**3.SPRINGDATAJPA**

**Project structure**

hibernate-jpa-query-demo/

├── pom.xml # Maven build configuration

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ └── com/example/demo/

│ │ │ ├── DemoApplication.java # Main Spring Boot class

│ │ │

│ │ │ ├── entity/

│ │ │ │ ├── Employee.java # Employee entity

│ │ │ │ └── Department.java # Department entity

│ │ │

│ │ │ ├── repository/

│ │ │ │ ├── EmployeeRepository.java # HQL, JPQL, Native Queries

│ │ │ │ └── DepartmentRepository.java

│ │ │

│ │ │ ├── service/

│ │ │ │ └── QueryService.java # Criteria API logic

│ │ │

│ │ │ └── controller/

│ │ │ └── QueryController.java # Exposes REST APIs

│ │

│ │ └── resources/

│ │ ├── application.properties # DB & JPA config

│ │ └── data.sql # Initial test data

│

└── target/ # Generated when you build the project

**Department.java**

package com.example.demo.entity;

import javax.persistence.\*;

import java.util.\*;

@Entity

public class Department {

    @Id

    @GeneratedValue

    private Long id;

    private String name;

    @OneToMany(mappedBy = "department", cascade = CascadeType.ALL)

    private List<Employee> employees = new ArrayList<>();

    public Long getId() { return id; }

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

    public String getName() { return name; }

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

    public List<Employee> getEmployees() { return employees; }

    public void setEmployees(List<Employee> employees) { this.employees = employees; }

}

**Employee.java**

package com.example.demo.entity;

import javax.persistence.\*;

@Entity

public class Employee {

    @Id

    @GeneratedValue

    private Long id;

    private String name;

    private double salary;

    @ManyToOne(fetch = FetchType.LAZY)

    private Department department;

    public Long getId() { return id; }

    public void setId(Long 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 Department getDepartment() { return department; }

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

}

**DepartmentRepository.java**

package com.example.demo.repository;

import com.example.demo.entity.Department;

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

import org.springframework.stereotype.Repository;

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Long> {

}

**EmployeeRepository.java**

package com.example.demo.repository;

import com.example.demo.entity.Employee;

import com.example.demo.entity.Department;

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

    @Query("SELECT e FROM Employee e WHERE e.salary > :salary")

    List<Employee> findHighEarners(@Param("salary") double salary);

    @Query(value = "SELECT \* FROM employee WHERE salary < ?1", nativeQuery = true)

    List<Employee> findLowEarners(double salary);

    @Query("SELECT COUNT(e) FROM Employee e WHERE e.salary > :salary")

    long countHighEarners(@Param("salary") double salary);

    @Query("SELECT d FROM Department d JOIN FETCH d.employees")

    List<Department> fetchDepartmentsWithEmployees();

}

**QueryService.java**

package com.example.demo.service;

import com.example.demo.entity.Employee;

import org.springframework.stereotype.Service;

import javax.persistence.\*;

import javax.persistence.criteria.\*;

import java.util.List;

@Service

public class QueryService {

    @PersistenceContext

    private EntityManager em;

    public List<Employee> getEmployeesUsingCriteria(double minSalary) {

        CriteriaBuilder cb = em.getCriteriaBuilder();

        CriteriaQuery<Employee> cq = cb.createQuery(Employee.class);

        Root<Employee> root = cq.from(Employee.class);

        cq.select(root).where(cb.gt(root.get("salary"), minSalary));

        return em.createQuery(cq).getResultList();

    }

}

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

    public static void main(String[] args) {

        SpringApplication.run(DemoApplication.class, args);

    }

}

**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=update

**data.sql**

INSERT INTO department (id, name) VALUES (1, 'HR'), (2, 'Engineering');

INSERT INTO employee (id, name, salary, department\_id) VALUES

(1, 'Alice', 90000, 2),

(2, 'Bob', 40000, 1),

(3, 'Charlie', 75000, 2),

(4, 'Diana', 30000, 1);

**Pom.xml**

<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

         http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>hibernate-jpa-query-demo</artifactId>

  <version>1.0.0</version>

  <packaging>jar</packaging>

  <name>Hibernate JPA Query Demo</name>

  <description>Spring Boot Demo with HQL, JPQL, Native Query, Criteria API</description>

  <parent>

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

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

    <version>2.7.1</version>

  </parent>

  <dependencies>

    <dependency>

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

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

    </dependency>

    <dependency>

      <groupId>com.h2database</groupId>

      <artifactId>h2</artifactId>

      <scope>runtime</scope>

    </dependency>

    <dependency>

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

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

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <plugin>

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

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

      </plugin>

    </plugins>

  </build>

</project>

**Output:**

