepository.findByCodeAndDateBetween("FB", Date.valueOf("2019-09-01"), Date.valueOf("2019-09-30"));

stocks.forEach(stock -> LOGGER.debug("Stock: {}", stock));

}

private void testGoogleStockPriceAbove1250() {

List<Stock> stocks = stockRepository.findByCodeAndCloseGreaterThan("GOOGL", new BigDecimal("1250"));

stocks.forEach(stock -> LOGGER.debug("Stock: {}", stock));

}

private void testTop3HighestVolume() {

List<Stock> stocks = stockRepository.findTop3ByOrderByVolumeDesc();

stocks.forEach(stock -> LOGGER.debug("Stock: {}", stock));

}

private void testTop3LowestNetflix() {

List<Stock> stocks = stockRepository.findTop3ByCodeOrderByCloseAsc("NFLX");

stocks.forEach(stock -> LOGGER.debug("Stock: {}", stock));

}

private void testGetEmployeeWithDepartment() {

Employee emp = employeeService.get(1);

LOGGER.debug("Employee: {}", emp);

LOGGER.debug("Department: {}", emp.getDepartment());

LOGGER.debug("Skills: {}", emp.getSkillList());

}

private void testAddEmployee() {

Employee employee = new Employee();

employee.setName("John Doe");

employee.setSalary(50000);

employee.setPermanent(true);

employee.setDateOfBirth(Date.valueOf("1990-01-01"));

Department dept = departmentService.get(1);

employee.setDepartment(dept);

employeeService.save(employee);

LOGGER.debug("Saved Employee: {}", employee);

}

private void testUpdateEmployee() {

Employee employee = employeeService.get(1);

Department newDept = departmentService.get(2);

employee.setDepartment(newDept);

employeeService.save(employee);

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

}

private void testDepartmentWithEmployees() {

Department dept = departmentService.get(1);

LOGGER.debug("Department: {}", dept);

dept.getEmployeeList().forEach(emp -> LOGGER.debug("Employee: {}", emp));

}

private void testAddSkillToEmployee() {

Employee emp = employeeService.get(1);

Skill skill = skillService.get(1);

emp.getSkillList().add(skill);

employeeService.save(emp);

LOGGER.debug("Added Skill to Employee: {}", emp);

}

}