Making the MiniProject interacting with MySQL DB s/w

========

step1) make sure that MySQL 8.x Db s/w in installled

of

step2) use GUI Db tool MySQL work bench (part of MySQL DB s/w) to perform the following operations

a) create a connection with MySQL DB s/w

use (+) Symbol for creating the connetion --->

Connection Name: con1

Standard (TCP/IP)

Connection Method:

Parameters SSL

Advanced

Hostname:

127.0.0.1

Port: 3306

=>if multiple projects a company are using the centrally installed physical DB s/w like oracle, MySQL and etc...then for each project one Logical DB will be created on 1 per Project Basis..In that Logical DB, we place project related db tables,sequences, PL/SQL procedures ,functions and etc..

=> Logical DB is the Logical Partiation of the physical DB s/w which will be created on 1 per project basis i.e each project related db tables, sequences and et will be placed in a Logical DB

Oracle (physical DB s/w)

Logical DB 2 for Proj2 sid:: p2 Logical DB3

Logical DB1 for Proj1

TEst connection

ok

sid:: p1

for Proj3 sid:: p3

DB Engine

mysql (physical DB s/w)

Logical DB2

for Proj2

Logical DB1 for Proj1

db name: s1,

DB Engine

db name: s2 Logical DB3 for Proj3 db name: s3

Username:

root

Password:

root Store in Vault...

Clear

Default Schema:

b) create Logical DB having db table "emp"

NTSPBMS618DB

launch con1 in MySQL workbench ---> create Logial DB --->

--->name ::

--> apply--> apply

NTSPBMS618DB

expand

---> tables ---> right click ---> create new table

Table Name: emp

Schema: ntspbms1102db

Charset/Collation: Default Charset

Default Collation

Engine:

InnoDB

Column Name

empno

◇ ename

◇ sal

◇ job

◇ deptno

Comments:

Datatype

PK

NN UQ

INT

VARCHAR(45) DOUBLE VARCHAR(45) INT(10)

000000

000000

UN

ZF

000000

3000000

B

9000000

000000

---> apply ok

AI G Default

4 000000

000000

PK --> Primary Key NN --> Not Null Key

UQ ---> Unique Key

AI ---> Autoincrement key

(uses previous val +1 formulae to insert the value to db table col having start with 1) [if we insert values to other cols, the value to PK col will come automatically]

we

note:: In MySQL DB s/w we do not have support for sequences, but we can bring that effect using AI (Auto Increment) constraint applied on the PK column .. AI constraint means if insert values to other colums of the row the value to PK column comes automatically starting with 1 increment by 1

intial value :1 increment by 1 (previois val +1)

step3) take the copy of MiniProject in Eclipse IDE

step4) add MySQL jdbc driver starter to the Project

note:: Oracle supports sequences, So we can use them in the INSERT SQL Query to insert the values to PK col dynamically

MySQL do not support sequences, but we can bring that effect

by taking PK column as the AI (AutoIncrement) column

Right click on the Project ---> spring ---> add starters ---> slect MySQL Driver ---> select pom.xml file --> apply -->finish

step5) place MySQL jdbc driver details in application.properties file

In application.properites

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

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

spring.datasource.password=root

step6) Develop separate DAO Impl class for MySQL DB s/w having SQL Queries accordingly

```
package com.nt.dao;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;
```

```
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.stereotype.Repository;

import com.nt.model.Employee;

@Repository("empMySQLDAO")

public class Employee MySQLDAOImpl implements IEmployee DAO {

private static final String GET_EMPS_BY_DESGS="SELECT EMPNO,ENAME, JOB,SAL, DEPTNO FROM EMP
WHERE JOB IN(?,?,?)"; private static final String INSERT_EMPLOYEE="INSERT INTO EMP(ENAME,SAL,
JOB, DEPTNO) VALUES(?,?,?,?)";

@Autowired

private DataSource ds;

@Override

--

Differrent from Oracle DB INSERT SQL Query

public List<Employee> getEmpsByDesgs(String desg1, String desg2, String desg3) throws Exception {

List<Employee> list=null;

try(//get Pooled connection

Connection con=ds.getConnection();

//create PreparedStatement obj having the pre-compiled SQL Query PreparedStatement
ps=con.prepareStatement(GET_EMPS_BY_DESGS);

X){

//set values to query params

ps.setString(1, desg1); ps.setString(2, desg2);ps.setString(3, desg3);

try(//execute the Query

ResultSet rs=ps.executeQuery();){

//process the Result Object

list=new ArrayList();

while(rs.next()) {

//copy each record into Java bean class object

Employee emp=new Employee();

emp.setEmpno(rs.getInt(1));

emp.setEname (rs.getString(2));

emp.setJob(rs.getString(3));

emp.setSalary(rs.getDouble(4));

emp.setDeptno(rs.getInt(5));

//add each JAva bean class object List Collection

list.add(emp);

}//while
```

```
}//fry2
}//try1
catch(SQLException se) {
}
System.out.println("some DB Problems, wait for sumTime");
throw se;
catch (Exception e) {
System.out.println("some up known Problems, wait for sumTime");
throw e;
}
return list;
}
@Override
public int insertEmployee (Employee emp) throws Exception {
int result=0;
try(//get Pooled connection
Connection con=ds.getConnection();
//create PreparedStatement obj having the pre-compiled SQL Query PreparedStatement
ps=con.prepareStatement(INSERT_EMPLOYEE);
X{
//set values to Query params
ps.setString(1, emp.getEname());
ps.setDouble(2, emp.getSalary());
ps.setString(3,emp.getJob());
ps.setInt(4, emp.getDeptno());
//exectue the SQL Query
}
result=ps.executeUpdate();
catch(SQLException se) {
}
se.printStackTrace();
throw se;
catch (Exception e) {
e.printStackTrace(); throw e;
}
return result; }//method
}//class
```

step7) Solve the ambiguity Problem towards injecting DAO class obj to Service class object using the

@Qualifier(-) annotation

In service Impl class

@Service("empService")

public class EmployeeMgmtServiceImpl implements IEmployeeMgmtService { @Autowired

@Qualifier("empMySQLDAO")

private EmployeeDAO empDAO;

step8) Run the Client app

note:: In spring boot Layered App, we can not take certain spring beans like DAO classes as final classes in which we inject AutoConfiguration generated Spring beans like DataSoruce object as IOC container tries to generate InMemory Proxy class as the sub class of that spring bean class and final classes can not have sub classes

=> The GUI DB tool for oracle is "SQL Developer" (separate installation) not part of oracle DB s/w installation

=> The GUI DB tool for mysql is MySQL workbench" (separate installation is not required) part of mysql DB s/w installation

=> The GUI DB tool for PostgreSQL is "PgAdmin 4" (separate installation is not required) part of PostgreSQL DB s/w installation oracle.com/tools/downloads/sqldev-install-windows.html

()-

ORACLE

Products Industries Resources Customers Partners Developers Co

SQL Developer supports JDK 7 and up.

To install and run:

• Ensure you have a JDK installed, if not, download here

• Download SQL Developer and extract sqldeveloper.zip into a new folder

• Within that folder, open the sqldeveloper folder

• Double-click sqldeveloper.exe

note:: application.properties file is part of Spring Project Eco System i.e it will be recognized automatically with out having any additional configurations through out project (No need of using @PropertySource and <context:property-placeholder> tag)

How to make the above mini Project as 100% Loosely coupled Project while changing from one DB s/w to another DB s/w? application.properties file + spring bean cfgs file (xml file) + <alias> tag for bean aliasing

Ans) For this we need to use

step1) In application.properites provide ur choice DAOImpl class bean id by taking key of ur choice

in application.properties

spring.application.name=BootIOCProj03-LayeredApp-MiniProject

#Oracle DataSource configuration

#spring.datasource.driver-class-name=oracle.jdbc.driver.Oracle Driver

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

#spring.datasource.username=system

#spring.datasource.password=tiger

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

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

spring.datasource.username=root

spring.datasource.password=root

#DAO class bean id

dao.id=empMySQLDAO

step2) develop the spring bean cfg file (applicationContext.xml) providing alias name for the DAO class bean id collected from the properties file

in applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context" xsi:schemaLocation="

http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

"$(dgo.id}"

<!-- provide alias name for DAO class bean id collected from the properties file --> <alias name="${dao.id}" alias="dbDAO"/>

</beans>

DAO class Bean id gathered from the properties file

Collect the schema import statements from https://docs.spring.io/spring-

alias name (fixed)

framework/docs/4.2.x/spring-framework- reference/html/xsd-configuration.html use chat GPT

(or)

step3) configure this applicationContext.xml file in the main class cum configuration class

@SpringBootApplication

@ImportResource("com/nt/cfgs/applicationContext.xml")

public class BootlocProj03 LayeredAppMiniProjectApplication {

}

step4)

place the alias name in @Qualifier(-) annotation of service class

@Service("empService")

public class Employee MgmtServiceImpl implements IEmployeeMgmtService {

@Autowired

@Qualifier("dbDAO")

private EmployeeDAO empDAO;

step5) run the application by changing the DAO class bean id

in the application.properties file (also chage the JDBC driver details)

ChatGPT

sample spring bean configuration file

O o

Q) When we add the starters like spring-boot-starter-JDBC/JPA/... what is default DataSource obj that comes pointing to JDBC con pool?

Ans) we will get HikariDataSource object

Q) Can we configure DAO class with @Service annotation and Service class with @Repository Annotation?

e

Q)

Ans) yes, we can do, but not recommended to use.. becoz of the following reasons

a) This will kill the readability of the Layered App classes like DAO class looks like Service class becoz of @Service and Service class looks like DAO class becoz of @Repository

b) These special stereo type annotation like @Service and @Repository will give additional behaviours like

@Repository perform underlying Persistence techno Exception translation to sprin specific exception

(SQLException

@Service perform Transaction management (Executing the logics by applying do every thing or nothing) note: It is recommended to use right annotation in right place

For DAO class ----> @Repository

For Service class ------> @Service

For Controller class ------> @Controller

For Normal class as Spring Bean ----> @Component

Can i use only @Component annotation to make all the java classes as the spring beans in the Layered App (Mini Project)?

Ans) yes, we can use .. but not recommended to use becoz of the following reasons

a) we will loose the redability of the code i.e identifying which class is service class, which class is DAO class becomes complex

b) we can not take benefit @Contorller, @Service and @Repository annotation additional behavior

i) @Service Transaction Management support

gives

gives

ii) @Repositori Exception Translation support

iii) @Controller Ability to take the requests

How can make spring /spring boot Layered App as 100% Loosely coupled App towards choosing our choice DAO class for our choice DB and towards choosing our Choice Service class ?

Ans1) using properties file/yml file + spring bean cfg file(xml) + bean aliasing using <alias> tag + @Qualifer

(not a best solution)

(or)

Ans2) Using Spring/Spring boot profiles (Best)