**Date : 21/10/2020**

**Spring Boot 7AM**

**Mr. RAGHU**

**-----------------------------------------------**

Workspace:

http://www.mediafire.com/file/iu4425ni0l0e576/SpringBoot7AM\_21102020\_RAGHU.zip/file

Videos:

https://www.youtube.com/watch?v=EA43S5R8LSc&list=PLVlQHNRLflP9XSWeY4x4FLwnL3UOIxnTr

---------------------------------------------------------

**Working with HQL Joins coding part**

HQL Join Syntax:

SELECT p.variable,c.variable

FROM ParentModelclass p

Join Type

p.hasAvariableName as C

WHERE <condition>;

=> we can execute this Joins query using @Query annotation.

-----------------------------------------------------------------

1...\*

Employee-----<>Project

HAS-A

Step#1 Create Application and insert data into DB tables

1. Model class

package in.nareshit.raghu.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

public class Project {

@Id

private Integer pid;

private String pcode;

private Double pcost;

private String client;

}

-----

package in.nareshit.raghu.model;

import java.util.List;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToMany;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Entity

public class Employee {

@Id

private Integer eid;

private String ename;

private Double esal;

private String addr;

@OneToMany

@JoinColumn(name="eidFk")

private List<Project> prjs;

}

-------

2) Repository Interfaces

package in.nareshit.raghu.repo;

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

import in.nareshit.raghu.model.Project;

public interface ProjectRepository

extends JpaRepository<Project, Integer> {

}

-------

package in.nareshit.raghu.repo;

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

import in.nareshit.raghu.model.Employee;

public interface EmployeeRepository

extends JpaRepository<Employee, Integer> {

}

---

3)Runner class to insert data

package in.nareshit.raghu.runner;

import java.util.Arrays;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import in.nareshit.raghu.model.Employee;

import in.nareshit.raghu.model.Project;

import in.nareshit.raghu.repo.EmployeeRepository;

import in.nareshit.raghu.repo.ProjectRepository;

@Component

public class DataInsertRunner implements CommandLineRunner {

@Autowired

private EmployeeRepository erepo;

@Autowired

private ProjectRepository prepo;

@Override

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

Project p1 = new Project(55, "P1", 5000.0, "HTC");

Project p2 = new Project(56, "P2", 6000.0, "NIT");

Project p3 = new Project(57, "P3", 5800.0, "XYZ");

Project p4 = new Project(58, "P4", 4200.0, "NIT");

Project p5 = new Project(59, "P5", 9800.0, "HTC");

prepo.save(p1);

prepo.save(p2);

prepo.save(p3);

prepo.save(p4);

prepo.save(p5);

Employee e1 = new Employee(10, "A", 3.3, "HYD", null);

Employee e2 = new Employee(11, "B", 3.4, "CHN", null);

Employee e3 = new Employee(12, "C", 4.6, "DHL", Arrays.asList(p2));

Employee e4 = new Employee(13, "D", 5.8, "BAN", null);

Employee e5 = new Employee(14, "E", 6.8, "HYD", Arrays.asList(p3));

erepo.save(e1);

erepo.save(e2);

erepo.save(e3);

erepo.save(e4);

erepo.save(e5);

}

}

4. application.properties

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/boot7am

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=create

===================================================================

Step#2 Define @Query methods in Parent Repository using Joins Concept.

Query#1 Fetch employee name,addr and project code based on client name input.

(Get only connected data) Get only projects assigned to employees.

(Inner Join Ex)

SELECT e.ename,e.addr,p.pcode

FROM Employee e

INNER JOIN

e.prjs AS p

WHERE p.client=?

=> Only connected --- inner join

=> All from <one table> and assigned/connected/linked from another <table> -- Left/Right joins

=> Everything -- full join

\*) HAS-A vaiable indicates a relation between two classes. HQL never supports taking directly className for child, it makes INAVILD/NO MEANING.

Here, HAS-A Relation creates a link even b/w tables as FK column.

List<T> = All columns

List<DT> = one column

List<Object[]> = multiple columns

---code---

1) Modified EmployeeRepository

package in.nareshit.raghu.repo;

import java.util.List;

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

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

import in.nareshit.raghu.model.Employee;

public interface EmployeeRepository

extends JpaRepository<Employee, Integer> {

@Query("SELECT e.ename,e.addr,p.pcode FROM Employee e INNER JOIN e.prjs AS p WHERE p.client=:client")

List<Object[]> getDataByClient(String client);

@Query("SELECT e.ename,e.addr,p.pcode FROM Employee e RIGHT JOIN e.prjs AS p WHERE p.client=:client")

List<Object[]> getDataByClientRJ(String client);

@Query("SELECT e.ename,e.addr,p.pcode FROM Employee e LEFT OUTER JOIN e.prjs AS p WHERE e.addr=:addr")

//@Query("SELECT e.ename,e.addr,p.pcode FROM Employee e JOIN e.prjs AS p WHERE e.addr=:addr")

//@Query("SELECT e.ename,e.addr,p.pcode FROM Employee e FULL JOIN e.prjs AS p WHERE e.addr=:addr")

List<Object[]> getDataByClientLJ(String addr);

}

2) Runner class for Testing:

package in.nareshit.raghu.runner;

import java.util.List;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import in.nareshit.raghu.repo.EmployeeRepository;

@Component

public class TestQueryRunner implements CommandLineRunner {

@Autowired

private EmployeeRepository repo;

@Override

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

List<Object[]> list = repo.getDataByClientLJ("HYD");

//List<Object[]> list = repo.getDataByClientRJ("NIT");

//List<Object[]> list = repo.getDataByClient("NIT");

for(Object[] ob:list) {

System.out.println(ob[0]+"-"+ob[1]+"-"+ob[2]);

}

}

}

========================================================================

**Note:**

In real time most of the scenarios that can be done by using inner join and left outer join only.

Full outer join can be done by findAll() and findById() methods, so there is useless to use full outer join here.

========================================================================

**Task:**

\*...1

Student---<>Course

=> insert student data and course data

=> Write all possible join types using @Query

=> Write Runner to call method and display data.

https://github.com/javabyraghu?tab=repositories&q=hib&type=&language=