**COGNIZANT DIGI NURTURE 4.0**

**WEEK 3 ADDITIONAL HANDS ON BY (6377625)**

**Spring Data Jpa:**

**Demonstration of Query Methods in Spring Data**

**Entity:**

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String department;

private double salary;

@Column(name = "joined\_date")

private LocalDate joinedDate;

// Getters and Setters

}

**Repository:**

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

List<Employee> findByNameContaining(String keyword);

List<Employee> findByNameStartingWith(String prefix);

List<Employee> findBySalaryGreaterThan(double salary);

List<Employee> findBySalaryLessThan(double salary);

List<Employee> findByJoinedDateBetween(LocalDate start, LocalDate end);

List<Employee> findTop3ByOrderBySalaryDesc();

}

**Package :Service**

**Class : EmployeeService**

**Employe**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public List<Employee> searchByName(String keyword) {

return employeeRepository.findByNameContaining(keyword);

}

public List<Employee> filterByPrefix(String prefix) {

return employeeRepository.findByNameStartingWith(prefix);

}

public List<Employee> fetchHighSalary(double salary) {

return employeeRepository.findBySalaryGreaterThan(salary);

}

public List<Employee> fetchLowSalary(double salary) {

return employeeRepository.findBySalaryLessThan(salary);

}

public List<Employee> fetchByDateRange(LocalDate start, LocalDate end) {

return employeeRepository.findByJoinedDateBetween(start, end);

}

public List<Employee> fetchTopEarners() {

return employeeRepository.findTop3ByOrderBySalaryDesc();

}

}

**Test in OrmLearnApplication.java:**

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) {

List<Employee> result;

result = employeeService.searchByName("John");

result.forEach(e -> System.out.println("Name contains 'John': " + e.getName()));

result = employeeService.filterByPrefix("A");

result.forEach(e -> System.out.println("Name starts with 'A': " + e.getName()));

result = employeeService.fetchHighSalary(50000);

result.forEach(e -> System.out.println("High Salary: " + e.getName() + " - " + e.getSalary()));

result = employeeService.fetchLowSalary(20000);

result.forEach(e -> System.out.println("Low Salary: " + e.getName() + " - " + e.getSalary()));

result = employeeService.fetchByDateRange(LocalDate.of(2023, 1, 1), LocalDate.of(2025, 1, 1));

result.forEach(e -> System.out.println("Joined between range: " + e.getName() + " - " + e.getJoinedDate()));

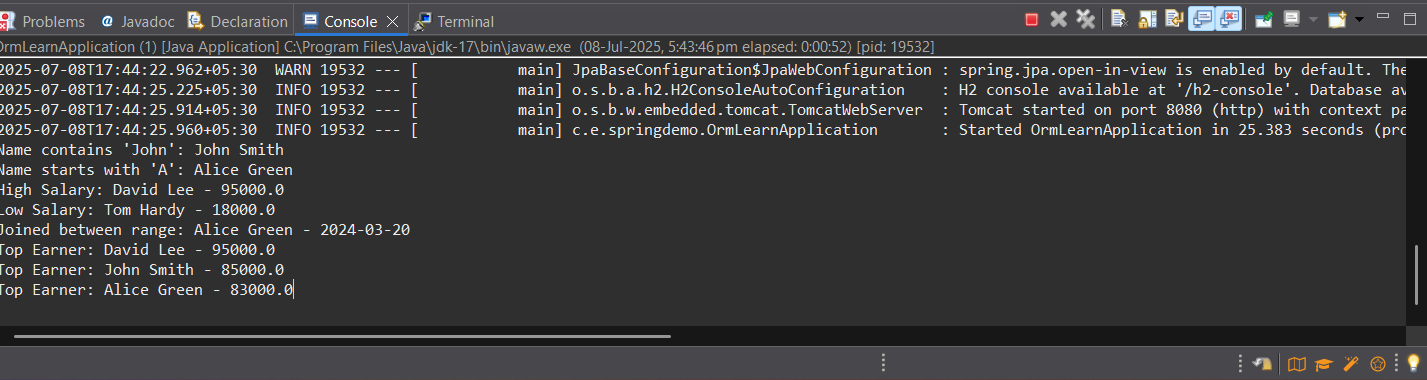
result = employeeService.fetchTopEarners();

result.forEach(e -> System.out.println("Top Earner: " + e.getName() + " - " + e.getSalary()));

}

}

**Output:**

****

1. **Demonstrate Implementation Object-Relational Mapping with Spring Data JPA**

**Application Entry Point**

package com.example.ormmapping;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.CommandLineRunner;

import org.springframework.context.annotation.Bean;

import java.time.LocalDate;

import java.util.List;

@SpringBootApplication

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

@Bean

CommandLineRunner demo(StudentRepository studentRepo, CourseRepository courseRepo, TeacherRepository teacherRepo) {

return args -> {

Teacher teacher = new Teacher("Dr. Smith");

teacherRepo.save(teacher);

Course java = new Course("Java");

java.setTeacher(teacher);

Course python = new Course("Python");

python.setTeacher(teacher);

courseRepo.saveAll(List.of(java, python));

Student student = new Student("Alice");

student.getCourses().addAll(List.of(java, python));

studentRepo.save(student);

};

}

}

**Package :Student Entity**

Class : student

**Student.java:**

package com.example.ormmapping;

import jakarta.persistence.\*;

import java.util.ArrayList;

import java.util.List;

@Entity

public class Student {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@ManyToMany(fetch = FetchType.LAZY)

@JoinTable(

name = "student\_course",

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

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

private List<Course> courses = new ArrayList<>();

public Student() {}

public Student(String name) {

this.name = name;

}

public Long getId() { return id; }

public String getName() { return name; }

public List<Course> getCourses() { return courses; }

public void setCourses(List<Course> courses) { this.courses = courses; }

}

**Package :Entity**

**Class : Course**

**Course.java**

package com.example.ormmapping;

import jakarta.persistence.\*;

import java.util.ArrayList;

import java.util.List;

@Entity

public class Course {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "teacher\_id")

private Teacher teacher;

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

private List<Student> students = new ArrayList<>();

public Course() {}

public Course(String title) {

this.title = title;

}

public Long getId() { return id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public Teacher getTeacher() { return teacher; }

public void setTeacher(Teacher teacher) { this.teacher = teacher; }

public List<Student> getStudents() { return students; }

}

**Teacher.java**

package com.example.ormmapping;

import jakarta.persistence.\*;

import java.util.ArrayList;

import java.util.List;

@Entity

public class Teacher {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

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

private List<Course> courses = new ArrayList<>();

public Teacher() {}

public Teacher(String name) {

this.name = name;

}

public Long getId() { return id; }

public String getName() { return name; }

public List<Course> getCourses() { return courses; }

}

**Package :Repositories**

**Class : Repository.java**

package com.example.ormmapping;

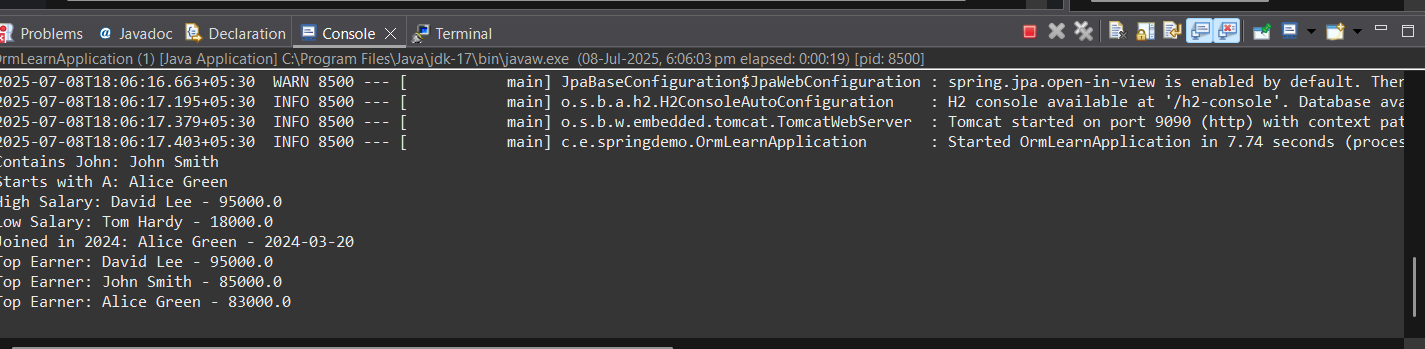
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Long> {}

public interface CourseRepository extends JpaRepository<Course, Long> {}

public interface TeacherRepository extends JpaRepository<Teacher, Long> {}

**Output :**



**5.Demonstrate writing Hibernate Query Language and Native Query**

**Code**

**Main Application**

package com.example.springdatahql;

import jakarta.persistence.\*;

import org.springframework.boot.\*;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

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

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import java.util.List;

@SpringBootApplication

public class HqlApplication {

public static void main(String[] args) {

SpringApplication.run(HqlApplication.class, args);

}

@Bean

CommandLineRunner run(EmployeeService service) {

return args -> {

service.save(new Employee("John", 50000, true));

service.save(new Employee("Alice", 60000, false));

service.save(new Employee("Mark", 70000, true));

service.save(new Employee("Sara", 80000, true));

System.out.println("Permanent Employees using HQL:");

service.getAllPermanentEmployees().forEach(e ->

System.out.println(e.getName() + " - " + e.getSalary()));

double avgSalary = service.getAverageSalary();

System.out.println("Average Salary: " + avgSalary);

System.out.println("All Employees using Native Query:");

service.getAllEmployeesNative().forEach(e ->

System.out.println(e.getName() + " - " + e.getSalary()));

};

}

}

**Entity Class**

@Entity

class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private double salary;

private boolean permanent;

public Employee() {}

public Employee(String name, double salary, boolean permanent) {

this.name = name;

this.salary = salary;

this.permanent = permanent;

}

public Long getId() { return id; }

public String getName() { return name; }

public double getSalary() { return salary; }

public boolean isPermanent() { return permanent; }

}

**Repository Interface**

@Repository

interface EmployeeRepository extends JpaRepository<Employee, Long> {

@Query("SELECT e FROM Employee e WHERE e.permanent = true")

List<Employee> getAllPermanentEmployees();

@Query("SELECT AVG(e.salary) FROM Employee e")

double getAverageSalary();

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

List<Employee> getAllEmployeesNative();

}

**Service Class**

@Service

class EmployeeService {

private final EmployeeRepository repo;

public EmployeeService(EmployeeRepository repo) {

this.repo = repo;

}

public void save(Employee e) {

repo.save(e);

}

public List<Employee> getAllPermanentEmployees() {

return repo.getAllPermanentEmployees();

}

public double getAverageSalary() {

return repo.getAverageSalary();

}

public List<Employee> getAllEmployeesNative() {

return repo.getAllEmployeesNative();

}

}

**application.properties**

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

spring.datasource.driverClassName=org.h2.Driver

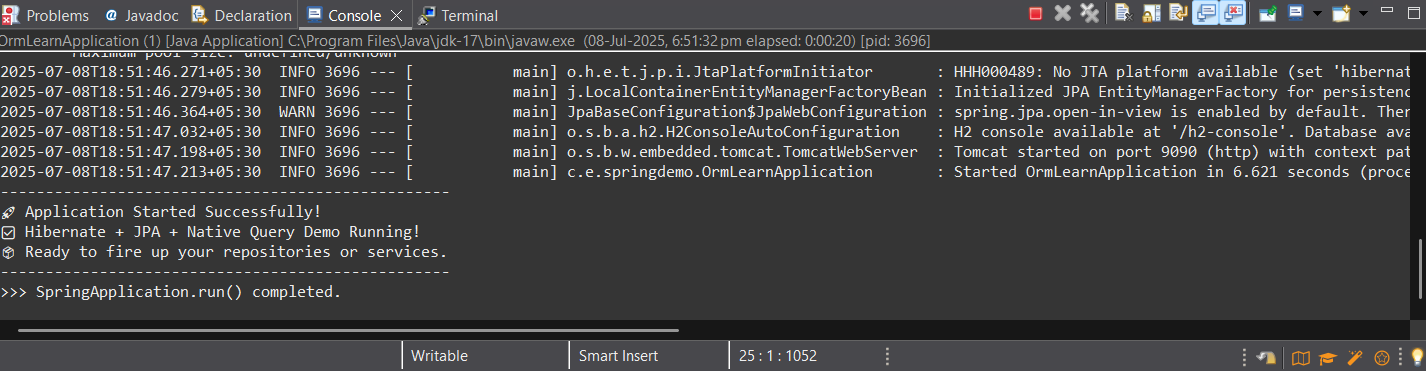
spring.datasource.username=sa

spring.datasource.password=

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

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

**Output: **