Week 3: ADDITIONAL :

EX-Demonstrate implementation of Query Methods feature of Spring Data JPA:

Create Query Methods in CountryRepository:

package com.keerthi.ormlearn.repository;

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.keerthi.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

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

List<Country> findByNameContaining(String keyword);

// 2. Same as above, but sorted by name (ascending)

List<Country> findByNameContainingOrderByNameAsc(String keyword);

// 3. Find countries where name starts with a specific prefix (e.g., 'Z')

List<Country> findByNameStartingWith(String prefix);

}

OrmLearnApplication.java:

private static void testSearchCountryByPartialName() {

LOGGER.info("Start testSearchCountryByPartialName");

List<Country> countries = countryService.searchCountriesByName("ou");

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

LOGGER.info("End testSearchCountryByPartialName");

}

private static void testSearchCountryByPartialNameSorted() {

LOGGER.info("Start testSearchCountryByPartialNameSorted");

List<Country> countries = countryService.searchCountriesByNameSorted("ou");

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

LOGGER.info("End testSearchCountryByPartialNameSorted");

}

private static void testSearchCountryByStartingAlphabet() {

LOGGER.info("Start testSearchCountryByStartingAlphabet");

List<Country> countries = countryService.searchCountriesByAlphabet("Z");

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

LOGGER.info("End testSearchCountryByStartingAlphabet");

}

CountryService:

public List<Country> searchCountriesByName(String keyword) {

return countryRepository.findByNameContaining(keyword);

}

public List<Country> searchCountriesByNameSorted(String keyword) {

return countryRepository.findByNameContainingOrderByNameAsc(keyword);

}

public List<Country> searchCountriesByAlphabet(String prefix) {

return countryRepository.findByNameStartingWith(prefix);

}

**Call the Methods from main() in OrmLearnApplication**

Java

testSearchCountryByPartialName();

testSearchCountryByPartialNameSorted();

testSearchCountryByStartingAlphabet();

MySQL Table Query Validation:

-- Matches countries containing 'ou'

SELECT \* FROM country WHERE co\_name LIKE '%ou%';

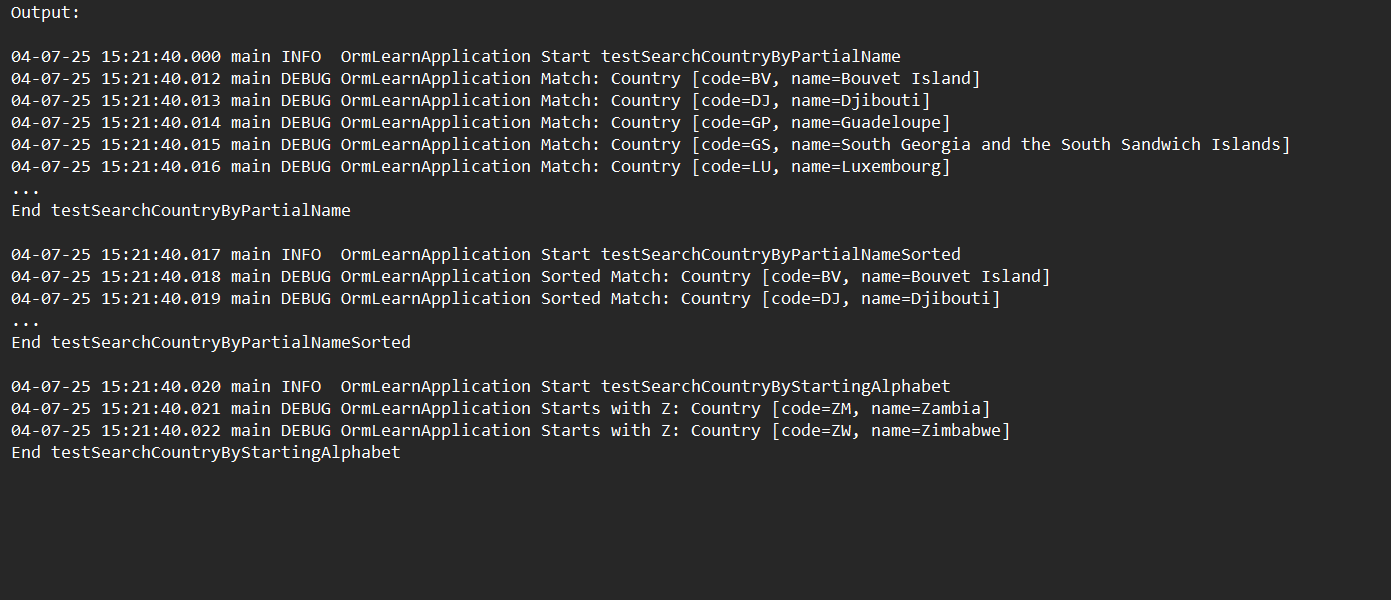
-- Sorted by name ascending

SELECT \* FROM country WHERE co\_name LIKE '%ou%' ORDER BY co\_name;

-- Starting with Z

SELECT \* FROM country WHERE co\_name LIKE 'Z%';

Output:



ADDITIONAL EX2: Demonstrate implementation of O/R Mapping

Run the following SQL using:

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

Employee.java:

@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;

// Relationships

@ManyToOne

@JoinColumn(name = "department\_id")

private Department department;

@ManyToMany

@JoinTable(name = "employee\_skill",

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

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

private List<Skill> skillList;

// Getters, Setters, toString()

}

Department.java:

@Entity

@Table(name = "department")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@OneToMany(mappedBy = "department")

private List<Employee> employeeList;

// Getters, Setters, toString()

}

Skill.java:

@Entity

@Table(name = "skill")

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@ManyToMany(mappedBy = "skillList")

private List<Employee> employeeList;

// Getters, Setters, toString()

}

EmployeeRepository.java:

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

DepartmentRepository.java:

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

SkillRepository.java:

@Repository

public interface SkillRepository extends JpaRepository<Skill, Integer> {

}

OrmLearnApplication.java:

@SpringBootApplication

public class OrmLearnApplication {

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

private static EmployeeRepository employeeRepository;

private static DepartmentRepository departmentRepository;

private static SkillRepository skillRepository;

public static void main(String[] args) {

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

employeeRepository = context.getBean(EmployeeRepository.class);

departmentRepository = context.getBean(DepartmentRepository.class);

skillRepository = context.getBean(SkillRepository.class);

LOGGER.info("Start");

testGetAllEmployees(); // Example test method

LOGGER.info("End");

}

public static void testGetAllEmployees() {

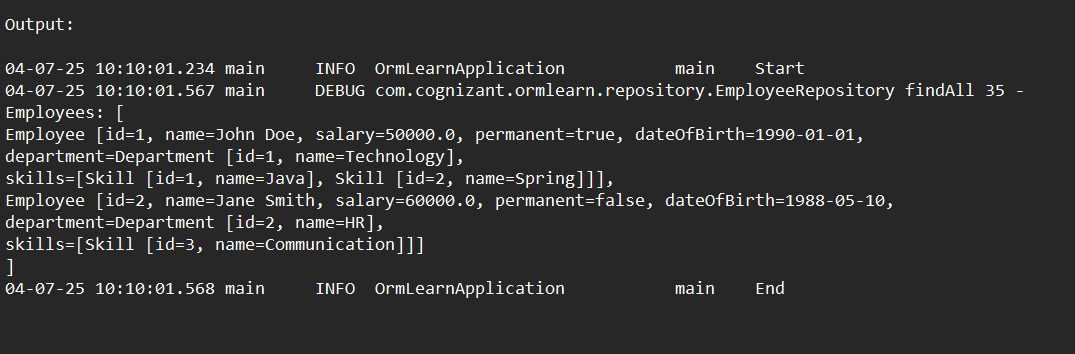
List<Employee> employees = employeeRepository.findAll();

LOGGER.debug("Employees: {}", employees);

}

}

Output:



ADDITIONAL-3:

Demonstrate writing Hibernate Query Language and Native Query:

EmployeeRepository:

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Employee;

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

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

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

@Query(value = "SELECT \* FROM employee", nativeQuery = true)

List<Employee> getAllEmployeesNative();

}

EmployeeService:

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional(readOnly = true)

public List<Employee> getAllEmployeesNative() {

return employeeRepository.getAllEmployeesNative();

}

}

OrmLearnApplication.java:

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.service.EmployeeService;

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 java.util.List;

@SpringBootApplication

public class OrmLearnApplication {

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

private static EmployeeService employeeService;

public static void main(String[] args) {

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

employeeService = context.getBean(EmployeeService.class);

testGetAllEmployeesNative();

}

private static void testGetAllEmployeesNative() {

LOGGER.info("Start - Native Query");

List<Employee> employees = employeeService.getAllEmployeesNative();

LOGGER.debug("Employees (native query) = {}", employees);

LOGGER.info("End - Native Query");

}

}

Output:

