Date : 26/10/2020
Spring Boot 7AM
Mr. RAGHU

Spring Boot : Data JPA - Stored Procedures

*)Stored Procedures:
To execute one/multiple statements/SQLs (including logics) at Database side
by taking INPUT (IN) parameters (optional) and returns OUTPUT (OUT) Parameters(optional).

Programmer has to follow below steps
a. Define Stored Procedure at database
b. call stored procedure from our application.

Q) Why Stored Procedures, as we have already methods in java?
A) Stored Procedures are used to reduce n/w calls for multiple queries
which are executed for a requirement.
In simple it improves application performance by reducing execution time
for a request.

*) Stored Procedures are recomanded only for performance improvement. Do not use
Stored Procedures for every requirement.

*) Using Stored Procedures is not good approch for database migrations.
Bcoz, Syntax for Stored Procedures in every database may be different.
We need to re-test related module in project.

-------Stored Procedures coding steps---
a. Create a Stored Procedure

```
 CREATE OR REPLACE PROCEDURE <name> (<parameters>)
   BEGIN
     ....
   END
```

b. execute/call a Stored Procedure

```
  call <name> (<parameters>);
```

===========================code
steup==
Dependencies : Data Jpa, MySQL/Oracle, Lombok

1. 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
```

2. 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 Employee {
	@Id
	private Integer empId;
	private String empName;
	private String empDesg;
	private String empDept;
	private Double empSal;
}
```

3. Repository 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> {

}
```

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.Employee;
import in.nareshit.raghu.repo.EmployeeRepository;

@Component
public class DataInsertRunner implements CommandLineRunner {
	@Autowired
	private EmployeeRepository repo;

	@Override
	public void run(String... args) throws Exception {
		repo.save(new Employee(101, "SAM", "Lead", "DEV", 86500.0));
		repo.save(new Employee(102, "RAM", "MGR", "DEV", 96000.0));
		repo.save(new Employee(103, "SYED", "ASSOCIATE", "QA", 32500.0));
		repo.save(new Employee(104, "ABD", "MGR", "BA", 55500.0));
	}

}
```

==
===

MySQL Database Stored Procedures:

1. No IN and OUT Params. (SELECT query data is final output)

```
DELIMITER $$
CREATE PROCEDURE GETALLEMPS()
	BEGIN
		SELECT * FROM EMPLOYEE;
	END$$
DELIMITER ;
```

2. Only IN param, No OUT param.

```
DELIMITER $$
CREATE PROCEDURE GETEMPBYDEPT(IN edept VARCHAR(25))
 BEGIN
   SELECT * FROM EMPLOYEE E WHERE  E.EMP_DEPT = edept;
 END $$
```

```
DELIMITER ;
```

3. Using both IN and OUT Parameters

```
DELIMITER $$
CREATE PROCEDURE GETEMPBYDESGCOUNT(IN edesg VARCHAR(25), OUT dcount INT)
 BEGIN
   SELECT COUNT(*) INTO dcount FROM EMPLOYEE E WHERE  E.emp_desg = edesg;
 END $$
DELIMITER ;
```

============CALLING PROCEDURES USING Spring Boot=========================

a) Using @Query annotation

By using @Query we can call Stored Procedures as Native SQL query.
It supports NO Param and IN Param procedure calls only.
For this we need define one abstract method with @Query and nativeQuery = true attribute inside repository interface

--code-------

```
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(value = "CALL GETALLEMPS()",nativeQuery = true)
	List<Employee> getAllEmps();

	@Query(value = "CALL GETEMPBYDEPT(?)",nativeQuery = true)
	List<Employee> getEmpsByDept(String edept);


}
```

=> Test Using 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.repo.EmployeeRepository;

@Component
public class TestProcRunner implements CommandLineRunner {
	@Autowired
	private EmployeeRepository repo;

	@Override
	public void run(String... args) throws Exception {
		//repo.getAllEmps().forEach(System.out::println);
		repo.getEmpsByDept("DEV").forEach(System.out::println);
	}

}
```
=---------------=-----------=------------------=--------------=

b) using JPA 2.x API

Task:

*) Write above Stored Procedures using Oracle Database.