Date : 09/02/2021
Spring Boot 9AM
Mr. RAGHU

Spring Boot : Data JPA [JOINS]

*) To Fetch Data from multiple tables using Single SQL query.

SQL Joins:

```
SELECT  <P>.<COLUMN> , <C>.<COLUMN>

FROM  <PARENT-TABLE> <P>
     [JOIN - TYPE ]
      <CHILD-TABLE>  <C>
ON <P>.<PK> = <C>.<FK> | <P>.<FK> = <C>.<PK>

WHERE <CONDITION> ;
```

HQL/JPQL Joins:-

```
SELECT  <P>.<VARIABLES> , <C>.<VARIABLES>

FROM  <PARENT-CLASS> <P>
     [JOIN - TYPE ]
  <P>.<HAS-A VARIABLE> AS <C>

WHERE <CONDITION> ;
```

*) <P>, <C> ARE alias names given for model classes.

```
                *...1
            Employee -----<> Project
```

```
class Project { //child class
  int pid;
  String pcode
}
class Employee {  //parent class
  int eid;
  String ename;

  @ManyToOne
  @JoinColumn(name="__")
  Project pob;//HAS-A
}
```

Inner Join: HQL/JPQL Syntax:

```
*)
SELECT E.ename, P.pcode

FROM Employee  E
    INNER JOIN
   E.pob  AS   P
```

*) EMPLOYESS WHICH ARE NOT CONNECTED TO PROJECT.

```
SELECT E.ename, P.pcode

FROM Employee  E
    LEFT OUTER JOIN
   E.pob  AS   P

WHERE P IS NULL;
```

*) HQL/JPQL Syntax is same for collection/non-collection type.
   But output is different.
*) HQL/JPQL finally converted to SQL by dialect.
*) SQL is DB dependent where HQL/JPQL is db independent.
*) word 'AS' indicates alias name (Also Know As) which is optional.
*) Words related to SQL are case-insensitive (SELECT,FROM,WHERE).
   Words related to java are case-sensitive (Employee, empId..)

*) INNER JOIN and JOIN gets same result. Word INNER is Optional.

*) FULL JOIN not supported by few Databases and Data JPA also.
   As it is meaning less join that gets all rows from DB tables.
   Better use findAll();

```
QuerySyntaxException: expecting "join", found 'OUTER' near line 1,
column 71 [SELECT P.pcode , V.vcode  FROM
in.nareshit.raghu.model.Product P FULL OUTER JOIN P.vob AS V ]
```

Dialect: Oracle9iDialect, Oracle10gDialect.

```
--Full Code--------------------
Name : SpringBoot2DataJpaManyToOne
Dep  : Data Jpa, MySQL, Lombok
```

1. Models

```
package in.nareshit.raghu.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="ven_tab")
public class Vendor {

        @Id
        @Column(name="vid_col")
        private Integer vid;

        @Column(name="vcode_col")
```

```
        private String vcode;

        @Column(name="vloc_col")
        private String vloc;

}
---
package in.nareshit.raghu.model;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="prod_tab")
public class Product {
        @Id
        @Column(name="pid_col")
        private Integer pid;

        @Column(name="pcode_col")
        private String pcode;

        @Column(name="pcost_col")
        private Double pcost;

        @Column(name="pmodel_col")
        private String pmodel;

        @ManyToOne(fetch = FetchType.LAZY,cascade = CascadeType.ALL)
        @JoinColumn(name="vidFk")
        private Vendor vob;// HAS-A



}

2. Repository interfaces
package in.nareshit.raghu.repo;

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

import in.nareshit.raghu.model.Vendor;

public interface VendorRepository
```

```
        extends JpaRepository<Vendor, Integer>
{

}
---
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.Product;

public interface ProductRepository
        extends JpaRepository<Product, Integer>
{

        //@Query("SELECT P.pcode , V.vcode  FROM Product P INNER JOIN
P.vob AS V ")
        //@Query("SELECT P.pcode , V.vcode  FROM Product P LEFT OUTER
JOIN P.vob AS V ")
        @Query("SELECT P.pcode , V.vcode  FROM Product P RIGHT JOIN
P.vob AS V ")
        //@Query("SELECT P.pcode , V.vcode  FROM Product P FULL OUTER
JOIN P.vob AS V ")
        public List<Object[]> getProductAndVendorCodes();
}

3. Runner classes
package in.nareshit.raghu.runner;

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

import in.nareshit.raghu.model.Product;
import in.nareshit.raghu.model.Vendor;
import in.nareshit.raghu.repo.ProductRepository;
import in.nareshit.raghu.repo.VendorRepository;

//@Component
public class DataInsertRunner implements CommandLineRunner {
        @Autowired
        private ProductRepository prepo;

        @Autowired
        private VendorRepository vrepo;

        @Override
        public void run(String... args) throws Exception {
                Vendor v1 = new Vendor(101, "ABC", "HYD");
                Vendor v2 = new Vendor(102, "NIT", "DHL");
                Vendor v3 = new Vendor(103, "PQR", "CHN");
                Vendor v4 = new Vendor(104, "IJK", "MUM");

                vrepo.save(v1);
                vrepo.save(v2);
```

```
		vrepo.save(v3);
		vrepo.save(v4);

		Product p1 = new Product(10, "PEN", 20.0, "A", v1);
		Product p2 = new Product(11, "BOOK", 40.0, "B", null);

		Product p3 = new Product(12, "BTL", 80.0, "A", null);
		Product p4 = new Product(13, "INK", 50.0, "A", v3);

		prepo.save(p1);
		prepo.save(p2);
		prepo.save(p3);
		prepo.save(p4);


	}

}
--
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.ProductRepository;

@Component
public class TestJoinRunner implements CommandLineRunner {

	@Autowired
	private ProductRepository repo;

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

		List<Object[]> list = repo.getProductAndVendorCodes();

		for(Object[] ob:list) {
			System.out.println(ob[0]+"-"+ob[1]);
		}

	}

}
==========================================================================
	Full Code Ex#2

Name:  SpringBoot2DataJpaOneToMany
Dep : Data Jpa, MySQL, Lombok

1. Model
package in.nareshit.raghu.model;

import javax.persistence.Column;
```

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="emptab")
public class Employee {
        @Id
        @Column(name="eid_col")
        private Integer eid;

        @Column(name="ename_col")
        private String ename;

        @Column(name="esal_col")
        private Double esal;
}
--
package in.nareshit.raghu.model;

import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="depttab")
public class Dept {

        @Id
        @Column(name="did_col")
        private Integer did;

        @Column(name="dcode_col")
        private String deptCode;

        @Column(name="aname_col")
```

```
        private String adminName;

        @OneToMany(cascade = CascadeType.ALL,fetch = FetchType.EAGER)
        @JoinColumn(name="didFk")
        private List<Employee> emps;
}
---
2. Repository
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> {

}
---
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.Dept;

public interface DeptRepo
        extends JpaRepository<Dept, Integer> {

        //@Query("SELECT D.deptCode,E.ename FROM Dept D INNER JOIN
D.emps AS E")
        //@Query("SELECT D.deptCode,E.ename FROM Dept D LEFT OUTER
JOIN D.emps AS E")
        @Query("SELECT D.deptCode,E.ename FROM Dept D RIGHT OUTER JOIN
D.emps AS E")
        public List<Object[]> getDeptEmpData();
}

3. Runner class
package in.nareshit.raghu.runner;

import java.util.Arrays;

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

import in.nareshit.raghu.model.Dept;
import in.nareshit.raghu.model.Employee;
import in.nareshit.raghu.repo.DeptRepo;
import in.nareshit.raghu.repo.EmployeeRepository;

//@Component
public class DataInsertRunner implements CommandLineRunner {

        @Autowired
```

```
	private EmployeeRepository erepo;

	@Autowired
	private DeptRepo drepo;

	@Override
	public void run(String... args) throws Exception {
		Employee e1 = new Employee(10, "A", 3.3);
		Employee e2 = new Employee(11, "B", 4.3);
		Employee e3 = new Employee(12, "C", 5.3);
		Employee e4 = new Employee(13, "D", 6.6);
		Employee e5 = new Employee(14, "E", 8.6);

		erepo.save(e1);
		erepo.save(e2);
		erepo.save(e3);
		erepo.save(e4);
		erepo.save(e5);

		Dept d1 = new Dept(521, "DEV", "SAM",
Arrays.asList(e1,e2));
		Dept d2 = new Dept(522, "QA", "SYED",
Arrays.asList(e2,e4));
		Dept d3 = new Dept(523, "BA", "AJAY", null);

		drepo.save(d1);
		drepo.save(d2);
		drepo.save(d3);


	}

}
---
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.DeptRepo;

@Component
public class TestJoinsRunner implements CommandLineRunner {
	@Autowired
	private DeptRepo repo;

	@Override
	public void run(String... args) throws Exception {
		List<Object[]> list = repo.getDeptEmpData();

		for( Object[] ob:list ) {
			System.out.println(ob[0]+"-"+ob[1]);
		}
	}
```

```
}

--yaml file
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/boot9am
    username: root
    password: root
  jpa:
    show-sql: true
    hibernate:
      ddl-auto: update
    database-platform: org.hibernate.dialect.MySQL8Dialect
```
