Calling PL/SQL procedure and functions using spring data JPA

====

=> Instead of writing same persistence logic or b.logic in every module of the Project..

It is recomaned to write them as PL/SQL procedure or function (stored procedures or function)

only for 1 time in Db s/w and call them mutiple modules of the Project.

=> In order to avoid important logics authentication and authorization from developers

of the project.. they will be develope as PL/SQL procedures or functions.. only their singture details

will be exposed to the developers.. (To maintain the secrecy on the logics)

Authentication:: Checking the identity of a user Authorization :: Checking the access permissions of a User

usecase1:: Authentication,Authorization logics will be devleoped as PL/SQL procedure or function and will be called from multiple moduels

usecase2:: Attendence calculation logics will be devleoped as PL/SQL procedure or function and will be called from multiple modules

usecase3:: Billing logics,settling claim amount logics, calculating student cgpa, sgpa logics,

some kind of batch processing operations and etc.. logics will be developed

as PL/SQL procedures or functions

A typical java project contains (spring/spring book projects)

60 to 70% persistence logics using JPQL/HQL + native SQL +o-r mapping logics (requlars spring boot data jpa logics) 30% to 40% persistence logics using PL/SQL procedures or functions..

of

Before the arrival of Distributed Technologies and Frameworks industry used to work with PL/SQL procedures and functions to develope distributed logics and to consume those logics from different types of Apps

=>PL/SQL programming is specific to each Db s/w becoz we use SQL queries in PL/SQL programming and the syntax PL/SQL Programming is specific to each DB s/w =>PL/SQL procedure does not return a value..but we can get multiple results/outputs using OUT params

=>PL/SQL function return a value..but we can get multiple results/outputs using return value + OUT params.

5 results from PL/SQL procedure -->take 5 OUT params (best)

5 results from PL/SQL function -->take 4 OUT params and 1 return value.

=> PL/SQL procedure ro function parameters not only contains type... they also maintain mode

the modes are

IN (default)

OUT

INOUT

PL/SQL logic

In oracle PL/SQL programming

In Java Programming, the method params contain

:= is for assignment

only type (data type).. where as the params of PL/SQL Programming contain modes, types and etc..

= is for comparision

z:=x+y;

x,y --> IN parameters

z --> OUT parameter

PL/SQL logic

x:=x*x;

x --> INOUT parameters

=>A Cursor(collection) is InMemory variable of oracle PL/SQL programming that is capable of holding

bunch of records (0 or more records) given by SELECT SQL Query execution.. (It is like ResultSet obj in JDBC Programming) =>SYS_REFCURSOR is pre-defined cursor data type of oracle PL/SQL programming and it can be used like this

details SYS_REFCURSOR; -> cursor variable declaration open details for

SELECT * FROM STUDENT;

The records given by Select SQL Query will be stored into cursor variable (details)

=> oracle PL/SQL programming's cursor is similar to JDBC ResultSet obj.. While recieving outputs from Cursor type OUT parameter.. we can take the support of JDBC ResultSet obj.

Creating PL/SQL procedure in Oracle Db s/w using SQL Developer

=======

===========

========

Jo-r mapping programming we can store

List<T> (List of Entity objects))

expand con ---> expand procedures ---> right click -->create new procedure ---> Schema: SYSTEM

Name:

P_GET_DOCTORS_BY_INCOME_RANGE

Add New Source In Lowercase

Parameters: Q name

Name

=> The PL/SQL procedure and the PL/SQL function are technically called stored Procedures and functions becoz they reside and execute in DB s/w on Perminent basis.

STARTINCOME

ENDINCOME DETAILS

Mode

No Copy

IN

IN

OUT

LI

Data Type

Defa

FLOAT

FLOAT

SYS_REFCURSOR

ok

↓

CREATE OR REPLACE PROCEDURE P_GET_DOCTORS_BY_INCOME_RANGE

(

STARTINCOME IN FLOAT

, ENDINCOME IN FLOAT

DETAILS OUT SYS_REFCURSOR

) AS

BEGIN

OPEN DETAILS FOR

SELECT * FROM JPA_DOCTOR_INFO WHERE INCOME>=STARTINCOME AND INCOME<=ENDINCOME;

END P_GET_DOCTORS_BY_INCOME_RANGE;

=> The records given by this SELECT

SQL Query will be stored in

"DETAILS" CURSOR Variable automatically

be

In Spring Data JPA we can use EntityManager object (like HB Session obj) to call PL/SQL procedure.. if we add spring data jpa starter this ityM

to Service class.

object will created through AutoConfiguration,So it can be injected

note:: EntityManager object encapsulates the fuctionality of underlying JPA with hibernate framework.

Process to develop spring data jpa application as spring boot App using Gradle

==========

==========

step1) make sure that BuildShip plugin(Gradle pluin) is installed in Eclipse IDE

note:: In new versions of Eclipse, it is built-in plugin. step2) create spring Boot starter project using Gradle as the build tool. File--> new --->other--->spring starter Project --->

note:: Ater creating gradle project, if we modify the name of the Projct then it should reflect in settings.gradle file and we need perform gradle refresh once

Spring Boot Version:

Frequently Used:

Service URL

https://start.spring.io

Name

BootDataJPAProj08-Calling PLSQLPRocedure-Oracle

Use default location Location

G:\Worskpaces\Spring\NTSPBMS714-BOOT\BootDataJPAProj08- Browse

Type:

Gradle

ackaging:

Jar

Java Version: Group

11

✓ Language:

Java

nit

Artifact

Version

Description

BootDataJPAProj08-Calling PLSQLPRocedure-Oracle

0.0.1-SNAPSHOT

SpringDataJPA Application

Package

com.nt

Working sets

next --->

JDBC API

Lombok

Oracle Driver

Spring Data JPA

In maven, the input file name is :: pom.xml (xml cfgs)

In gradle, the input file name is :: build.gradle (dsl -- domain specific language more of groovy language)

=> Maven build tool is used more and more in Java Projects

=> Gradle build tool is used different language Projects.

Boot-DataJPA-Proj7-EntityManager-CallingPL-SQLProcedure [boot]

>Spring Elements

#src/main/java

com.nt

› BootDataJpaProj7EntityManagerCallingPlSqlProcedureApplic

com.nt.entity

> Doctor.java

com.nt.runner

> CallingPLSQLProcedureTest.java

com.nt.service

next ----> finish.

step3) add more dependencies in build.gradle using dependencies {.....} (enclouser)

(u can collect info from mvnrepository.com)

(gradle tab)

step4) the develop code in service Interface and in service Impl class

In service interface

application.properties

>

>

>

DoctorServiceMgmtImpl.java

IDoctorManagementService.java

src/main/resources

src/test/java

>

JRE System Library [JavaSE-17]

>

Project and External Dependencies

>

bin

>

gradle

> src

```
public interface IDoctorManagementService {
public List<Doctor> showDoctorsByIncomeRange(double start,double end);
}
```

Service Impl class

```
/*CREATE OR REPLACE PROCEDURE P_GET_DOCTORS_BY_INCOME_RANGE
(
STARTINCOME IN FLOAT
,ENDINCOME IN FLOAT
DETAILS OUT SYS_REFCURSOR
,
```

) AS

build.gradle

gradlew

gradlew.bat

WHELP.md

settings.gradle

BEGIN

OPEN DETAILS FOR

SELECT * FROM JPA_DOCTOR_INFO WHERE INCOME>=STARTINCOME AND INCOME<=ENDINCOME;

END P_GET_DOCTORS_BY_INCOME_RANGE; */

@Service("doctorService")

public class DoctorServiceMgmtImpl implements IDoctorManagementService {

@Autowired

private EntityManager manager;

@Override

public List<Doctor> showDoctorsByIncome Range(double start, double end) {

//create Stored ProcedureQuery object pointing PL/SQL procedure

StoredProcedureQuery query=manager.createStored ProcedureQuery("P_GET_DOCTORS_BY_INCOME_RANGE",Doctor.class);

// register both IN,OUT params by specifying their mode

query.registerStored ProcedureParameter(1, Double.class, ParameterMode./N);

query.registerStored ProcedureParameter(2, Double.class, ParameterMode./N); query.registerStored ProcedureParameter(3, 1Object.class, ParameterMode.REF_CURSOR); //set values to IN params

query.setParameter(1,start);

query.setParameter(2, end);

//call PL/SQL procedure

List<Doctor> list-query.getResultList(); return list;

registers the given 3 rd param

as out cum REF Cursor Param

}

}

step6) write the following code in runner class

@Component

public class CallingPLSQLProcedureTest implements CommandLineRunner {

@Autowired

private IDoctorManagementService service;

application.properties

#jdbc properties (for oracle)

spring.datasource.driver-class-name-oracle.jdbc.driver.OracleDriver

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe

spring.datasource.username=system

spring.datasource.password=manager

spring.datasource.hikari.maximum-pool-size=100

spring.datasource.hikari.minimum-idle=10

spring.datasource.hikari.keepalive-time=100000

@Override

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

//invoke the b.method of service

service.showDoctorsByIncome Range(10000.0, 2000000.0).forEach(System.out::println);

spring.jpa.datasource-platform=org.hibernate.dialect.Oracle 10gDialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto-update

spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true

}

}

Package javax.persistence

Interface EntityManager

`public interface EntityManager`

nataraz sir Senior Java Consulta

Interface used to interact with the persistence context.

An EntityManager instance is associated with a persistence context. A persistence context is a set of entity instances in which for any persistent entity identity there is a unique entity instance. Within the persistence context, the entity instances and their lifecycle are managed. The EntityManager API is used to create and remove persistent entity instances, to find entities by their primary key, and to query over entities.

The set of entities that can be managed by a given EntityManager instance is defined by a persistence unit. A persistence unit defines the set of all classes that are related or grouped by the application, and which must be colocated in their mapping to a single database. (EntityManager object represents underlying ORM s/w)

Assignment:: call PL/SQL procedure or function of MySQL DB s/w from our spring data jpa application

note:: The Syntax of PL/SQL Programming is different in different DB s/ws.

=> Each Custom Repository in spring data jpa programing is specific to one Entity class, Since PL/SQL Procedure or function

is not specific to any one db table, can contain multiple db tables logics,So we can use EntityManager object support

to call PL/SQL procedure becoz the EntityMAmager represents underlying ORM f/w and common for multiple entities

Calling PL/SQL Procedure of MySQL DB s/w from spring data JPA application

==========

==========

=>In MySQL there is no support for Cursors. Acturally Cursors are not required

in MySQL PL/SQL programming.

=> use jpa_doctor_tab db table which already created in mysql Db s/w

=>To create PL/SQL procedure in mysql

launch MySQL workbench --> select con ---> expand ntspbms616db----> right click procedure --->

create new Procedure ---> type procedure code --->

```
CREATE PROCEDURE `GET_DOCTORS_BY_INCOME_RANGE` (in start float, in end float)
BEGIN
SELECT * FROM JPA_DOCTOR_TAB WHERE INCOME>=start AND INCOME<=end;
END
```

apply --->next--->

```
USE `ntspbms516db`;
DROP procedure IF EXISTS `GET_DOCTORS_BY_INCOME_RANGE`;
DELIMITER $$
USE `ntspbms516db`$$
CREATE PROCEDURE `GET_DOCTORS_BY_INCOME_RANGE` (in start float, in end float)
BEGIN
SELECT * FROM JPA_DOCTOR_TAB WHERE INCOME>=start AND INCOME<=end;
END$$
DELIMITER;
```

==> apply ===> finish

To get bunch of records given by

SELECT SQL query there is no need

of taking any kind of Cursor in PL/SQL

programming of mysql..

=> select query returned bunch of records automaticaly goes

to ResultSet from the PL/SQL procedure of

mysql i.e there is no need of storing in any kind of cursor.

note:: in MYSQL PL/SQL programming there are no cursors

DEvelope spring Data JPA Application as shown below using Entity Manager support

application.properties

```
#DataSource cfg
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql:///ntspbms714db
spring.datasource.username=root
spring.datasource.password=root
```

In build.gradle as mysql connctor/j dependency as addtional

dependnecy

// https://mvnrepository.com/artifact/mysql/mysql-connector-java implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.26'

#JPA-Hiberante properties

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

other possible values create, validate,create-drop

Emp.java (Entity class)

==========

//Employee.java

package com.nt.entity;

import jakarta.persistence.Column;

import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Employee_Info {

@GeneratedValue(strategy = GenerationType.AUTO)

@Id

private Integer eno;

private String ename;

private String desg;

private Double salary; private Double gross_salary;

private Double net_salary;

//setters && getters

public Integer getEno() { return eno;

Boot-DataJPA-Proj8-EntityManager-Mysql CallingPL-SQLProcedure2 [t

> Spring Elements

#src/main/java ✓com.nt

› BootDataJpaProj8EntityManagerCallingPlSqlProcedureApplic

com.nt.entity

> Employee_Info.java

com.nt.runner

> CallingPLSQLProcedure Test.java

com.nt.service

> EmployeeMgmtServiceImpl.java

> EmployeeMgmtService.java

src/main/resources

application.properties

>src/test/java

> JRE System Library [JavaSE-17]

>

Project and External Dependencies

}

> bin

> gradle

public void setEno(Integer eno) {

› src

this.eno eno;

build.gradle

gradlew

}

gradlew.bat

HELP.md

public String getEname() {

settings.gradle

return ename;

}

public void setEname(String ename) {

}

this.ename = ename;

public String getDesg() {

}

return desg;

public void setDesg(String desg) {

}

this.desg = desg;

public Double getSalary() {

}

return salary;

public void setSalary(Double salary) {

}

```
this.salary = salary;
public Double getGross_salary() {
}
return gross_salary;
public void setGross_salary(Double gross_salary) {
this.gross_salary = gross_salary;
}
//toString()
@Override
public String toString() {
}
}
return "Employee_Info [eno=" + eno + ", ename=" + ename + ", desg=" + desg + ", salary=" + salary
+ ", gross_salary=" + gross_salary + ", net_salary=" + net_salary + "]";
```

In the Entity class development, if the class name is matching with db table name

and property names are matching with db table col names then using @Table and

@Column annotations is optional in the Entity class.

How to make Lombok api working with Eclipse IDE if it is failed work even afte configuration by launching lombok-api-<ver>.jar file?

Ans) add the following plugin eclipse IDE (https://projectlombok.org/p2)

help menu --->install new software ---> add --> name :: lombok

url:: https://projectlombok.org/p2 ---> next ---> next ---> .... ---> restart IDE ..

Service inteface

==========

```
public interface IEmployeeMgmtService {
public List<Employee_Info> showEmployeeBySalary Range(double start,double end);
}
```

service Impl class

==========

```
/*CREATE DEFINER=`root`@`localhost` PROCEDURE `p_emp_details_by_salaryRange`(IN startSalary float,
IN endSalary float) BEGIN
SELECT * from employee_info where salary>=startSalary and salary<=endSalary;
END
*/
@Service("empService")
@Autowired
public class EmployeeMgmtServiceImpl implements IEmployeeMgmtService {
private EntityManager manager;
```

Procedure to call PL/SQL procedure of MySQL DB s/w that performs Authentication Activities

==========

step1) create table in MySQL Logical DB having the registered usernames and passwords Expand NTSPBMS 1102---> right click on tables ---> new table --->

P_GET_ARTISTS_BY_FEE - R... login_info - Table x

Column Name

username

→ password

Table Name: login_info

Charset/Collation: Default Charset

Comments:

Schema: ntspbms

Default Collation

Engine:

InnoDB

Datatype

PK

VARCHAR(30)

VARCHAR(30)

000

2000

3000

000

NN UQ B

UN ZF

000

000

--> apply --next --->

CREATE TABLE `ntspbms1102db`.`login_info` (`username` VARCHAR(30) NOT NULL,

`password` VARCHAR(30) NOT NULL,

PRIMARY KEY (`username`),

UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE);

--> finish

Result Grid

→Filter Rows:

username

password

raja

rani

ramesh

hyd

suresh

vizag

NULL

NULL

G

000

step2) Create PL/SQL procedure in MySQL DB s/w having authentication logic

Expand NTSPBMS1102DB ---> right on Procedures --->

....

```
CREATE PROCEDURE `p_authentication` (in uname varchar(20),
in pwd varchar(20),
out result varchar(20))
BEGIN
@Override
query.registerStored Procedure Parameter(1,Double.class, ParameterMode./N);
query.registerStored Procedure Parameter(2, Double.class, ParameterMode./N);
public List<Employee_Info> showEmployee BySalary Range(double start, double end) {
// Create Stored ProcedureQuery object
StoredProcedureQuery query=manager.createStored ProcedureQuery("p_emp_details_by_salaryRange",
Employee_Info.class); //register the parameters of the Stored Procedure
//set parameteer values
query.setParameter(1, start);
query.setParameter(2, end);
//call PL/SQL procedure
List<Employee_Info> list-query.getResultList();
return list;
}
}
```

Runner class

==========

```
@Component
public class CallingPLSQLProcedureTest implements CommandLineRunner {
@Autowired
```

```
private IEmployee MgmtService service;
@Override
public void run(String... args) throws Exception {
//invoke the b.method of service
service.showEmployee BySalaryRange(10000.0, 200000.0).forEach(System.out::println);
}
}
declare cnt int(2); //local variable
select count(*) into cnt from login_info where username=uname and password=pwd;
if(cnt<>0) then
set result="Valid credentials";
else
set result="InValid Credentials";
end if;
END
USE `ntspbms1102db`;
DROP procedure IF EXISTS `p_authentication`;
DELIMITER $$
USE `ntspbms1102db`$$
CREATE PROCEDURE `p_authentication` (in uname varchar(20),
in pwd varchar(20),
out result varchar(20))
BEGIN
declare cnt int(2);
select count(*) into cnt from login_info where username=uname and password=pwd;
if(cnt<>0) then
set result="Valid credentials";
else
set result="InValid Credentials";
end if;
END$$
DELIMITER;
```

step3) Develop the Spring boot data jpa application using Gradle build tool

file menu ----> spring boot starter project ----> next -->

Service URL

Name

https://start.spring.io

BootJpaProj11-Calling Procedure-Authentication-MYSQL

✓ Use default location

Location

E:\Worskpaces\Spring\NTSPBMS1102\BootJpaProj11-Calling Procedure-A Browse

Type:

Gradle - Kotlin

Java Version:

17

Packaging:

Jar

Language:

Java

Group

com.nit

Artifact

BootJpaProj11-Calling Procedure-Authentication-MYSQL

Version

0.0.1-SNAPSHOT

Description

Demo project for Spring Boot

Package

com.nt

Working sets

Add project to working sets

Working sets:

step3) develop the source code

BootJpaProj11-Calling Procedure-Authentication-MYSQL [boot]

src/main/java

*

#com.nt

W

BootJpaProj11 Calling Procedure Authentication MysqlApplication.j

com.nt.runners

› CallingPL_SQLProcedureTest.java

com.nt.service

> ILoginMgmtService.java

> LoginMgmtServiceImpl.java

src/main/resources

application.properties

> src/test/java

>

JRE System Library [JavaSE-17]

> Project and External Dependencies

> bin

>

gradle

>

src

build.gradle.kts

gradlew

gradlew.bat

HELP.md

settings.gradle.kts

service interface

New...

Select...

=>build.gradle is gradle configuatin file

that allows groovy based DSL =>build.gradle.kts is gradle configuatin file that allows kotlin based DSL

application.properties

spring.application.name=BootJpaProj07

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql:///ntspbms1102db

spring.datasource.username=root

spring.datasource.password=root

 # JPA - hibernate properites

#spring.jpa.database-platform-org.hibernate.dialect.Oracle Dialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto-update

 # For enabling lazy loading

spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true

public interface I LoginMgmtService {

public String doLogin(String user, String pwd);

}

Service Impl class

```
@Service("loginService")
public class LoginMgmtServiceImpl implements ILoginMgmtService {
@Autowired
private EntityManager manager;
/*CREATE DEFINER=`root`@`localhost` PROCEDURE `p_authentication`(in uname varchar(20),
Runner class
=========
@Component
in pwd varchar(20),
out result varchar(20))
BEGIN
declare cnt int(2);
select count(*) into cnt from login_info where username=uname and password=pwd;
if(cnt<>0) then
set result="Valid credentials";
else
set result="InValid Credentials";
end if;
END
*/
@Override
public String doLogin(String user, String pwd) {
}
// create SToredPRocedureQuery object
Stored ProcedureQuery query=manager.createStored ProcedureQuery("p_authentication");
//register the params
query.registerStored ProcedureParameter(1,String.class, ParameterMode./N);
query.registerStored Procedure Parameter(2,String.class, ParameterMode./N);
query.registerStored Procedure Parameter(3,String.class, ParameterMode.OUT);
//set Param values
query.setParameter(1, user);
query.setParameter(2, pwd);
//Call the PL/SQL procedure
String result=(String) query.getOutputParameterValue(3);
return result;
public class CallingPL SQLProcedure Test implements CommandLineRunner {
@Autowired
```

```
private ILoginMgmtService loginService;
@Override
public void run(String... args) throws Exception {
try {
//invoke the method
}//try
String result=loginService.doLogin("raja", "rani");
System.out.println(result);
catch(Exception e) {
e.printStackTrace();
}
}
}
```