**Hands on 1**

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

**Country.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

*@Entity*

public class Country {

*@Id*

private String code;

private String name;

// Required no-argument constructor

public Country() {

}

// Constructor with fields (IMPORTANT!)

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.cognizant.ormlearn.repository;

import java.util.List;

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

import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

// 1. Search by name containing a substring

List<Country> findByNameContaining(String text);

// 2. Search by name containing and return sorted

List<Country> findByNameContainingOrderByNameAsc(String text);

// 3. Filter by name starting with a character

List<Country> findByNameStartingWith(String prefix);

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import java.util.List;

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.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

*@SpringBootApplication*

public 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 {

// Seed Data

countryRepository.save(new Country("ZA", "South Africa"));

countryRepository.save(new Country("DJ", "Djibouti"));

countryRepository.save(new Country("GP", "Guadeloupe"));

countryRepository.save(new Country("LU", "Luxembourg"));

countryRepository.save(new Country("BV", "Bouvet Island"));

countryRepository.save(new Country("GS", "South Georgia and the South Sandwich Islands"));

countryRepository.save(new Country("SS", "South Sudan"));

countryRepository.save(new Country("TF", "French Southern Territories"));

countryRepository.save(new Country("UM", "United States Minor Outlying Islands"));

countryRepository.save(new Country("ZM", "Zambia"));

countryRepository.save(new Country("ZW", "Zimbabwe"));

System.***out***.println("Countries containing 'ou':");

List<Country> result1 = countryRepository.findByNameContaining("ou");

result1.forEach(c -> System.***out***.println(c.getCode() + " " + c.getName()));

System.***out***.println("\nCountries containing 'ou' sorted by name:");

List<Country> result2 = countryRepository.findByNameContainingOrderByNameAsc("ou");

result2.forEach(c -> System.***out***.println(c.getCode() + " " + c.getName()));

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

List<Country> result3 = countryRepository.findByNameStartingWith("Z");

result3.forEach(c -> System.***out***.println(c.getCode() + " " + c.getName()));

}

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=create

**pom.xml**

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

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>orm-learn</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-milestones</id>

<name>Spring Milestone Repository</name>

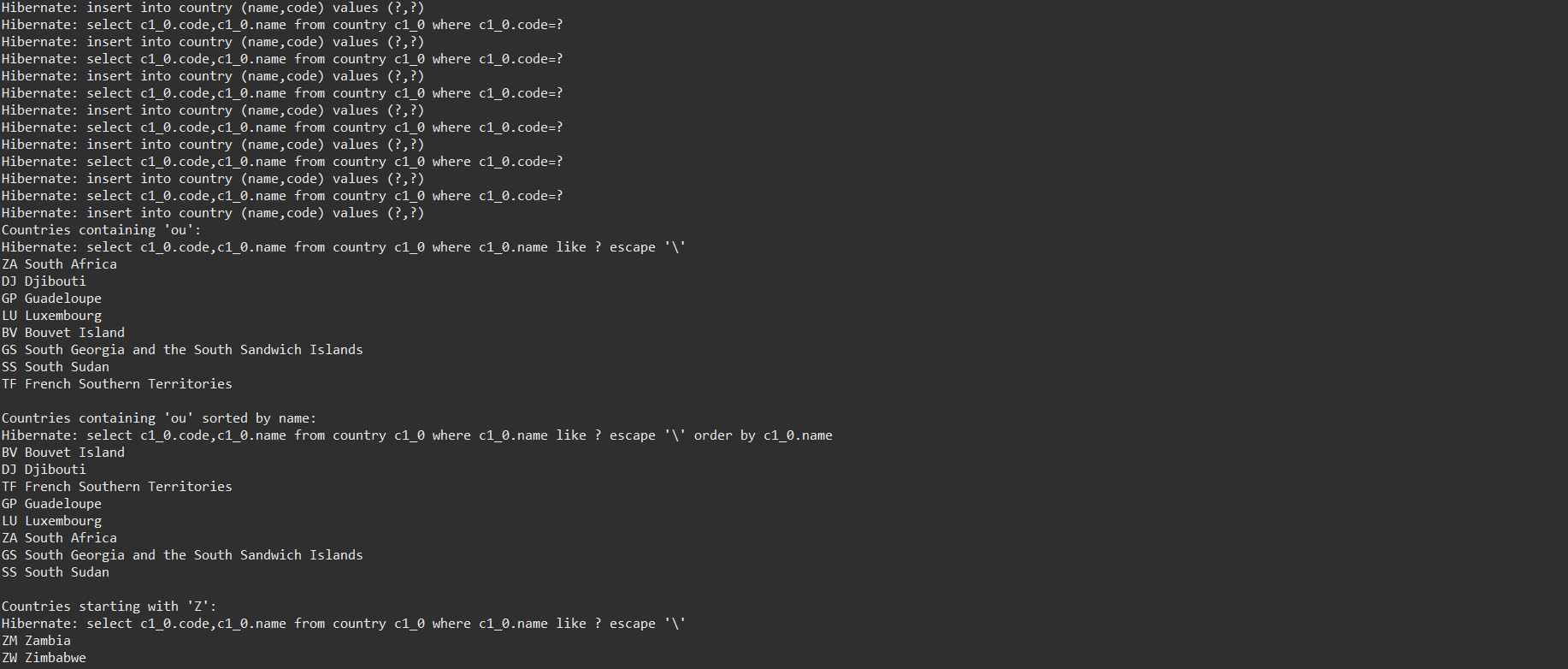
<url>https://repo.spring.io/milestone</url>

</repository>

</repositories>

</project>

**Output:**

****

**Hands on 2**

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

**Stock.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDate;

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "st\_id")

private Integer id;

@Column(name = "st\_code")

private String code;

@Column(name = "st\_date")

private LocalDate date;

@Column(name = "st\_open")

private BigDecimal open;

@Column(name = "st\_close")

private BigDecimal close;

@Column(name = "st\_volume")

private Long volume;

// Getters and Setters

public Integer getId() { return id; }

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

public String getCode() { return code; }

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

public LocalDate getDate() { return date; }

public void setDate(LocalDate date) { this.date = date; }

public BigDecimal getOpen() { return open; }

public void setOpen(BigDecimal open) { this.open = open; }

public BigDecimal getClose() { return close; }

public void setClose(BigDecimal close) { this.close = close; }

public Long getVolume() { return volume; }

public void setVolume(Long volume) { this.volume = volume; }

}

**StockRepository.java**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Stock;

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

import org.springframework.stereotype.Repository;

import java.math.BigDecimal;

import java.time.LocalDate;

import java.util.List;

*@Repository*

public interface StockRepository extends JpaRepository<Stock, Integer> {

// 1. Facebook stock in September 2019

List<Stock> findByCodeAndDateBetween(String code, LocalDate startDate, LocalDate endDate);

// 2. Google stocks where close > 1250

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

// 3. Top 3 highest volume transactions

List<Stock> findTop3ByOrderByVolumeDesc();

// 4. 3 dates with lowest Netflix stock close

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Stock;

import com.cognizant.ormlearn.repository.StockRepository;

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;

*@SpringBootApplication*

public 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) {

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

List<Stock> fbStocks = stockRepository.findByCodeAndDateBetween("FB",

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

fbStocks.forEach(System.***out***::println);

System.***out***.println("\nGoogle stocks with close > 1250:");

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

googlStocks.forEach(System.***out***::println);

System.***out***.println("\nTop 3 highest volume stocks:");

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

topVolumes.forEach(System.***out***::println);

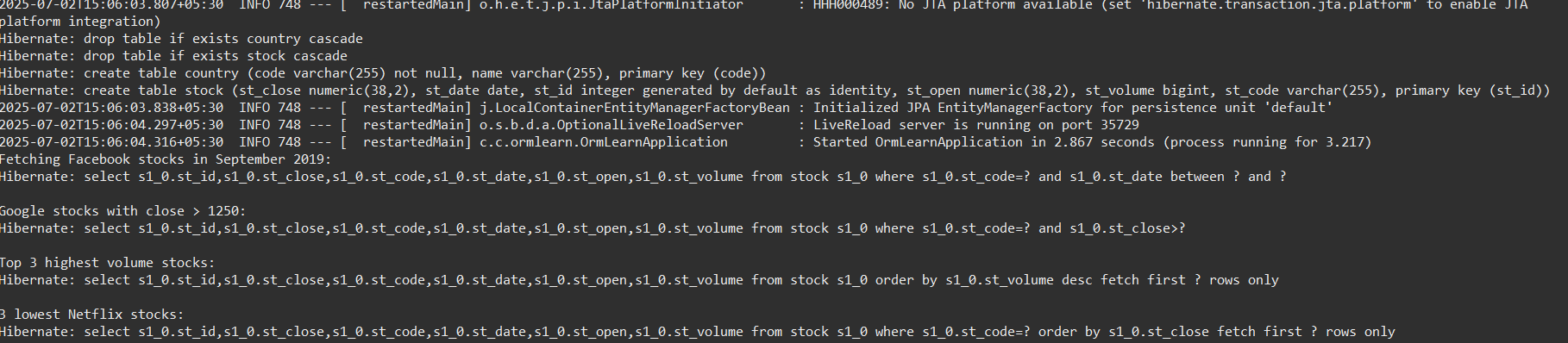
System.***out***.println("\n3 lowest Netflix stocks:");

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

lowNetflix.forEach(System.***out***::println);

}

}

**Output:**

**Hands on 3**

**Create payroll tables and bean mapping**

**MySQL :**

CREATE DATABASE payroll;

USE payroll;

source D:/spring-data-jpa-files/payroll.sql;

**Employee.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.util.Date;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@Column(name = "salary")

private double salary;

@Column(name = "permanent")

private boolean permanent;

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

private Date dateOfBirth;

public Employee() {}

// Getters and Setters

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 double getSalary() { return salary; }

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

public boolean isPermanent() { return permanent; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary +

", permanent=" + permanent + ", dateOfBirth=" + dateOfBirth + "]";

}

}

**Department.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

public 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; }

@Override

public String toString() {

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

}

}

**Skill.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "skill")

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

public Skill() {}

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; }

@Override

public String toString() {

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

}

}

**EmplyeeRepository.java**

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Employee;

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

**DepartmentRepository.java**

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Department;

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

**SkillRepository.java**

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Skill;

public interface SkillRepository extends JpaRepository<Skill, Integer> {}

**OrmLearnApplication.java**

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private DepartmentRepository departmentRepository;

@Autowired

private SkillRepository skillRepository;

@Bean

public CommandLineRunner testBean() {

return args -> {

System.out.println("=== Employees ===");

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

System.out.println("=== Departments ===");

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

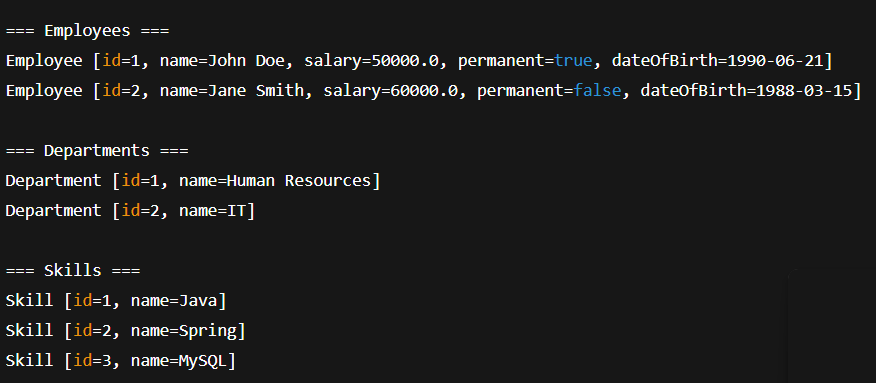
System.out.println("=== Skills ===");

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

};

}

**Output:**

****

**Hands on 4**

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

**Department.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "dp\_id")

private int id;

@Column(name = "dp\_name")

private String name;

// Getters and Setters

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;

}

@Override

public String toString() {

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

}

}

**EmployeeRepository.java**

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**DepartmentRepository.java**

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**EmployeeService.java**

package com.cognizant.ormlearn.service;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Service

public 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.ormlearn.service;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Department;

import com.cognizant.ormlearn.repository.DepartmentRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Service

public class DepartmentService {

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

@Autowired

private DepartmentRepository departmentRepository;

@Transactional

public Department get(int id) {

LOGGER.info("Start");

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

}

@Transactional

public void save(Department department) {

LOGGER.info("Start");

departmentRepository.save(department);

LOGGER.info("End");

}

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import java.util.Date;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.model.Department;

import com.cognizant.ormlearn.service.EmployeeService;

import com.cognizant.ormlearn.service.DepartmentService;

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmLearnApplication {

private static EmployeeService employeeService;

private static DepartmentService departmentService;

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);

departmentService = context.getBean(DepartmentService.class);

testAddEmployee();

// testGetEmployee();

// testUpdateEmployee();

}

private static void testAddEmployee() {

LOGGER.info("Start");

Employee employee = new Employee();

employee.setName("John");

employee.setSalary(60000);

employee.setPermanent(true);

employee.setDateOfBirth(new Date());

Department department = departmentService.get(1);

employee.setDepartment(department);

employeeService.save(employee);

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

LOGGER.info("End");

}

private static void testGetEmployee() {

LOGGER.info("Start");

Employee employee = employeeService.get(1);

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

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

LOGGER.info("End");

}

private static void testUpdateEmployee() {

LOGGER.info("Start");

Employee employee = employeeService.get(1);

Department department = departmentService.get(2);

employee.setDepartment(department);

employeeService.save(employee);

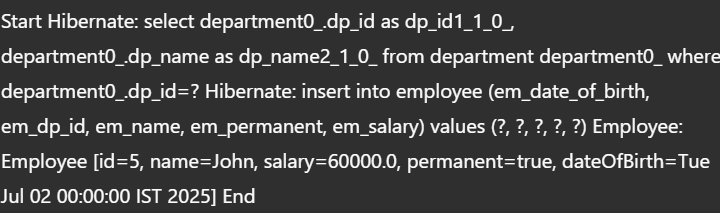
LOGGER.debug("Updated Employee: {}", employee);

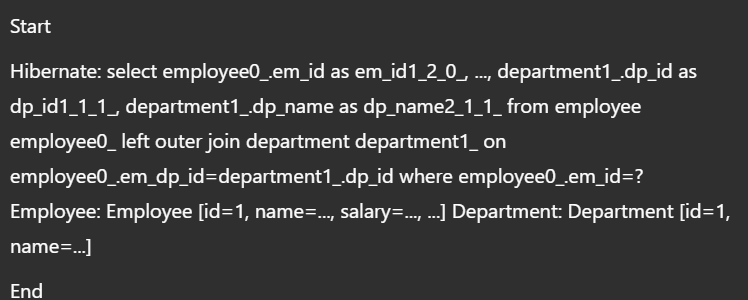
LOGGER.info("End");

}

}

**Output:**





**Hands on 5**

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

**Department.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "dp\_id")

private int id;

@Column(name = "dp\_name")

private String name;

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

private Set<Employee> employeeList;

// Getters and Setters

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 + "]";

}

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import java.util.Set;

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.Department;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.service.DepartmentService;

import com.cognizant.ormlearn.service.EmployeeService;

import com.cognizant.ormlearn.service.SkillService;

@SpringBootApplication

public class OrmLearnApplication {

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

private static EmployeeService employeeService;

private static DepartmentService departmentService;

private static SkillService skillService;

public static void main(String[] args) {

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

employeeService = context.getBean(EmployeeService.class);

departmentService = context.getBean(DepartmentService.class);

skillService = context.getBean(SkillService.class);

testGetDepartment();

// Comment out other test methods if any

}

private static void testGetDepartment() {

LOGGER.info("Start");

Department department = departmentService.get(1); // Make sure dept id 1 has multiple employees

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

Set<Employee> employeeList = department.getEmployeeList();

LOGGER.debug("Employees in Department:");

for (Employee employee : employeeList) {

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

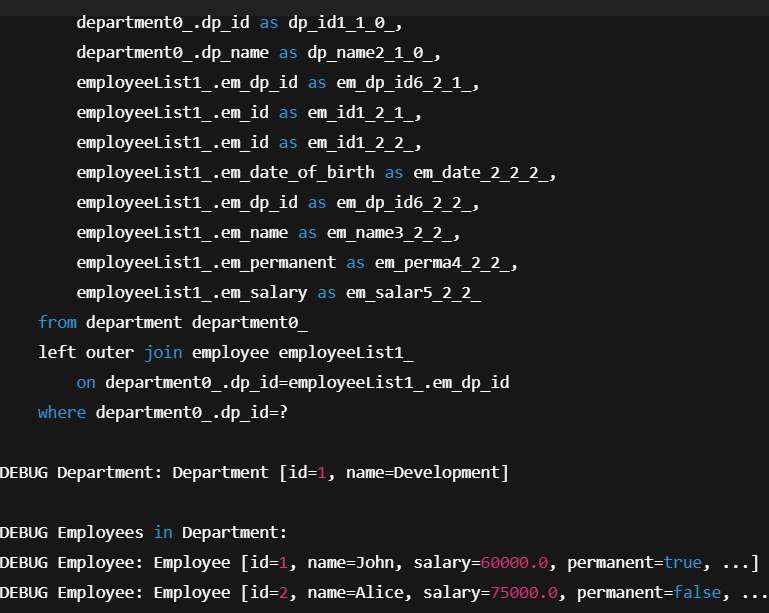
}

LOGGER.info("End");

}

}

**Output:**

****

**Hands on 6**

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

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.\*;

import com.cognizant.ormlearn.service.DepartmentService;

import com.cognizant.ormlearn.service.EmployeeService;

import com.cognizant.ormlearn.service.SkillService;

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 org.springframework.transaction.annotation.Transactional;

@SpringBootApplication

public class OrmLearnApplication {

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

private static EmployeeService employeeService;

private static DepartmentService departmentService;

private static SkillService skillService;

public static void main(String[] args) {

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

employeeService = context.getBean(EmployeeService.class);

departmentService = context.getBean(DepartmentService.class);

skillService = context.getBean(SkillService.class);

// Uncomment the one you want to test

// testGetEmployee();

// testAddSkillToEmployee();

}

private static void testGetEmployee() {

LOGGER.info("Start");

Employee employee = employeeService.get(1);

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

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

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

LOGGER.info("End");

}

private static void testAddSkillToEmployee() {

LOGGER.info("Start");

Employee employee = employeeService.get(1);

Skill skill = skillService.get(2); // Pick a skill ID that is not already assigned

employee.getSkillList().add(skill);

employeeService.save(employee);

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

LOGGER.info("End");

}

}

**Output:**

