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

Spring Data JPA : Association Mapping

----Association Mapping Implementation Steps---
a) write Two Model classes and apply
HAS-A Relation between Model classes
b) Check For Collection/Non-Collection Type
if Collection type then modify HAS-A Variable
as Collection variable.

c) Apply Multiplicity Annotation

1...1 @ManyToOne + Unique
1...* @OneToMany
*...1 @ManyToOne
... @ManyToMany

d) Provide JoinColumn (FK Column) or JoinTable(JoinColumn + JoinColumn)

Many-To-One (*...1)

HAS-A
Product -----<> Vendor
*...1

a. Define Vendor Model class
b. Define Product Model class
c. Create (HAS-A) variable of Vendor inside Product
d. It is non-collection type, keep has-a variable as it is.
e. Apply @ManyToOne Annotation on has-a variable
f. Apply @JoinColumn(name="__") on has-a variable
g. Define Repository Interfaces for both model classes
h. Define Runner class for Data Insert
i. Check DB Tables finally for result.

----------Full Code----------
Name : SpringBoot2ManyToOne
Dep : Spring Data JPA, Lombok, MySQL

1. Model classes

```
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.Column;
import javax.persistence.Entity;
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
	@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 org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.model.Product;

public interface ProductRepository
        extends JpaRepository<Product, Integer>
{

}
```

3. Runner class

```
package in.nareshit.raghu.runner;

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

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");

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

                //Product p1 = new Product(10, "PEN", 20.0, "A",
vrepo.findById(101).get());
                Product p1 = new Product(10, "PEN", 20.0, "A", v1);
```

```
                //Product p2 = new Product(11, "BOOK", 40.0, "B",
null);
                Product p2 = new Product(11, "BOOK", 40.0, "B", v1);

                //Product p3 = new Product(12, "BTL", 80.0, "A", null);
                Product p3 = new Product(12, "BTL", 80.0, "A", v2);
                Product p4 = new Product(13, "INK", 50.0, "A", v2);

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


        }

}
```

4. application.yml

```
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: create
    database-platform: org.hibernate.dialect.MySQL8Dialect
```

---

*) Note:

-> We can use any column name for JoinColumn (need not be same name as like child table PK column).

-> Writing @JoinColumn is optional. If we do not provide this then Join Column name is :

hasAVariableName_PrimaryKeyColumnNameInChildTable

ex: vob_vid_col

-> If we do not provide @ManyToOne then Exception is raised:

```
MappingException:
  Could not determine type for: in.nareshit.raghu.model.Vendor
```

---

one-to-many

```
        1...*
Dept -----<> Employee
        HAS-A
```

a. Define Employee Model class
b. Define Dept Model class
c. Create (HAS-A) variable of Employee inside Dept.
d. It is collection type, modify has-a variable as List.
e. Apply @OneToMany Annotation on has-a variable

f. Apply @JoinColumn(name="__") on has-a variable
g. Define Repository Interfaces for both model classes
h. Define Runner class for Data Insert
i. Check DB Tables finally for result.

---Full code---------------
Name : SpringBoo2OneToMany
Dep : Data Jpa, MySQL, Lombok

1. Model classes

```
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.Column;
import javax.persistence.Entity;
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
	@JoinColumn(name="didFk")
	private List<Employee> emps;
}
```

2. Repsitory interface

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

import in.nareshit.raghu.model.Dept;

public interface DeptRepo
	extends JpaRepository<Dept, Integer> {

}
```

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 org.springframework.stereotype.Component;

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);

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

		Dept d1 = new Dept(521, "DEV", "SAM",
Arrays.asList(e1,e2));
		Dept d2 = new Dept(522, "QA", "SYED",
Arrays.asList(e3,e4));

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


	}

}
```

4. application.yml

```
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: create
    database-platform: org.hibernate.dialect.MySQL8Dialect
```

Hibernate Q&A

a) What is Cascading ?

b) what is FetchType ?