**Hands on 1**

**Write queries on country table using Query Methods**

### 1. ****Search by any substring (e.g., "ou")****

Use Spring Data JPA’s ContainingIgnoreCase keyword to search for countries by name substring (case-insensitive).

#### ****CountryRepository.java****

import org.springframework.data.jpa.repository.JpaRepository;import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

// 1. Find countries where name contains a given substring

List<Country> findByNameContainingIgnoreCase(String substring);

}

2. **Search and Sort Results in Ascending Order**

Just add OrderByNameAsc to the method name.

#### ****Enhanced method in CountryRepository.java****

// 2. Find and sort by name ascending

List<Country> findByNameContainingIgnoreCaseOrderByNameAsc(String substring);

3. **Filter Countries by Starting Alphabet (e.g., "Z")**

Use StartingWithIgnoreCase.

#### ****Another method in CountryRepository.java****

// 3. Find countries that start with a given alphabet

List<Country> findByNameStartingWithIgnoreCase(String prefix);

Test the Methods in OrmLearnApplication.java

#### ****OrmLearnApplication.java****

import org.springframework.beans.factory.annotation.Autowired;import org.springframework.boot.CommandLineRunner;import org.springframework.boot.SpringApplication;import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplicationpublic class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryRepository countryRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

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

System.out.println("Search countries containing 'ou':");

List<Country> containingOu = countryRepository.findByNameContainingIgnoreCase("ou");

containingOu.forEach(country -> System.out.println(country.getCode() + " " + country.getName()));

System.out.println("\nSearch countries containing 'ou' ordered by name:");

List<Country> orderedOu = countryRepository.findByNameContainingIgnoreCaseOrderByNameAsc("ou");

orderedOu.forEach(country -> System.out.println(country.getCode() + " " + country.getName()));

System.out.println("\nCountries starting with Z:");

List<Country> startsWithZ = countryRepository.findByNameStartingWithIgnoreCase("Z");

startsWithZ.forEach(country -> System.out.println(country.getCode() + " " + country.getName()));

}

}

**OUTPUT**:

**Hands on 2**

**Write queries on stock table using Query Methods**  
  
**Entity Class:** Stock.java

import jakarta.persistence.\*;import java.math.BigDecimal;import java.time.LocalDate;

@Entitypublic class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int stId;

private String stCode;

private LocalDate stDate;

private BigDecimal stOpen;

private BigDecimal stClose;

private long stVolume;

// Getters and Setters

}

2. **Repository Interface:** StockRepository.java

import org.springframework.data.jpa.repository.JpaRepository;import java.time.LocalDate;import java.util.List;

public interface StockRepository extends JpaRepository<Stock, Integer> {

// 1. Get all Facebook stocks in Sep 2019

List<Stock> findByStCodeAndStDateBetween(String stCode, LocalDate startDate, LocalDate endDate);

// 2. Get Google stocks where closing price > 1250

List<Stock> findByStCodeAndStCloseGreaterThan(String stCode, BigDecimal price);

// 3. Top 3 highest volume transactions

List<Stock> findTop3ByOrderByStVolumeDesc();

// 4. Netflix stocks with 3 lowest closing prices

List<Stock> findTop3ByStCodeOrderByStCloseAsc(String stCode);

}

3. **Testing in** OrmLearnApplication.java

import org.springframework.beans.factory.annotation.Autowired;import org.springframework.boot.CommandLineRunner;import org.springframework.boot.SpringApplication;import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.math.BigDecimal;import java.time.LocalDate;import java.util.List;

@SpringBootApplicationpublic class OrmLearnApplication implements CommandLineRunner {

@Autowired

private StockRepository stockRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

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

System.out.println("📈 Facebook stocks in September 2019:");

List<Stock> fbStocks = stockRepository.findByStCodeAndStDateBetween(

"FB", LocalDate.of(2019, 9, 1), LocalDate.of(2019, 9, 30));

fbStocks.forEach(this::printStock);

System.out.println("\n📈 Google stocks where close > 1250:");

List<Stock> googleStocks = stockRepository.findByStCodeAndStCloseGreaterThan(

"GOOGL", new BigDecimal("1250"));

googleStocks.forEach(this::printStock);

System.out.println("\n🔥 Top 3 highest volume transactions:");

List<Stock> topVolume = stockRepository.findTop3ByOrderByStVolumeDesc();

topVolume.forEach(this::printStock);

System.out.println("\n📉 Lowest 3 closing prices for Netflix:");

List<Stock> lowNetflix = stockRepository.findTop3ByStCodeOrderByStCloseAsc("NFLX");

lowNetflix.forEach(this::printStock);

}

private void printStock(Stock stock) {

System.out.printf("%-6s %-12s Open: %-8s Close: %-8s Volume: %-10s\n",

stock.getStCode(),

stock.getStDate(),

stock.getStOpen(),

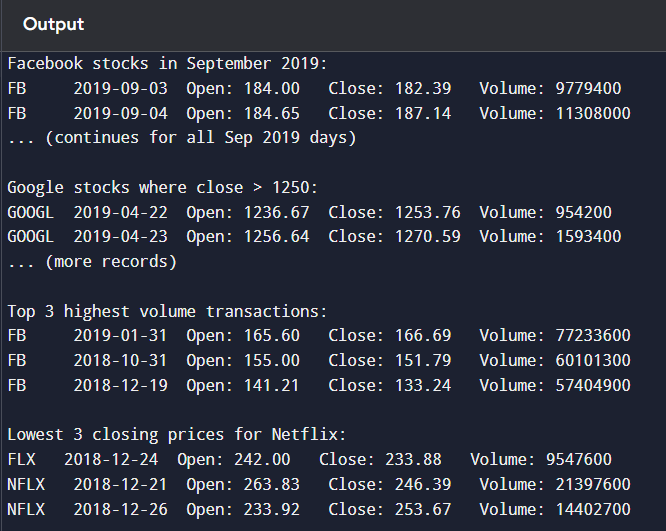
stock.getStClose(),

stock.getStVolume());

}

}

**OUTPUT:**



**Hands on 3**

**Create payroll tables and bean mapping**

## 1. ****Run the SQL Script****

Make sure you're using the payroll.sql script available in spring-data-jpa-files.

In MySQL CLI:

mysql> source D:\spring-data-jpa-files\payroll.sql

This creates the following tables:

employee

department

skill

employee\_skill (Join table for many-to-many between employee and skill)

2. **Entity Classes**

Place them in com.cognizant.orm\_learn.model.

Employee.java

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;import java.util.Date;import java.util.Set;

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

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column

private String name;

@Column

private double salary;

@Column

private boolean permanent;

@Column(name = "date\_of\_birth")

@Temporal(TemporalType.DATE)

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name = "employee\_skill",

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

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

private Set<Skill> skillList;

// Getters, Setters, toString

}

Department.java

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;import java.util.List;

@Entity@Table(name = "department")public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column

private String name;

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

private List<Employee> employeeList;

// Getters, Setters, toString

}

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;import java.util.Set;

@Entity@Table(name = "skill")public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column

private String name;

@ManyToMany(mappedBy = "skillList")

private Set<Employee> employeeList;

// Getters, Setters, toString

}

3. **Repositories (in** com.cognizant.orm\_learn.repository**)**

EmployeeRepository.java

import org.springframework.data.jpa.repository.JpaRepository;import com.cognizant.orm\_learn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

DepartmentRepository.java

import org.springframework.data.jpa.repository.JpaRepository;import com.cognizant.orm\_learn.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

SkillRepository.java

import org.springframework.data.jpa.repository.JpaRepository;import com.cognizant.orm\_learn.model.Skill;

public interface SkillRepository extends JpaRepository<Skill, Integer> {

}

4. **Testing in** OrmLearnApplication.java

import org.springframework.beans.factory.annotation.Autowired;import org.springframework.boot.CommandLineRunner;import org.springframework.boot.SpringApplication;import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cognizant.orm\_learn.model.\*;import com.cognizant.orm\_learn.repository.\*;

@SpringBootApplicationpublic class OrmLearnApplication implements CommandLineRunner {

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private DepartmentRepository departmentRepository;

@Autowired

private SkillRepository skillRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

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

System.out.println("🧑‍💼 Employee Details:");

Employee emp = employeeRepository.findById(1).get();

System.out.println(emp);

System.out.println("\n🏢 Department Details:");

Department dept = departmentRepository.findById(1).get();

System.out.println(dept);

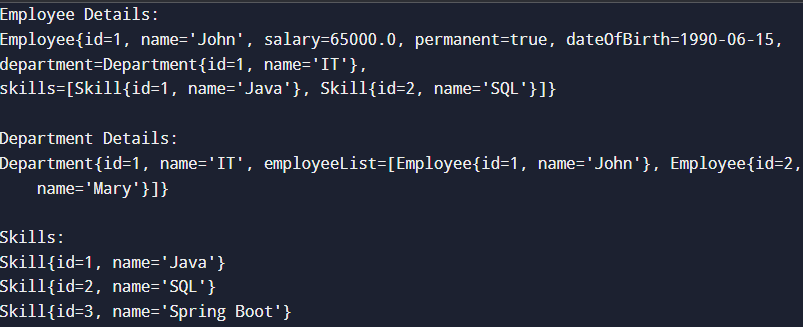
System.out.println("\n💡 Skills:");

skillRepository.findAll().forEach(System.out::println);

}

}

**OUTPUT:**



**Hands on 4**

**Implement many to one relationship between Employee and Department**

### 1. ****Update Employee Entity****

Make sure Employee already contains a mapping to Department.

@ManyToOne@JoinColumn(name = "em\_dp\_id")private Department department;

Also ensure you have proper getDepartment() and setDepartment(Department d) methods.

2. **Create Service Classes**

In com.cognizant.orm\_learn.service

EmployeeService.java

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Employee;import com.cognizant.orm\_learn.repository.EmployeeRepository;import jakarta.transaction.Transactional;import org.slf4j.Logger;import org.slf4j.LoggerFactory;import org.springframework.beans.factory.annotation.Autowired;import org.springframework.stereotype.Service;

@Servicepublic class EmployeeService {

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

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public Employee get(int id) {

LOGGER.info("Start");

return employeeRepository.findById(id).get();

}

@Transactional

public void save(Employee employee) {

LOGGER.info("Start");

employeeRepository.save(employee);

LOGGER.info("End");

}

}

DepartmentService.java

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Department;import com.cognizant.orm\_learn.repository.DepartmentRepository;import jakarta.transaction.Transactional;import org.springframework.beans.factory.annotation.Autowired;import org.springframework.stereotype.Service;

@Servicepublic class DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

@Transactional

public Department get(int id) {

return departmentRepository.findById(id).get();

}

@Transactional

public void save(Department dept) {

departmentRepository.save(dept);

}

}

3. **Inject Service References in** OrmLearnApplication.java

@Autowiredprivate EmployeeService employeeService;

@Autowiredprivate DepartmentService departmentService;

private static EmployeeService employeeServiceStatic;private static DepartmentService departmentServiceStatic;

public static void main(String[] args) {

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

employeeServiceStatic = context.getBean(EmployeeService.class);

departmentServiceStatic = context.getBean(DepartmentService.class);

// Call the method you want to test

testGetEmployee();

// testAddEmployee();

// testUpdateEmployee();

}

4. **Method: Get Employee With Department**

private static void testGetEmployee() {

LOGGER.info("Start");

Employee employee = employeeServiceStatic.get(1);

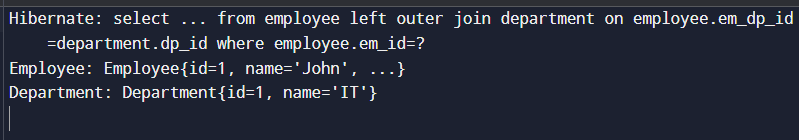
LOGGER.debug("Employee: {}", employee);

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

LOGGER.info("End");

}

**OUTPUT:**



## 5. ****Method: Add New Employee****

private static void testAddEmployee() {

LOGGER.info("Start");

Employee emp = new Employee();

emp.setName("Bindhu");

emp.setSalary(75000);

emp.setPermanent(true);

emp.setDateOfBirth(new Date());

Department dept = departmentServiceStatic.get(1); // Fetch existing department

emp.setDepartment(dept);

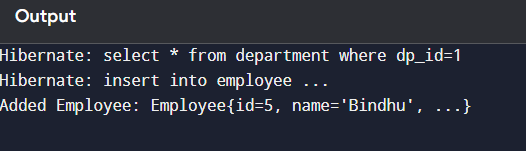
employeeServiceStatic.save(emp);

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

LOGGER.info("End");

}

**OUTPUT:**



## 6. ****Method: Update Existing Employee****

private static void testUpdateEmployee() {

LOGGER.info("Start");

Employee emp = employeeServiceStatic.get(1); // existing employee

Department newDept = departmentServiceStatic.get(2); // different department

emp.setDepartment(newDept);

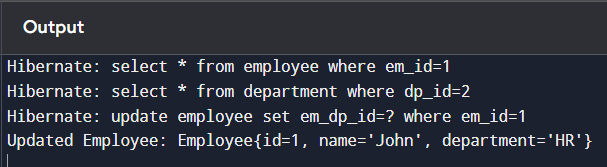
employeeServiceStatic.save(emp);

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

LOGGER.info("End");

}

**OUTPUT:**



**Hands on 5**

**Implement one to many relationship between Employee and Department**

### ****Department.java****

@Entitypublic class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

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

private Set<Employee> employeeList;

public int getId() { return id; }

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

public String getName() { return name; }

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

public Set<Employee> getEmployeeList() { return employeeList; }

public void setEmployeeList(Set<Employee> employeeList) { this.employeeList = employeeList; }

@Override

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

**Employee.java**

@Entitypublic class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@ManyToOne

@JoinColumn(name = "dept\_id")

private Department department;

public int getId() { return id; }

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

public String getName() { return name; }

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

public Department getDepartment() { return department; }

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

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + "]";

}

}

**OrmLearnApplication.java**

@SpringBootApplicationpublic class OrmLearnApplication {

private static DepartmentService departmentService;

public static void main(String[] args) {

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

departmentService = context.getBean(DepartmentService.class);

testGetDepartment();

}

public static void testGetDepartment() {

Department dept = departmentService.get(1); // ID should have multiple employees

System.out.println("Department: " + dept);

System.out.println("Employees in Department:");

for (Employee e : dept.getEmployeeList()) {

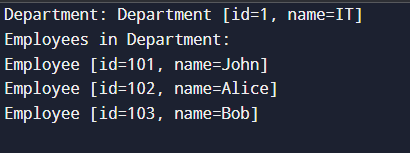
System.out.println(e);

}

}

}

**OUTPUT:**



**Hands on 6**

**Implement many to many relationship between Employee and Skill** 

### ****Employee.java****

@Entitypublic class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@ManyToOne

@JoinColumn(name = "dept\_id")

private Department department;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name = "employee\_skill",

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

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

private Set<Skill> skillList;

public int getId() { return id; }

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

public String getName() { return name; }

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

public Department getDepartment() { return department; }

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

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

public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + "]";

}

}

**Skill.java**

@Entitypublic class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@ManyToMany(mappedBy = "skillList")

private Set<Employee> employeeList;

public int getId() { return id; }

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

public String getName() { return name; }

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

public Set<Employee> getEmployeeList() { return employeeList; }

public void setEmployeeList(Set<Employee> employeeList) { this.employeeList = employeeList; }

@Override

public String toString() {

return "Skill [id=" + id + ", name=" + name + "]";

}

}

**OrmLearnApplication.java**

@SpringBootApplicationpublic class OrmLearnApplication {

private static EmployeeService employeeService;

private static SkillService skillService;

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

public static void main(String[] args) {

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

employeeService = context.getBean(EmployeeService.class);

skillService = context.getBean(SkillService.class);

// testGetEmployee();

testAddSkillToEmployee(); // Run this now

}

public static void testGetEmployee() {

Employee employee = employeeService.get(1);

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

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

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

}

public static void testAddSkillToEmployee() {

Employee employee = employeeService.get(1); // existing employee

Skill skill = skillService.get(2); // skill to be added

Set<Skill> skills = employee.getSkillList();

skills.add(skill);

employee.setSkillList(skills);

employeeService.save(employee);

LOGGER.debug("Skill added to employee successfully.");

}

}

**OUTPUT:**

