Date : 20/01/2021
Spring Boot 9AM
Mr. RAGHU

Spring Boot: Data JPA - CrudRepository

1. Project : Spring Data JPA, Lombok, MySQL.
2. Properties (driver,url,un,pwd,show-sql,ddl-auto,dialect)
3. Model class
4. Repository Inteface
5. Runner class (Test)

Oracle DB :

```
<dependency>
        <groupId>com.oracle.database.jdbc</groupId>
        <artifactId>ojdbc8</artifactId>
        <scope>runtime</scope>
</dependency>

<dependency>
    <groupId>com.jslsolucoes</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.2.0.1.0</version>
    <scope>runtime</scope>
</dependency>

spring:
  datasource:
    driver-class-name: oracle.jdbc.driver.OracleDriver
    url: jdbc:oracle:thin:@localhost:1521:ORCL
    username: system
    password: tiger

Oracle DB: SQL: select * from global_name;  => Output ORCL/XE
Cmd Prompt> tnsping ORCL      /  tnsping XE
  ....port = 1521 ...

MySQL:-
show databases;
create database boot9am;
use boot9am;
-------

*) Driver class Name is optional (JDBC-4 API) has provided Auto-
Register
/ Auto Loading of driver class based on URL and JAR provided in
classpath

 > Maven Dependnecies
    > mysql-connector-jar
        > META-INF
           > services
              > java.sql.Driver (open this file)

*) Dialect/Database Platform also auto-detected based on URL by
   Hibernate F/w.
```

*) We must provide URL, username and password.
*) show-sql is optional, default is false. That indicates do not display
   Generated SQL at console.

*) hibernate.ddl-auto : is optional default value is none(validate).
   That indicates 'Programmer Create/alter their tables'. Hibernate Does
   nothing. Recomanded option: 'update'.

*) For Our Repository Interface, one impl class is generated using
   Sun/Oracle Proxy concept.

CrudRepository

T= Model class

a) save(S obj):S   <S extends T>
 This method behaves like either insert or update.
 It supports taking Model class objects and even their sub class objects.

=> This method first executes SELECT query with given ID in object.
   Checks Given ID exist or not? If ID not exist in DB table then
   INSERT , else Update.

--code--
*)Name:SpringBoot2CrudOperationsEx
*)Dep: Spring Data JPA, Lombok, MySQL

1. Model
```
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 Student {
        @Id
        private Integer sid;
        private String sname;
        private Double sfee;
}
```

2. Repo

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

import org.springframework.data.repository.CrudRepository;

import in.nareshit.raghu.model.Student;

public interface StudentRepository
        extends CrudRepository<Student, Integer>
{

}
```

3. yaml

```
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
```

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

@Component
public class StudentTestRunner implements CommandLineRunner {
        @Autowired
        private StudentRepository repo; //=Impl class Object

        @Override
        public void run(String... args) throws Exception {
                /*
                Student s1 = new Student(10,"A",2.2);
                repo.save(s1);

                Student s2 = new Student(10,"B",3.2);
                repo.save(s2);
                */
                repo.save(new Student(101, "A", 2.2));
                repo.save(new Student(102, "B", 3.2));
                repo.save(new Student(103, "C", 4.2));
                repo.save(new Student(104, "D", 5.2));

        }

}
```

++
b) saveAll(Iterable<S> entities)
  This method is used to insert/update multiple rows at a time.

=> Iterable is super type for collections. So, choose any one
Collection.

*)Runner class code:

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

import java.util.Arrays;
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.model.Student;
import in.nareshit.raghu.repo.StudentRepository;

@Component
public class StudentTestRunner implements CommandLineRunner {
	@Autowired
	private StudentRepository repo; //=Impl class Object

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


		repo.saveAll(
				Arrays.asList(
						new Student(105, "A",
2.0),
						new Student(106, "B",
6.2),
						new Student(107, "C",
2.0),
						new Student(108, "D",
2.7)
				)
				);



	}

}
```

*) JDK 9 has provided new way of creating Collections (Factory
methods)
  of() method, also called as ImmutableCollections. Once created
  can not be modified.

*) Optional<T> JDK 1.8/8.
  To avoid NullPointerException, handle null values.

Dynamic Data(UI/DB/File..) may be null value.
So, before processing it, null check must be done.

---

c) findById(ID id):Optional<T>
This method is used to fetch data from DB using PrimaryKey value.
Given id may or many not exist in DB. So, return type is Optional<T>

Runner class Code:

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

import java.util.List;
import java.util.Optional;

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 StudentTestRunner implements CommandLineRunner {
	@Autowired
	private StudentRepository repo; //=Impl class Object

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

		repo.saveAll(
				List.of(
						new Student(105, "A",
2.0),
						new Student(106, "B",
6.2),
						new Student(107, "C",
2.0),
						new Student(108, "D",
2.7)
				)
				);

		Optional<Student> opt = repo.findById(109);
		if(opt.isPresent()) {
			Student s = opt.get();
			System.out.println("Data is " + s);
		} else {
			System.out.println("Data Not Found");
		}

	}

}
```

---

d) existsById(Id):boolean
  If given ID exist in Database table then it returns true
  else it returns false.

Runner class code:

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

import java.util.List;
import java.util.Optional;

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 StudentTestRunner implements CommandLineRunner {
        @Autowired
        private StudentRepository repo; //=Impl class Object

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

                repo.saveAll(
                                List.of(
                                                new Student(105, "A",
2.0),
                                                new Student(106, "B",
6.2),
                                                new Student(107, "C",
2.0),
                                                new Student(108, "D",
2.7)
                                )
                                );

                boolean exist = repo.existsById(108);
                System.out.println(exist);

        }

}
```

Q) What are cursors in Java? Which one is added in JDK 1.8?
A) Enumerator, Iterator, ListIterator, Spliterator(1.8).

---

e) findAll():Iterable<T>
  This method is used to fetch all rows from DB table.
  SQL: SELECT * FROM <TABLE-NAME>;

=> It will fetch data from DB Table using SELECT QUERY into List<T>
   Collection, casted to Iterable, that supports reading data in

multiple ways.

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

import java.util.Iterator;
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.model.Student;
import in.nareshit.raghu.repo.StudentRepository;

@Component
public class StudentTestRunner implements CommandLineRunner {
	@Autowired
	private StudentRepository repo; //=Impl class Object

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

		repo.saveAll(
				List.of(
						new Student(105, "A",
2.0),
						new Student(106, "B",
6.2),
						new Student(107, "C",
2.0),
						new Student(108, "D",
2.7)
				)
				);

		Iterable<Student> itb = repo.findAll();
		System.out.println(itb.getClass().getName());
//ArrayList

		//core concepts for print
		//Java 8 - Method Reference
		itb.forEach(System.out::println);
		System.out.println("--------------------");

		//Java 8 - Lambda Expression
		itb.forEach(s->System.out.println(s));
		System.out.println("--------------------");

		//Iterator
		Iterator<Student> itr = itb.iterator();
		while (itr.hasNext()) {
			Student s = itr.next();
			System.out.println(s);
		}

		System.out.println("--------------------");
		//For Each Loop
```

```
		for(Student s:itb) {
			System.out.println(s);
		}

	}

}
```

---

f) findAllById(Iterable<ID> ids):Iterable<T>
To fetch random selected rows using in operator, use this method.

SQL:
```
  select * from student where sid in (__, __, __, __);
```

Runner class code:
```
package in.nareshit.raghu.runner;

import java.util.Arrays;
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.model.Student;
import in.nareshit.raghu.repo.StudentRepository;

@Component
public class StudentTestRunner implements CommandLineRunner {
	@Autowired
	private StudentRepository repo; //=Impl class Object

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

		repo.saveAll(
				List.of(
						new Student(105, "A", 2.0),
						new Student(106, "B", 6.2),
						new Student(107, "C", 2.0),
						new Student(108, "D", 2.7)
				)
				);

		Iterable<Student> list = repo.findAllById(Arrays.asList(110,105,108,221,365));
		list.forEach(System.out::println);

	}

}
```

---

g) count():long

This method is used to calculate total no.of rows in DB table.
SQL: select count(*) from student;

Runner class code:

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

import java.util.Arrays;
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.model.Student;
import in.nareshit.raghu.repo.StudentRepository;

@Component
public class StudentTestRunner implements CommandLineRunner {
	@Autowired
	private StudentRepository repo; //=Impl class Object

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

		repo.saveAll(
				List.of(
					new Student(105, "A", 2.0),
					new Student(106, "B", 6.2),
					new Student(107, "C", 2.0),
					new Student(108, "D", 2.7)
				)
				);

		long c = repo.count();
		System.out.println(c);

	}

}
```

Core Java :
a. Generics <S extends T>
b. Optional<T> JDK 8
c. List/Set/Map of() collection methods JDK 9
d. SQL Concepts : count concept, in operator.

```
class Sample<T> { }
```

For Multiple Databases:
https://www.youtube.com/watch?v=nzszxQbQ5WU

Java 8
https://www.youtube.com/watch?v=FYAqqH9oyUo
Java 8 Method Ref
https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html