Date : 26/01/2021
Spring Boot 9AM
Mr. RAGHU

Spring Data JPA: Database Operation

=> JpaRepository :
   Custom Query(findBy, @Query).
   For SQL DBs.
   JPA Annotations : @Temporal, @Lob, @ElementCollection..etc
   Calling Procedures/Functions.
   Association Mapping (1...1/1...*/....) with JOINS.
   Native SQL (Pure SQL based on DB without Dialect).

----------Code------------------------
1. Temporal (Working with Date and Time)

=> Tow work with Date and Time we can use this annotation with enum
   TemporalType(DATE,TIME,TIMESTAMP**)

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

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Lob;

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

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Product {
        @Id
        private Integer pid;
        private String pcode;
        private Double pcost;


        @Temporal(TemporalType.DATE)
        private Date dteA;
        @Temporal(TemporalType.TIME)
        private Date dteB;
        @Temporal(TemporalType.TIMESTAMP)
        private Date dteC;

}
--Repo-
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> {

}
--Runner--
package in.nareshit.raghu.runner;

import java.io.FileInputStream;

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.repo.ProductRepository;

@Component
public class TestJpaRepoRunner implements CommandLineRunner {

        @Autowired
        private ProductRepository repo;

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

                repo.save(new Product(
                                100, "ABC", 500.0,
                                new Date(), new Date(), new Date()
                                )
                                );

        }

}
---------------------------------------------------------
@Lob : Working with Large Objects(Oracle : BLOB and CLOB)
  BLOB : Binary data (Images, Audio, Videos, PDF, docs...)
  CLOB : Lengthy Text data

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

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Lob;

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

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Product {
        @Id
        private Integer pid;
        private String pcode;
        private Double pcost;

        @Lob //byte[]+@Lob = BLOB
        private byte[] img;
        @Lob // char[] + @Lob = CLOB
        private char[] data;
}
--Repo--
(same as before example)

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

import java.io.FileInputStream;

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.repo.ProductRepository;

@Component
public class TestJpaRepoRunner implements CommandLineRunner {

        @Autowired
        private ProductRepository repo;

        @Override
        public void run(String... args) throws Exception {
                FileInputStream fis = new
FileInputStream("F:/Images/SpringBoot6PM_03112020.png");
                byte[] img = new byte[fis.available()];
                fis.read(img);

                String dataStr = "HEllo abcdefgh!HEllo abcdefgh!HEllo
abcdefgh!HEllo abcdefgh!HEllo abcdefgh!HEllo abcdefgh!HEllo
abcdefgh!HEllo abcdefgh!HEllo abcdefgh!HEllo abcdefgh!HEllo
abcdefgh!HEllo abcdefgh!";
                char[] data=dataStr.toCharArray();

                repo.save(new Product(
                                100, "ABC", 500.0,
                                img,data
                                )
                                );
                fis.close();
        }

}
```

===

Collections using JPA

*) Primitives/Wrappers (Including String) one variable that stores
   one value. To store multiple values we use Collection variable.

```
Ex: studentId -- Integer
    studentMarks--- List<Integer>
    studentSubjst-- Set<String>
    studentLabs  -- Map<String,String>
```

*) Every Collection Variable in Model class is mapped with one Database table(Child table).

```
        ------------------------------------------------
                      Types of Collection(2)
                         (in Data JPA)
        ------------------------------------------------
          Index based              Non-Index based
         Collection                    Collection

           List,Map                      Set
        ------------------------------------------------
```

=> List gets index number starts from zero by default when we add data.
   Map key behaves like index (it is unique). Set never contains such format.

=> List and Map are index based collections. So, for these variables child table is created using 3 columns : Key Column (Join Key Col/ FK Column), Element(Data) Column, Index (position) column.

  Where as Set comes under non-index based. So, only 2 columns are provided for child table-> Key Column, Element Column.

=> For every collection variable we must apply : @ElementCollection
   Optional Annotation: @CollectionTable, @Column, @OrderColumn, @MapKeyColumn.

=> @ElementCollection says one collection variable--> one child table.

--code---
1. Model class

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

import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.persistence.CollectionTable;
import javax.persistence.Column;
import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.MapKeyColumn;
import javax.persistence.OrderColumn;
import javax.persistence.Table;

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

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="std_tab")
public class Student {
        @Id
        @Column(name="sid")
        private Integer stdId;
        @Column(name="sname")
        private String stdName;
        @Column(name="sfee")
        private Double stdFee;

        @ElementCollection
        @CollectionTable(name="std_sub_tab", //tablename
                joinColumns = @JoinColumn(name="sid") //Key Column
        )
        @Column(name="subj") //element column
        private Set<String> subjects;


        @ElementCollection
        @CollectionTable(
                        name="std_marks_tab", //table name
                        joinColumns = @JoinColumn(name="sid") //key
column
                        )
        @Column(name="mrks") //element
        @OrderColumn(name="pos") //index column
        private List<Integer> marks;


        @ElementCollection
        @CollectionTable(
                        name="std_labs_tab", //table name
                        joinColumns = @JoinColumn(name="sid") //key
column
                        )
        @Column(name="grade") //element column
        @MapKeyColumn(name="lname") //Index column
        private Map<String,String> labExam;
}
```

2. Repository

```
package in.nareshit.raghu.repo;

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

import in.nareshit.raghu.model.Student;

public interface StudentRepository
        extends JpaRepository<Student, Integer> {

}
```

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

import java.util.List;
import java.util.Map;
import java.util.Set;

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

import in.nareshit.raghu.model.Student;
import in.nareshit.raghu.repo.StudentRepository;

@Component
public class TestDataRunner implements CommandLineRunner {

        @Autowired
        private StudentRepository repo;

        @Override
        public void run(String... args) throws Exception {
                repo.save(
                                new Student(
                                                10, "SAM",300.0,

Set.of("ENG","MAT","SCI"),
                                                List.of(90,75,80),

Map.of("CD","A+","JAVA","A","CHEM","B+")
                                                )
                                );
                System.out.println("DONE");
        }

}
--------------------------------------
```

*)Note:
DDL : create/alter/drop --- hibernate.hbm2ddl.auto
DML: insert, update, delete -- Repository Interface

Q) What are Codd Normalization Rules?