Transaction Management (SEcondary logic)

The process of combining related operations into single unit and executing them by applying do everything or nothing principle is called Transaction Management..

usecase1:: executing withdraw , deposite operations of transfer money task by applying do every thing or nothing priciple.
usecase2:: Employee registration in the company should insert record in hr db table and also in finance db table having do every thing or nothing priciple
usecase3:: In E-Commece applications.. updating inventory, payment operations should be executed having do every thing or nothing priciple

note:: All senstive logics in Application development must be executed by enabling TRansaction Management...

3 important operations of Tx Mgmt

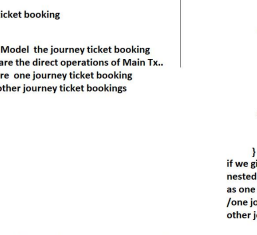
Tx ---> Transcation
mgmt ->Managmemt

```
try{
//begin Transaction (1)
  con.setAutoCommit(false); --> example in jdbc
 //execute logics (Trsnactional logics) (2)
    ->logic for with draw operation (operation1)
      int count1=st.executeUpdate(" query removing amt from balance");
    ->logic for deposite operation (operation2)
      int count2=st.executeUpdate("query for adding amt to balance");

      if(count1==1 && count2==1)
         flag=true;
       else
         flag=false;
  }//try
  catch(Exception e){
     flag=false;
   }
   finally{
     try{
       if(flag)
          con.commit();// commit the Tx (3)
       else
          con.rollback() // rollback the Tx (3)
       }
       catch(SQLException se){
          se.printStackTrace();
       }
    }
```

a) Begin Transaction
b) Execute Transaction logics
c) commit or rollack Transaction

Service class == b.logic + Trasnaction Mgmt ..

Transaction Management Architecture



note: The TxManager takes commit or rollback instruction from Transactional App and contacts Resource through Resource manager and either commits all results Transactional logics or rollbacks all the result..

=>files can not be taken as Transaction resources .. becoz there is not provision to commit or rollback data..

We have two types of transactions based on no.of resources that involved in a Transaction Boundary

a) Local Transactions
   --> All the logics of Transactional code/App will be execute in a single resource
   eg:: transfer money operation b/w two accounts of same bank.
b) Global Transactions/Distributed Transactions/ XA transactions
   ---> All the logics of Transactional code/App will be happening on multiple resources of a Transaction boundary
   eg:: transfer money operation b/w two accounts of two different banks.

note:: JDBC, Hibernate, Spring , EJB and etc.. supports Local Tx Mgmt
Using JDBC, hibernate we can bring effect of Global Tx Mmt, but that is not a good pratice..do

```
In jdbc
======
try{
  ....  //queries execution...
  if(count1==0 || count2==0)
       flag=false;
   else
       flag=true;
}
catch(SQLException se){
   flag=false
 }
catch(Exception e){
   flag=false;
}
finally{
   if(flag){
       con1.commit();
       con2.commit();
     }
    else{
       con1.rollback();
       con2.rollback();
     }
}
```

we need to write similar code in hiberatne....

```
List <String> list=new ArrayList();
```

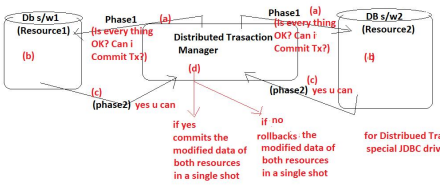
Exception is raised..

Exception is raised..

conclusion:: Spring ,EJB are good for Distributed Tx mgmt...

In Spring , EJB Distributed Tx Mgmt runs on 2pc principle

2pc principle :: 2 phase Commit Priciple

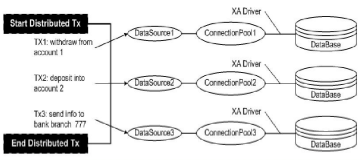


if yes commits the modified data of both resources in a single shot

if no rollbacks the modified data of both resources in a single shot

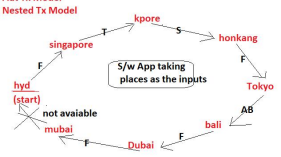
for Distribued TransactionMgmt we need special JDBC drivers like XA drivers..

with respect to oracle docs



Different types of Transactions based on mode of Transaction managment

(a) Flat Tx Model
(b) Nested Tx Model



priorities of ticket booking

1) Flight
2) Train
3) Bus
4) Taxi
5) ship
6) Air Baloon

Flat Tx Model

```
Main Tx{
   jounery1 ticket booking
   jounery2 ticket booking
   jounery3 ticket booking
    ...
    ....
    ....
  journeyn ticket booking
  }
```

=>In Flat Tx Model the journey ticket booking operations are the direct operations of Main Tx.. So the failure one journey ticket booking will effect other journey ticket bookings

Nested Tx Mgmt model

```
Main Tx{
   sub Tx1{
      journey1 ticket booking
    }
   sub Tx2{
      journey2 ticket booking
    }
   sub Tx3{
      journey3 ticket booking
    }
    ...
    ...
    ...
   sub Txn{
        journeyn ticket booking
      }
}
```

if we given tour places to a s/w App that is running with nested tx model ,then each journey ticket booking will be taken as one sub tx of main tx.. So the success or failure of one subTx /one journey ticket booking does not effect other sub Txs and other journey ticket bookings..

=>JDBC, Spring, Hiberate, EJB and etc.. supports Flat Txs..
=> Only Spring supports nested Tx..

Need of Spring Transaction Management

=>For Local Transaction managment we prefre jdbc or hibernate

JDBC

```
try{
  con.setAutoCommit(flase); //begin Tx
   //execute logics/queries
   --------
   .........
     flag=true;
   }
  catch(Exception e){
      flag=false;
  }
  finally{
    try{
     if(flag)
        con.commit();
     else
       con.rollback();
     }
     catch(Exception e){
       e.pristackTrace();
     }
  }
```

on single DB

Hibernate

```
try{
  tx=ses.beginTransaction();
    //execute logics /queries on single DB
       ....
       ....
      flag=true;
  }
 catch(HibernateException he){
      flag=false;
  }
  finally{
    if(flag)
       tx.commit();
    else
       tx.rollback();
  }
```

=>Both JDBC and Hiberante supports Programatic Tx mgmt.. that means we should write Txmgmt code along with App logics.. which is against AOP programming..i.e JDBC,Hibernate do not support delcarative mgmt which says seperate Txmgmt code from App logics ..

=> For Distributed mgmt we need Distributed Tx Manager supplied App server (like weblogic) or third party apis (like Atomokis, narayana and etc..)..

In weblogic Jndi registry DistributedTransactionManager /service is placed having the jndi name "javax.transaction.UserTransaction" (fixed)

=>Developers use JTA (part of JEE) to perform Programatic Distributed Transaction Management

Sample code Using JTA

```
InitialContext ic=new InitialContext();
UserTrasaction ut=ic.lookup("javax.transaction.UserTransaction");
try{
 ut.begin(); //begin the Tx
  //execute logics /queries belonging different DB s/ws..
    .... withdraw query on DB1
    .... deposite query on DB2
     flag=true;
  }
  catch(Exception e){
     flag=false;
  }
  finally{
    if(flag)
       ut.commit();
    else
       ut.rollback();
  }
```

This Programmatic approach code is against of AOP style Programming..

=>Developers have used EJB for declarative Distributed Tx mgmt (Seperating Tx mgmt code from applications logics through xml files.)

Sample code

```
public class BankComp implemets javax.ejb.SessionBean{

 public String transferMoney(long srcAcno, long destAcno,float amount){
    //write withdraw logic /query on DB1
       ....
    //write deposite logic /query on DB2
       ....
    return ".....";
  }
}
```

Here we are placing only Primary logics/B.logics

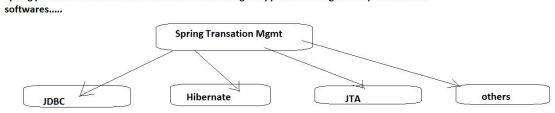
It is supporting AOP style programming..

ejb-jar.xml

EJB :: Enterprise Java Bean

```
--> cfg class name
--> cfg method name
-->enable Trasnactionmgmt by giving the jndi name
      of Distributed Tx Service
```

Spring provides unified model to work different Tx mgmt approches using diff implementation softwares.....



Since springframework provides abstraction layer on multiple techologies and frameworks... So it can generate diff transacton logics in diff approaches based on the underlying framework or technology that we have choosen..

Diff ways of writing spring Unified Transaction mgmt code

```
(a) Using Programatic Approach (against of AOP)
(b) Using spring AOP decl Approach (xml (old))
(c) Using spring AOP Annoations Approach
(d) Using spring AOP 100%Code Approach
(e) Using spring AOP spring boot Approach
```

outdated becoz spring AOP is outdated

```
(f) USing AspectjAOP xml driven approach (decl)
(g) USing AspectjAOP annotation approach
(h) USing AspectjAOP 100% code approach
(i) USing AspectjAOP spring boot approach
```

best practices

(f),(g) :: in Maintence projects
(i) :: in latest projects..

In Spring Tx Mgmt knows wheather it has to genrate Local Tx mgmt code or Global Tx mgmt code by using which technology will be decided based on the TRansactionManager we choose..

DataSourceTransactionManager :: For Local Tx using JDBC
HibernateTransactionManager :: for Local Tx using Hibernate
ToplinkTrasactionManager :: for Local Tx using toplink
and etc..

JtaTransctionManger :: For Distrbuted Tx using the injected Dstribuited Tx Service and manager.

note:: According to AOP Tx Mgmt is the combination of Around advice and throws advice
start -->begin ->rollback
end->commit

@Transctional ---------> To enable Tx on b.method / serivce calss

@Service ----> cfg java class as spring bean cum service class
(contains b.logic + Trsnaction mgmt)

@Repository ----> cfg java class Spring bean cum DAO class
(Persistence logic)

@Component ----> cfg java class as spring bean with out any specialities..

@Autowired ---> For Dependency injection

@Controller ----> cfg java class as spring bean cum controller class..

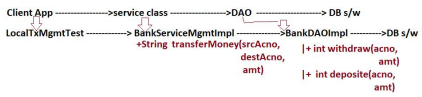
Thum rule in Annotation driven Programming

=> Cfg user-defined classes with streo type annotaitons (@Component,@Service, @Controller,@Repositoru and etc..)
and link them with spring bean cfg file using <context:component-scan > tag

=>cfg pre-defined classes with <bean> tag .. as spring beans in spring bean cfg file .

=>Work with ApplicationContext container...

=> It is always good practice to apply @Transctional on service methods becoz they indirectly apply on DAO class methods becoz service class methods internally calls DAO calls DAO class one or more methods directly or indirectly..



In Decl Tx mgmt our b.method just contains b.logic +@Transactional annotation and we do not begin , commit/rollback Tx.

=> Generally at the start of b.method the Tx begins automatically
and commits tx at end of the b.method automatically if no exception is raised.. if any runtime exception(unchecked exception) is raised ,then the Tx will be rolled back automatically.

@Transctional ---------> To enable Tx on b.method / serivce calss

@Service ----> cfg java class as spring bean cum service class
(contains b.logic + Trsnaction mgmt)

@Repository ----> cfg java class Spring bean cum DAO class
(Persistence logic)

@Component ----> cfg java class as spring bean with out any specialities..

@Autowired ---> For Dependency injection

@Controller ----> cfg java class as spring bean cum controller class..

Thum rule in Annotation driven Programming

=> Cfg user-defined classes with streo type annotaitons (@Component,@Service, @Controller,@Repositoru and etc..)
and link them with spring bean cfg file using <context:component-scan > tag

=>cfg pre-defined classes with <bean> tag .. as spring beans in spring bean cfg file .

=>Work with ApplicationContext container...

=> It is always good practice to apply @Transctional on service methods becoz they indirectly apply on DAO class methods becoz service class methods internally calls DAO calls DAO class one or more methods directly or indirectly..

Client App ----------->service class ----------->DAO ----------> DB s/w

LocalTxMgmtTest ----------> BankServiceMgmtImpl ---------->BankDAOImpl ---------->DB s/w
+String transferMoney(srcAcno, destAcno, amt)
|+ int withdraw(acno, amt)
|+ int deposite(acno, amt)

In Decl Tx mgmt our b.method just contains b.logic +@Transactional annotation and we do not begin , commit/rollback Tx.

=> Generally at the start of b.method the Tx begins automatically and commits tx at end of the b.method automatically if no exception is raised.. if any runtime exception(unchecked exception) is raised ,then the Tx will be rolled back automatically.

<tx:annotation-driven transaction-manager="*dsTxMgmr*"/>

makes IOC cntainer to enable Transactions based @Transactional annotation in declarative mode by taking given bean id based Trasaction manager..

=> Generally the TransactionManager rollbacks the Tx only for unchecked exceptions..raised in b.methods But we want to make TransactionManager rollbacking the Tx even for checked exceptions then specify that exceptions list in rollbackFor, rollbackForClassName params of @Transactional as shown below..

```
//@Transactional(propagation = Propagation.REQUIRED,timeout =10,rollbackFor = IllegalAccessException.class )
@Transactional(propagation = Propagation.REQUIRED,timeout =10,rollbackForClassName ="java.lang.IllegalAccessExeception")
```

In order to make Transaction manager for not rollbacking Tx for certain unchecked exceptions.. we need to specify that unchecked exception class name in "noRollbackFor" or "noRollbackForClassName" params of @Transactional as shown below

```
@Transactional(propagation = Propagation.REQUIRED,timeout =10,noRollbackFor = IllegalArgumentException.class)
```

we can specify the transaction manager either in xml file or in @Transactional .. but specifying xml file is good practice becoz it the flexibility of modification with out touching source code..

<tx:annotation-driven transaction-manager="dsTxMgmr"/> (xml)

(or)

```
@Transactional(propagation = Propagation.REQUIRED,transactionManager = "dsTxMgmr")
```

<tx:annotation-driven transaction-manager="*dsTxMgmr*" proxy-target-class="*true*"/>

proxy-target-class="true" :: In . Memory Proxy class having TxMgmt code will be coming as the sub class of service calss/target class usisng CGLIB Libraries , So we can not take service class as final class here.. (against of strtegy DP)

proxy-target-class="false" :: (default) In Memory Proxy class halvng TxMgmt code will be coming as the impl class of Proxy/service interface using jdk libraries ,so we can take service/target as final class.. (supports StrategyDP)

if u r working with programatic Tx Mgmt , every b.method runs with new Tx becoz we manually begin the Tx as start of b.method definitation.

=> In Delcarative Tx mgmt b.method may run in no Tx, new Tx or caller supplied Tx becoz Transaction manager controls the Transaction service.. This will be based on the decided

Tx attribute that we configure on the b.method..

Six Transaction Attributes
==========

Required (default) (best)
RequiresNew
supports
mandatory
notsupported
never
nested (special)
(for nested Tx mgmt)

One b.methods logics executing in the tx service started by another b.metod /client App (caller) is called Tx Propagration..

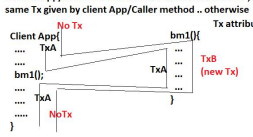
Client App{
bm1(); Txa
}

bm1(){
... Txa
}

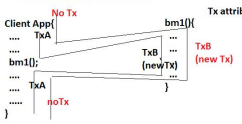
=>Programatic mgmt does not support Tx Propagation i.e Tx of one b.method can not be passed another b.method.

=>Declarative mgmt supports Tx Propagation i.e Tx of one b.method can be passed another b.method

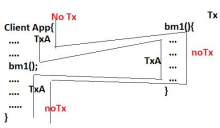
Required
========
if client App/caller method calls the b.method with Tx , then b.method runs in the same Tx given by client App/Caller method .. otherwise b.method runs in the new Tx..



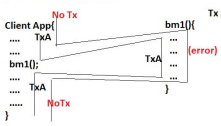
requiresNew
==========
B.method always runs in the new Tx irrespective of wheather client App is calling b.method with or with out Tx..



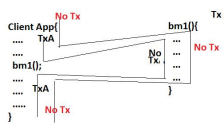
supports
=========
if Client App calls b.method with Tx then b.method runs in same Tx .. if Client App calls b.method with out Tx then b.method runs with out Tx



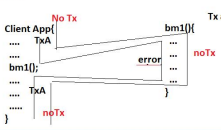
mandatory
==========
=>Client App must call b.method in a Tx .. and b.method runs in the same Tx .. otherwise exception will be thrown..



NotSupported
==========
b.method runs with out Tx , irrespective wheather client App calls the b.method with or with out Tx.



Never
=====
Client app must call the b.method with out Tx.. if called with Tx exception will be raised..



---

client App
(noTx) bm1()

al
@Transaction(propagation=REQUIRED)
bm1(){
Txb (new Tx)
bm2();

}
@Transactional(propagation=SUPPORTS)
bm2(){
...
..
}
(Txb)

REQUIRED(Txb)
REQUIRESNEW (Txc)
Supports (Txb)
Mandatory (Tx b)
NotSupports (no Tx)
Never (error)

Summary table on Tx Attributes
==========

Tx Attribute	Caller Tx/client Tx	B.method Tx
required (default) (best)	TxA noTx	TxA TxB(new Tx)
requiresNew	Tx A noTx	TxB(new Tx) TxB(new Tx)
mandatory	TxA noTx	TxA error
Never	TxA noTx	Error NoTx
Supports	Tx A no Tx	TxA no Tx
NotSuppoted	TxA noTx	noTx noTx

Developing Tx mgmtapplication as 100% code driven App (Java Config Approach)
========== (No xml code)

Thumb rule::
==>configure user-defined classes using Streo type annotations and link them with Configuration classes (@Configuration class) using @ComponentScan (@Component,@Service,@Repository and etc..)
(alternate to sprng bean cfg file (xml))

==>Configure pre-defined classes using @Bean methods.. on 1 method per each spring Bean class object in @Configuration calsses..
(alternate to xml file level <bean> tags)

@Configuration (To make java class as Configuation class)
@ComponetScan(basePackages="com.nt.dao")
public class PersistenceConfig{

@Bean(name="hkDs")
public DataSource createDS(){
HikariDatasource hkDs=new HikariDataSource();
hkDs.setDriverClassName("oracle.jdbc.driver.OracleDriver");
...
....
...
return hkDs;
}

@Bean(name="template")
public JdbcTemplate createJT(){
return new JdbcTemplate(createDs());
}

}

In @Bean methods based sprng beans , if we do not provide bean id/name .. then the method name becomes default bean id like "createDs" , "createJT" and etc..

==> Use AnnotationConfigApplicationContext class to create IOC container by passing Configuration class as the input class name

note:: instead of <tx:annotation-driven> tag place @EnableTransactionManagement on the top of @Configuration class...

<tx:annotation-drivern> ----> @EnableTransactionManagement
<context:componet-scan ....> ----> @ComponentScan
<import > ----------> @Import

Spring Boot driven TransactionManagement
==========

Spring boot is abstract project that is designed on top of Spring ... having
beautiful feature called AutoConfiguration (Making java classes as spring beans
based on the setup that is taken and
based on the jar files that are added to
classpath)

=>our Project extends readymade Spring boot parent Project..inherting
fetures like autoconfiguration.. So that we can code very less in our Project developemt...

Thumb rule for spring boot driven App development (No xml)
==========

=> Configure user-defined classes as spring beans using stero type annotations..
(These will be linked with Configuration class automatically ... if we keep in the
sub packages of @SpringBootApplication class /MainClass/STarter package)

there
class

1.@Configuration
2.@ComponentScan
3.@EnableAutoConfiguration

com.nt
|----->MainApplication.java (@SpringBootApplication)
|-->com.nt.service
|--->BankMgtmService.java
|--->BankMgmtServiceImpl.java (@Service)
|-->com.nt.dao
|--->BankDAO.java, BankDAOImpl.java (@Repoistory)

=> Configure pre-defined classes as spring beans using @Bean methods in
@SpringBootApplication class (main class/starter class) only if the they are not coming
through auto configuration..
note:: we can give inputs to AutoConfiguration process through application.properties/yml file

These dependents will give main jars,
dependent jars and related jars

Like supplying jdbc properties to
create jdbc con objs in jdbc con pool
using fixed keys..

if add spring-starter-jdbc-<version>.jar file to classpath/buildpath
we will be getting the following 3 spring beans through auto configuration
a) HikariDataSource (This will collect jdbc properties from application.properites file
to know for which DB s/w it has to create con objs in the
jdbc con pool)
b) JdbcTemplate (injected with the above DataSource object)
c) DataSourceTransactionManager (injected with the above DataSource object)

application.properties
==========
-name
spring.datasoruce.driver-class=oracle.jdbc.driver.OracleDriver
spring.datasoruce.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasoruce.username=system
spring.datasoruce.password=manager

note:: No need placing @EnableXxx annotations.. (In 99.99% cases) they will be enabled automatically
based on jar files that added to the application..

Procedure to develop LocalTxMgmt Application as spring boot Application
==========

step1) make sure that STS Plugin is installed in eclipse IDE
step2) Create Spring boot starter Project giving following details...

File --> new -->spring starter project --->
Project name:: TxMgmtProj3-AspectjAOP-LocalTxMgmt-Boot
Type :: gradle packging :: jar (standlone App)
version :: 11 language: java
group :: nit
Artifact : TxMgmtProj3-AspectjAOP-LocalTxMgmt-Boot
version:: default ..
package :: com.nt (root package name)

-->next ---> select follwing jars/dependenciers
(a) jdbc api (b) oracle driver
-->finish..

note:: make sure that above chosen java version is bound with project..

step3) add aspect-rt-<ver> .jar , aspecjtweaver-<ver>.jar files info in build.gradle by
collecting them from mvnrepository.com

step4) Analyze ,Which spring beans will come through AutoConfiguration... and supply
neccessary inputs through application.properties file..

application.properties
==========
#jdbc properties for DataSource
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=10
spring.datasource.hikari.minimum-idle=2

To get all the keys of properties file::
https://docs.spring.io/spring-boot/docs/current/reference/html/appendix-application-properties.html#data-properties

Spring boot 2.x auto configuration Datasoruce
A) hikari CP (default)
b) Apache DBCP2

Spring boot 1.x auto configuration Datasoruce
a) Tomcat cp (default)
A) hikari CP
b) Apache DBCP
c) ApacheDBcp2

step5) make sure that all service,dao developed and configured with streo type annotations..

note:: The DataSourceTransactionManager that comes through autoConfiguration will have the default bean id
"transactionManager",

Distributed Tx Mgmt/Global Tx mgmt/ XA TxMgmt
==========

usecase:: transferMoney Operation b/w two diff accounts of two different banks..

=> We need Distributed TransactionManager , Distributed Transaction Service

=>We can get these two things either from third party vendords like Atomikos ,narayana (for standalone Apps) or from Application Servers like weblogic , GlassFish, Wildfly and etc..(for deployable Applicaitons->web applicaitons)

=> Here more than 1 resource (Db s/w) will be involved in a Transaction boundary.. and the Distributed Tx mgmr should perform commit or rollback operation on multiple resources in a single shot using 2pc principle

=> We need special JDBC drivers like XA jdbc drivers..
(AppServer and Atomikos api providing thier own XA jdbc drivers for diff Db s/w)

=> In spring env.. we need to cfg JtaTransactionManager as spring bean injecting Distributed Tx Service and Distributed Tx Manager as show below..

```
<!-- cfg Atomikos DTx service -->
<bean id="atmsDTxService" class="com.atomikos.icatch.jta.UserTransactionImp">
  <property name="transactionTimeout" value="60"/> (optional)
</bean>
<!-- cfg Atomikos DTx Manager -->
<bean id="atmsDtxManager" class="com.atomikos.icatch.jta.UserTransactionManager">
  <property name="forceShutdown" value="true"/> (optiona)
</bean>
```

note:: All TransactionServices are implementing javax.transaction.UserTransaction (I)
note:: All TransactionManagers are implemeting javax.transaction.UserTransactionManager(I)

```
<!-- cfg JtaTransaction Manager-->
<bean id="jtaTxMgmr" class="org.springframework.transaction.jta.JtaTransactionManager">
  <property name="userTransaction" ref="atmsDTxService"/>
  <property name="transactionManager" ref="atmsDtxManager"/>
</bean>
```

JtaTransactionManager intenrally uses the injected thrird party or server managed Distributed Tx service and Distributed Tx manager to perform distributed TxMgmt..

```
In Service class
================
@Transactional(propagation=Propagation.REQUIRED,transactionManager="jtaTxMgmr")
public String transferMoney(long srcAcno, long destAcno, float amt){
  ....
  ...    // the operations on multiple Db s/ws will be committed or will be rolledback
         in single short..
}
```

Db tables (two)
=======

```
Oralce ----> BankAccount
                |--->acno(pk) (n)          | 101  raja   900000
                |-->holderName(vc2)        |
                |--->balance (n)           |

Mysql ------> BankAccount
                |--->acno(pk) (int)        | 102  suresh  500000
                |--->holderName(vc)        |
                |----->balance (float)     |
```

```
OraXADS(Atomokos) ----> JdbcTemplate ---> WithdrawDAOImpl
                                            |-->widraw(long acno, float amount){
                                                 ...
                                                 ..
                                              }
mysqlXSDs(Atomikos) ---> JdbcTemplate -----> DepositeDAOImpl
                                            |--->depoiste(long acno,float amount){
                                                 ....
                                              }
```

=>ojdbc6/7/8/9/10.jar file represents both normal jdbc driver and XA JDBC driver for oracle
=>mysql-connector-java-<ver>.jar file represents both normal jdbc driver and XA JDBC driver for mysql

Developing Disributed Tx Mgmt App(Atomikos) using SpringBoot

==>AtomikosDataSource bean classes will not come through AutoConfiguration
So go for @BeanMethods

⇨ PErsisnteConfig.java

==> JdbcTemplate obj that is coming through AutoConfiguration is having
HikariDs object .. which is not suitable for us..so go Explicit cfg of JdbcTemplate classes
using @Bean methods and injecting AtomokisDatasource Bean objects

=> We need to configure AtomokisTransactionServiceImp, AtomokisTransactionManager and
JtaTransactionManager using @Bean Methods

⇨ AopConfig.java

In order make Spring Boot not performing autoconfiguration on certain spring beans..
we can use the exclude option of @SpringBootApplication annotation as shown below..

```
@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class,
                                  JdbcTemplateAutoConfiguration.class,
                                  DataSourceTransactionManagerAutoConfiguration.class})
```

In Spring Boot AOP /TxMgmt Env.. to take main classes final classes supporting Strategy DP we
need to make spring Boot generating poxy class based proxy interface , not based on target class
for this we need to add the following entry in application.properties file

application.properties
==========

```
spring.aop.proxy-target-class=false    (default is true)
```

=>application.properties/yml file can have both fixed properties and user-defined propeties..
=> To inject user-defineed properties of application.properties file to user-defined spring beans
the Spring boot App we can use
@Value :: To Inject one value to one property in spring bean class (single value loading) -->spring annotation
@ConfigurationProperties :: To inject multiple values to multiple propeties of Spring bean class (bulk loading) -->spring boot Annotation

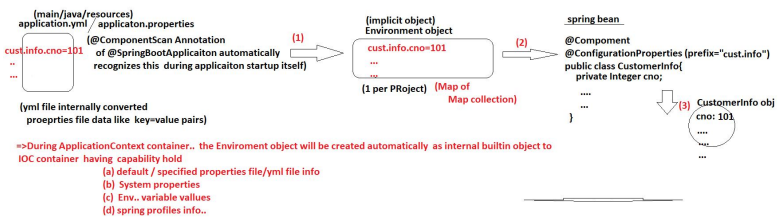
spring bean class can have 4 types bean properties
==========

a) simple type properites (primitive datatypes, String ,wrapper data types)
b) array type properties
c) Collection type propeties (List /Map/Set/Properties)
d) Object Type/Reference type

=>The file that maintains entries in the form key=value pairs is called properties file
=> yml/yaml files extension properties file to avoid repeation of nodes in the keys
and to maintain hierarchy..
=>Spring boot internally converts yml/yaml file to properties file ...
=> we can convert proeprties file to yml file by using Eclipse tool IDE

For yml (Points to remember)
1. No repeat any common node twice
2. value and colon should one space between
3. Only colon is allowed
4. Hierarchy must be maintained
5. Nodes with the same level is mandatory to be same line.
6. Order is not mandatory

Internals related to proeprties/yml files data injection to Spring bean in spring /spring boot env
========== (while working with @Value or @ConfigurationProperties)

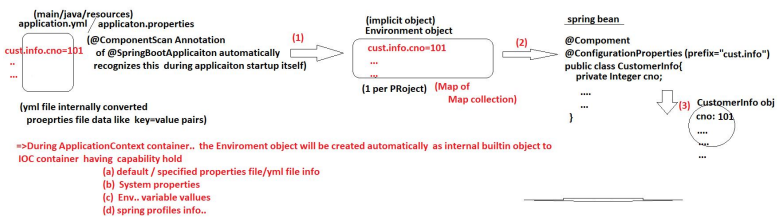


=>During ApplicationContext container.. the Enviroment object will be created automatically as internal builtin object to IOC container having capability hold
(a) default / specified properties file/yml file info
(b) System properties
(c) Env.. variable vallues
(d) spring profiles info..

=>The file that maintains entries in the form key=value pairs is called properties file
=> yml/yaml files extension properties file to avoid repeation of nodes in the keys
and to maintain hierarchy..
=>Spring boot internally converts yml/yaml file to properties file ...
=> we can convert proeprties file to yml file by using Eclipse tool IDE

For yml (Points to remember)
1. No repeat any common node twice
2. value and colon should one space between
3. Only colon is allowed
4. Hierarchy must be maintained
5. Nodes with the same level is mandatory to be same line.
6. Order is not mandatory

Internals related to proeprties/yml files data injection to Spring bean in spring /spring boot env
========== (while working with @Value or @ConfigurationProperties)



=>During ApplicationContext container.. the Enviroment object will be created automatically as internal builtin object to IOC container having capability hold
(a) default / specified properties file/yml file info
(b) System properties
(c) Env.. variable vallues
(d) spring profiles info..

Realtime utilization of @ConfigurationProperties
==========

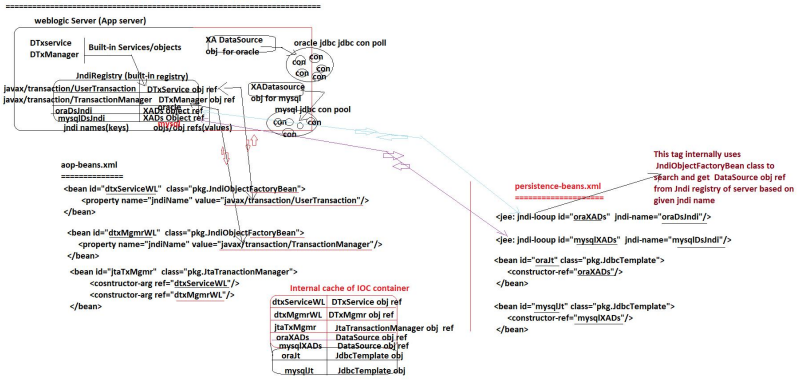
application.propeties

```
# aop
spring.aop.proxy-target-class=false

#for oracle AtomikosDataSoruceBean
dtx.orads.uniqueResourceName=oraXA
dtx.orads.xaDataSourceClassName=oracle.jdbc.xa.client.OracleXADataSource
dtx.orads.xaProperties.databaseName=xe
dtx.orads.xaProperties.URL=jdbc:oracle:thin:@localhost:1521:xe
dtx.orads.xaProperties.user=system
dtx.orads.xaProperties.password=manager

#for mysql AtomikosDataSoruceBean
dtx.mysqlds.uniqueResourceName=mysqlXA
dtx.mysqlds.xaDataSourceClassName=com.mysql.cj.jdbc.MysqlXADataSource
dtx.mysqlds.xaProperties.databaseName=mtsp612db
dtx.mysqlds.xaProperties.URL=jdbc:mysql:///mtsp612db
dtx.mysqlds.xaProperties.user=root
dtx.mysqlds.xaProperties.password=root
```

In PersistenceConfig.java
==========

```
@Configuration
public class PersistenceConfig {
    @Bean(name="oraXADs")
    @ConfigurationProperties(prefix = "dtx.orads")
    public AtomikosDataSourceBean createOracleXADS() {
        AtomikosDataSourceBean ds=null;
        ds=new AtomikosDataSourceBean();
        return ds;
    }

    @Bean(name="mysqlXADs")
    @ConfigurationProperties(prefix = "dtx.mysqlds")
    public AtomikosDataSourceBean createMySqlXADS() {
        AtomikosDataSourceBean ds=null;
        ds=new AtomikosDataSourceBean();
        return ds;
    }

    @Bean(name="oraJt")
    public JdbcTemplate createOraJt() {
        return new JdbcTemplate(createOracleXADS());
    }

    @Bean(name="mysqlJt")
    public JdbcTemplate createMysqlJt() {
        return new JdbcTemplate(createMySqlXADS());
    }

}
```

Working with SErver Managed Distributed TxService and Distributed Tx Manager
==========



aop-beans.xml
==========

```
<bean id="dtxServiceWL" class="pkg.JndiObjectFactoryBean">
    <property name="jndiName" value="javax/transaction/UserTransaction"/>
</bean>

<bean id="dtxMgmrWL" class="pkg.JndiObjectFactoryBean">
    <property name="jndiName" value="javax/transaction/TransactionManager"/>
</bean>

<bean id="jtaTxMgmr" class="pkg.JtaTranactionManager">
    <cosntructor-arg ref="dtxServiceWL"/>
    <constructor-arg ref="dtxMgmrWL"/>
</bean>
```

Internal cache of IOC container

dtxServiceWL	DTxService obj ref
dtxMgmrWL	DTxMgmr obj ref
jtaTxMgmr	JtaTransactionManager obj ref
oraXADs	DataSource obj ref
mysqlXADs	DataSource obj ref
oraJt	JdbcTemplate obj
mysqlJt	JdbcTemplate obj

persistence-beans.xml
==========

This tag internally uses JndiObjectFactoryBean class to search and get DataSource obj ref from Jndi registry of server based on given jndi name

```
<jee: jndi-loop id="oraXADs" jndi-name="oraDsJndi"/>

<jee: jndi-loop id="mysqlXADs" jndi-name="mysqlDsJndi"/>

<bean id="oraJt" class="pkg.JdbcTemplate">
    <constructor-ref="oraXADs"/>
</bean>

<bean id="mysqlJt" class="pkg.JdbcTemplate">
    <constructor-ref="mysqlXADs"/>
</bean>
```

note:: FactoryBeans are selfless beans i.e they do not give their objects...
they always given resultantant objects spring beans to use other beans..

Spring ORM Advantages
=====================
(a) Avoids boilter plate code by supplying Template classes
(b) Common exception handling (we need not to handle ORM f/w specific exceptions
we need to catch and handle the common DataAccessExcepion)
note: spring jdbc, spring orm , spring data modules throw common exceptions
like DataAccessException class hierarchy classes
(c) Persistence logic is portable acorss the multiple Db s/ws.. and Entity classes are portable across the multiple ORM frameworks

(d) Common Transaction Management support

(e) Common single row methods calls (given by JPA) and
common JPQL (Java Persitence Query Language)
and etc...

Spring with Hibernate
=====================
=>It says write B.logic in spring and write peristence logic in hibernate by taking the support of the Spring ORM supplied HibernateTemplate class

given based Template method Design Pattern which say that it defines a algorithm where super class/common class will take care commong logic by leaving specific logics to sub classes/developers to supply.

DAO class
|-->
HibernateTemplate obj
|-->SessionFactory object (dependent) (SessionFactory obj multiple services of hibernate)
|---->Ds (DataSource object)
|----> Entity classes with annotations/ mapping files
(reconanded) (old)
|-->hibernate properties
1.dialect (capable generating SQL queries)
2.show_sql (to see SQL queries on the console)
3.hbm2ddl.auto :: validate(default),create, update(best), create-drop

creates the db tables if they are not available, uses them if ther are alredy availabe.. alters them by adding new cols if neccessary...

4
we can develop Spring ORM Apps in approaches
(a) Using xml driven cfgs (Cfg both user-defined and pre-defined classes using xml)
(b) Using xml +annotation driven cfgs (user-define calsses using annotations and pre-defined clases using xml)
(c) 100% Code driven cfgs (user-defined classes uisng annotations and pre-defined classes using @Bean methods in @Configuration class)
(d) Using spring boot cfgs (User-defind classes using annotation and prefined classes using @Bean methods if at all there are not coming through auto configuration)

impotant spring Annotations for layered Apps
@Component -->To make java class as spring bean with no specialties
@Service -->To make java class as sprign bean cum service class (support TxMgmt)
@Controller ---> To make java class as spring bean cum controller
@Repository ---> To make java class as spring bean cum DAO class (With Exception transalation support)

Entity classes with Annotations
Basic Annotations
@Entity (JPA) (mandatory)
@Table (JPA) (optional)
@Id (JPA) (mandatory)
@Column (JPA) (optional)
@Type (HB) (optional)

Annotations in Entity classes
a) JPA Annotations
b) Hibernate Annotations
c) Java Config annotataions (JSE,JEE modules)
d) Third party annotations

if Entity class name is matching with DB table name and Entity properites are matching with Db table col names then placing @Table, @Column annotaitons optional.. if want to use dynamic schema generation (db tables generations) it is remcomanded to place them to controll on type, length , unique and etc.. details..

To supprt Disk caching of HB second Level cache

```
@Entity
@Table(name="STUDENT")
public class Student implements Serializable{
   @Id ------------------------> makes this property related colum as pk column .. while creating Db table..
   @Column(name="SNO")
   @Type(type="int")
   private Integer sno;
   @Column(name="SNAME",length="20",nullable=false)
   @Type(type="string")
   private String sname;
   @Column(name="SADD",length="20")
   @Type(type="string")
   private String sadd;
   @Column(name="AVG")
   @Type(type="float")
   private Float avg;
   getters && getters
   ....
   ....
   ...
   ...
}
```

To support java numberic data type ranges while creating Dynamic db table.. length attribute values will not be reflected
In Db table creation.. but String cols the length will be reflected.

(The above operations will not take class. If u are working already available Db tables)

persistence-beans.xml
```
<beans .....>
   <!--Datasource cfg-->
   <bean id="hkDs" class="pkg.HikariDataSource">
      ....
      ...
   </bean>
   <!--SessionFactory cfg-->
   <bean id="sesfact" class="pkg.LocalSessionFactoryBean" >
      <property name="dataSource" ref="hkDs"/>
      <propety name="annotatedClasses">
         <array>
            <value>com.nt.entity.Student</value>
         </array>
      </property>
      <property name="hibernateProperties">
         <props>
            <prop key="dialect">org.hibernate.dialect.Oracle10gDialect</prop>
            <prop key="show_sql">true</prop>
            <prop key="hbm2ddl.auto">update</prop>
         <props>
      </property>
   </bean>
   <!-Hibernate Template class cfg-->
   <bean id="template" class="pkg.HibernateTemplate">
      <costructor-arg ref="sesfact"/>
   </bean>

</beans>
```

Factory Bean that gives HB SessionFactory obj as Resultant obj based on the given injected values like DS, HB Properties ane tc..)
(Selfless Bean)

Internal cache of IOC container

hkDs	HikariDataSoruce obj ref
sesfact	SessionFactory obj ref
template	HibernateTemplate class object ref

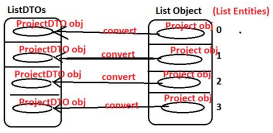
Bulk Operations in Hiberante
==========
a) HQL(Hiberate Query Language)/JPQL (Java Perssitence Query language)
b) Native SQL
c) Criteria API

To execute HQL/JPQL Select Queries we use find(), findXxx() methods on
HiberanteTemplate class object.. similary for non-select queries use bulkUpdate(-) method

SQL ---> DB s/w dependent Queries --> writtern by using Db tablename, col names.
HQL/JPQL --> DB s/w independent queries -->Written Using Entity class name and its Properties

HB dialect converts HQL/JPQL queries into underlying DB s/w SQL queries..

=>upto hb 5.1 HQL/JPQL supports both named(:<name>) and positional params(?) in the HQL/JPQL queries.. but from hiberaten 5.2 we have support only for named Parameters..
So most of the find() findXxx() that supports positional params are deprecated in HiberanteTemplate class.. (they did this work in support to spring data jpa)



Thum rule to Spring app as 100%Code driven Application (no xml file)
==========

rule1) configure user-defined classes as spring bean using stereo type annotations
(@Repository,@Service,@Component and etc..)
and link them with @Configuration class @ComponentScan annotation

rule2) configure pre-defined classes as spring beans using @Bean methods (1 method per 1 bean obj)
of @Configuration class (alternate spring bean cfg file (xml file))

rule3) create AnnotationConfigApplicationContext container as IOC container by giving
@Configuration class as input class.

Thumb rules to develop spring Boot App (No xml + auto configuration)
---
makes certain classes as spring beans
based on the jar files the added to the
Application..

rule1) Configure user-defined classes as spring
beans using stereo type annotations

rule2) configure pre-defined classes as spring beans using @Bean methods
only if they are not coming as spring beans through AutoConfiguration..
note:: we can give inputs to AutoConfiguration beans with the support of
application.properties/yml file

rule3) get IOC container from SpringApplication.run(-) from @SpringBootApplication
class(main class/starter class) to write further coding..

a)@Configuration (make current class configuration class)
b)@ComponentScan(Automatically scans current pkg and sub pkgs spring beans,configuration classes)
c)@EnableAutoConfiguration (To give certains pre-defined classes as spring bean classes based
on the jar files that are added)

if we added spring-boot-jdbc-starter-<ver>.jar file to classpath/buildpath the following
spring beans will come through autoConfiguration
a) HikariDataSource obj
b) JdbcTemplate obj
c) DataSourceTransactionManager obj

Procedure to develop spring ORM with hiberante App as spring boot app using eclipse
==========

step1) create spring starater Project by choosing
project name:: ORMProj3-Boot
type ::Gradle (3.x) packaging :: jar
version :: 11 language:: java
groupId: nit artifcatId :: ORMProj3-Boot
default package :: com.nt -->next -->
choose oracle driver , jdbc api as starter jars

step2) Add spring orm , hibernate-core jars to build.gradle

step3) add following entries in application.properties (main/java/resoruces) to support
DataSoruce Autoconfiguration..

```
#Datasource cfg
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
```

step4) and develope remaining code as per spring boot App requirement...

Spring Data
==========

Before arrival of Spring Data
==========

Java App----------> SQL DB s/w
jdbc, spring jdbc, ORM f/ws like hibernate , spring ORM

Two types of Db s/w^{s}
1. SQL DB s/w(RDBMS DB s/w)
eg:: oracle,mysql,postgresql, db/2 and etc..
Maintain data in the form Db table rows columns

employee with 4 details
employee with 10 details
employee with 40 details
For this we need design db table with 40 cols .. and most of time the unused cols memory is wasted..

Data with fixes structure then go for SQL Db s/ws...

2. NOSQL Db s/ws
Java App -----mongoDB API------> MongoDB (noSQL)
Java App -----Couchbase API------> CouchBase (noSQL)
and etc..
eg:: MongoDB, Cassandra, Kafka and etc... couch base ,redis

=> Here No db tables, rows and cols every thing stored in form of documents
So memory will be alloected only for given values.. Not wastage of memory
empoyee with 4 values ==>Doc with values
employee with 10 values ==>Doc with values

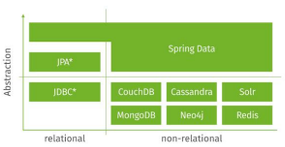
Data with Dynamic structure whose data fields dynamically varies then go for No SQL..

note:: No support from Spring to talk No SQL DB s/ws
note :: No ORM frameworks to talk with NO SQL Db s/ws.

Before arrival of spring data.. we need to interact SQL , NOSQL DBs seperately..

Spring Data provides unified model programming to interact with both SQL and NO SQL Db s/w
Spring Data is given based on two design patterns 1) Repository Pattern 2) Proxy Pattern

Spring Data



Spring Data JDBC to talk with SQL Db s/w in JDBC style
Spring Data JPA to talk with SQL Db s/w in ORM style
Spring Data MongoDB to talk with NOSQL Db s/w (MongoDB)
Spring Data CouchDB to talk with NOSQL Db s/w (CouchDB)
and etc...

Sring data is extenstion module fo spring framework.. developed based on spring to interact both SQL and NOSQL DB s/ws in a unified model...

refer https://spring.io/projects/spring-data for spring data its sub modules..

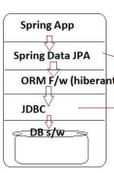
Spring modules
==========
Core
AOP
JDBC/DAO
ORM
Web MVC
Context
Web
TxMgmt
spring mail
and etc..

Spring Extension modules
==========
Data
SpringBatch
Spriring social
Spring security
and etc..

Spring data features
==========
=>Powerful repository and custom object-mapping abstractions
=>Dynamic query derivation from repository method names
=>Implementation domain base classes providing basic properties
=>Support for transparent auditing (created, last changed)
=>Possibility to integrate custom repository code
=>Easy Spring integration via JavaConfig and custom XML namespaces
=>Advanced integration with Spring MVC controllers
=>Experimental support for cross-store persistence

We cover
spring data JPA (for SQL Db s/w)
spring Data MongoDB (for No SQL Db s/w)

Spring DAta JPA
==========



To learn and use Spring Data JPA we need strong plain hiberante knowledge .. becoz it intenally uses hiberante...

use plain ORM or spring ORM or spring data JPA while dealing limited amount of data..
with
of
while dealing huge amount data that comes batch by batch)
(Census App development)
(plain JDBC or spring JDBC)
(recomanded)

=>No DAO classes while working with Spring data modules.. becoz work with Repository Interfaces by specifying Entity class name and its ID Property type.. For that interface
Spring Data internally generates proxy class having persistence logic .. i.e we do not write PErsistence logic we work with Dynmically generated PErsistence logic.. in the proxy class..

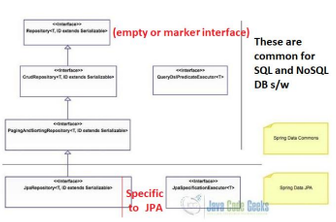
Student db table in oracle
==========
Student (db table)
|--->sno (pk) (n)
|--->sname (vc2)
|--->sadd (vc2)
|--->avg (float)

Entity class
```
@Entity
public class Student implements Serializable{
   @Id
   private Integer sno;
   private String sname;
   private String sadd;
   private  float avg;
     //getters & getters
      .....
}
```
Should estends Repository(I) directory or indirectly

Repository interface
==========
So many method declrations for CURD operations

```
package com.nt.repo;
public interface  StudentRepo  extends CrudRepository<Student,Integer>{

}
```
Entity class name
ID Property type

=>The IOC container creates InMemory Proxy class at runtime implementing StudentRepo interface and provides impl for the all methods that are inheriteed from CrudRepository Interface



Before spring Data using Spring ORM or plain ORM
100 Db tables ---> 100 DAO interfaces
100 DAO impl clases having curd operation persistence logics methods (10 method in each class) (bit heavy)

With Spring data JAP
==========
100 db tables ----> 100 repository interfaces (programmer part is over)
Here 100 impl classes are generated internally using Proxy DP having PErsistence logic So here burden on the programmer is reduced..

Using spring data REpository in SErvice class
==========

```
public interfce StudentMgmtService{
   public  String   register(StudentDTO dto);
}
```

```
@Service("studService")
public class StudentMgmtServiceImpl
                    implements StudentMgmtService{
  @Autowired
  private   StudentRepo studRepo;   //Injectes Generated proxy class obj

  public  Stirng  register(StudentDTO dto){
    //convert dto to entity
      ......
    //use  repo
    Student st=studRepo.save(entity);
    if(st!=null)
      return " student registered";
    else
      return "student not registered";
  }
}
```

For Proxy DP
==========
https://github.com/natarazworld/NTHB914/tree/master/HBProj3-ProxyDP

For Hiberante Intro Videos
==========
https://www.youtube.com/watch?v=XNEtyH3xqwQ&list=PL9_WQN37oCKpgNh867mzU6Sn1IBx7i-_b&index=5

https://www.youtube.com/watch?v=CGVM4-Z7od8&list=PL9_WQN37oCKpgNh867mzU6Sn1IBx7i-_b&index=6

Sat -->9am :: lombok api ->part1
sun --> 9am :: lombok api ->part2
sat --> 6pm :: Spring data
sun --> 11am :: spring data

What is the difference b/w DAO DesignPattern and Repository Design Pattern?

DAO Pattern	Repository Pattern
(a) Contains DAO (I) and DAO Impl class developed by Programmer	(a) Contains Repository(I) given by the programmer and implemementation class for Interface given underlying framework or container
(b) Does not use Proxy Pattern at all	(b) Repository(I) impl class will come based on Proxy Design Pattern
(c) Writing Persistence logics is resposibility of the Programmer	(c) here the underying F/w or Cotnainer will take care
(d) In one DAO we can write persisntece logics of multiple DB table	(d) One Repository(I) we can deal with only one DB table or Entity class
(e) In DAO we can write both jdbc style , o-r mapping style persistence logic	(e) here only o-r mapping style persistence logic possible
(f) DAO is DB table centric	(f) Repository is Entity class centric

note:: Both are given to seperate persisntece logic from other logics of the Application.

In Spring DAta JPA we can develop persistence logics by using

(a) Using the inherited methods of Repository Interface to the dynamically generated proxy class (basics)
(for curd perations)

(b) Using explicitly declared findXxx methods in our Repository Interface (select operations)

(c) Using @Query , @Query with @Modify methods based Custom HQL/JPQL queries or Native SQL queries
(select) (non-select)

note:: We can call Storeed procedures using spring data JPA

Different approaches of developing spring data Applications

a) Using xml driven cfgs
b) Using annotation driven cfgs
c) Using 100% Code /Java Config cfgs
d) Using Spring Boot cfgs ....

Cleint App ---DTO--->service class (b.logic) ---BO/Entity---> Repository (persistence logic) ---->DB s/w

(Controller is ignored here)

Resources in Application Development

a) Customer.java --->Entity class @Entity
b) CustomerDTO.java --->DTO class
c) CustomerRepo.java ---> Repository Interface
d) CustomerMgmtService.java -->service Interface
e) CustomerMgmtSErviceImpl.java --> service Impl class @Service
f) Stater class /Main class with @SpringBootApplicaiton
h) application.properties (opt)
|--->DataSource properties (driver classname ,url , username, password)
|---> ORM/JPA prooerties (dialect, hbm2ddl.auto , show_sql,format_sql)
(opt) (opt)

steps to develop Spring Data App as spring Boot App

step1) Create Spring Boot starter Project
File -->new --->Spring starter project
Project name:: DataProj1
type :: gradle 3.x packaging :jar
language :: jaa version:: 11
groupId :: nit
artifcatId :: DataProj1
package :: com.nt ---> select jars --> spring data jpa, oracle driver , lombok api

step2) add entries application.properperties

step3) deveop the remaining resources... (See SpringDataProj1-CRUDRepo)

step5) Run the App ..

note:: <S extends T> S save(S entity); --> this takes entity object and returns another object entity class as saved object..

=> save(-) given spring boot data jpa perfoms save object operation if record is not available having given id property value in pk colum.. otherwise it will perform update operations (when no Generator is cfg , if gnerator is cfg always performs save object operation..)

=> generally in spring boot jpa we use @Qyery + @Modify annoation methods for update opeations... becoz we always take id property with Generator cfg..

note:: there are no merge(-) ,saveOrU pdate(-) ,update(-) methods in Spring boot data jpa... we can bring theie effect using with @Query+@Modify annoation methods..

<S extends T> Iterable<S> saveAll(Iterable<S> entities)

use

This mehtod is very ful in batch insertion of records... usecases like group ticket reservation, group ticket booking and etc...

Sir in batch process if get any exception comes for ny one of object... is it save remaining or not saved any thing?
ans) no --->Transaction management is enabled by default..

Differ methods for delete object operation in CrudRepository

==========

void delete(T entity)
void deleteAll(Iterable<? extends T> entities)
void deleteAll()
void deleteById(ID id)

Always prefer -->load and delete or check delele operation to return success or failure message from service class method..

=> save(-) given spring boot data jpa perfoms save object operation if record is not available having given id property value in pk colum.. otherwise it will perform update operations (when no Generator is cfg , if gnerator is cfg always performs save object operation..)

=> generally in spring boot jpa we use @Qyery + @Modify annoation methods for update opeations... becoz we always take id property with Generator cfg..

note:: there are no merge(-) ,saveOrU pdate(-) ,update(-) methods in Spring boot data jpa... we can bring theie effect using with @Query+@Modify annoation methods..

<S extends T> Iterable<S> saveAll(Iterable<S> entities)

use

This mehtod is very ful in batch insertion of records... usecases like group ticket reservation, group ticket booking and etc...

Sir in batch process if get any exception comes for ny one of object... is it save remaining or not saved any thing?
ans) no --->Transaction management is enabled by default..

Differ methods for delete object operation in CrudRepository

```
void delete(T entity)
void deleteAll(Iterable<? extends T> entities)
void deleteAll()
void deleteById(ID id)
```

Always prefer -->load and delete or check delele operation to return success or failure message from service class method..

To check wheather record is record is availdble or not

```
boolean existsById(ID id)
```

To perform save object opeation

```
<S extends T> S save(S entity)
<S extends T> Iterable<S> saveAll(Iterable<S> entities)
```

To select operations

```
Iterable<T> findAll()
Iterable<T> findAllById(Iterable<ID> ids)
Optional<T> findById(ID id)
```

Iterable is supper interface for all collections

from java 5 (earlier it was Collection(I))

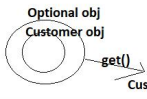
```
Customer cust=custRepo.findById(no);
if(cust!=null){
  CustomerDTO dto=new CustomerDTO();
  BeanUtils.copyProperties(cust,dto);
}
```

kids code

```
Optional<Customer> opt;
   Customer cust=null;
   CustomerDTO dto=null;

//use repo
   opt=custRepo.findById(id);
   if(opt.isPresent()) {
        cust=opt.get();
        dto=new CustomerDTO();
        BeanUtils.copyProperties(cust, dto);
   }
   return dto;
```

```
Optional<Customer> opt=custRepo.findById(id);
    if(opt.isPresent()) {
        Customer cust=opt.get();
        CustomerDTO dto=new CustomerDTO();
        BeanUtils.copyProperties(cust, dto);
    }
```

adult code..



Optional API is introduced from Java8 To checks wheather recieved object is null or not.. and to perform various operations when present.. when not present.. This basically avoid NullPointerException from java code...

PagingAndSortingRepository

```
=>sub Interface of CrudRepository (I)
=> Super interface of JpaRepository(I)
=> Given for sorting (ASE/DESC) and pagination (displaying records page by page -->report genertion) activities

methods
=======
   Iterable<T> findAll(Sort sort) // Only Sorting
   Page<T> findAll(Pageable pageable) // for pagination with /with our sorting
```

Iterable<T> findAll(Sort sort) // Only Sorting

```
//use repo
    itEntities=custRepo.findAll(Sort.by(asc?Direction.ASC:Direction.DESC,
                                        property));
```

single String

```
//use repo
    itEntities=custRepo.findAll(Sort.by(asc?Direction.ASC:Direction.DESC,
                                        properties));
```

var args (String)

Page<T> findAll(Pageable pageable) // for pagination with /with our sorting

For pagination

to give 0 based >=1

we need inputs (pageNo, pageSize) in the Pageable object.

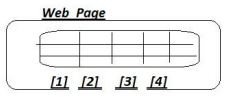
```
// pageNo,pageSize  PageRequest class implements Pageable
Pageable pageable=PageRequest.of(3,5);
```

pageNo, pagesize

20 -> 5,5,5,5

on

Web Page



Instead of displaying all records in a single page ,display them page by page .. that to based given pageNo and PageSize

We get output from findAll(Pageable obj) either in the form Page object/Slice obj haivng List<T> (list entities) other details like pageNo, total no.of pages , count of recods, nextpageInfo and etc..

eg::

```
Page<Customer> page=custRepo.findAll(pageable);
```

=> save(-) given spring boot data jpa perfoms save object operation if record is not available having given id property value in pk colum.. otherwise it will perform update operations (when no Generator is cfg , if gnerator is cfg always performs save object operation..)

=> generally in spring boot jpa we use @Qyery + @Modify annoation methods for update opeations... becoz we always take id property with Generator cfg..

note:: there are no merge(-) ,saveOrU pdate(-) ,update(-) methods in Spring boot data jpa... we can bring theie effect using with @Query+@Modify annoation methods..

<S extends T> Iterable<S> saveAll(Iterable<S> entities)

use

This mehtod is very ful in batch insertion of records... usecases like group ticket reservation, group ticket booking and etc...

Sir in batch process if get any exception comes for ny one of object... is it save remaining or not saved any thing?
ans) no --->Transaction management is enabled by default..

Differ methods for delete object operation in CrudRepository

```
void delete(T entity)
void deleteAll(Iterable<? extends T> entities)
void deleteAll()
void deleteById(ID id)
```

Always prefer -->load and delete or check delele operation to return success or failure message from service class method..

To check wheather record is record is availdble or not

```
boolean existsById(ID id)
```

To perform save object opeation

```
<S extends T> S save(S entity)
<S extends T> Iterable<S> saveAll(Iterable<S> entities)
```

To select operations

```
Iterable<T> findAll()
Iterable<T> findAllById(Iterable<ID> ids)
Optional<T> findById(ID id)
```

Iterable is supper interface for all collections

from java 5 (earlier it was Collection(I))

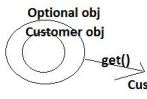
```
Customer cust=custRepo.findById(no);
if(cust!=null){
  CustomerDTO dto=new CustomerDTO();
  BeanUtils.copyProperties(cust,dto);
}
```

kids code

```
Optional<Customer> opt;
   Customer cust=null;
   CustomerDTO dto=null;

//use repo
   opt=custRepo.findById(id);
   if(opt.isPresent()) {
        cust=opt.get();
        dto=new CustomerDTO();
        BeanUtils.copyProperties(cust, dto);
   }
   return dto;
```

```
Optional<Customer> opt=custRepo.findById(id);
    if(opt.isPresent()) {
        Customer cust=opt.get();
        CustomerDTO dto=new CustomerDTO();
        BeanUtils.copyProperties(cust, dto);
    }
```

adult code..



Optional API is introduced from Java8 To checks wheather recieved object is null or not.. and to perform various operations when present.. when not present.. This basically avoid NullPointerException from java code...

PagingAndSortingRepository

```
=>sub Interface of CrudRepository (I)
=> Super interface of JpaRepository(I)
=> Given for sorting (ASE/DESC) and pagination (displaying records page by page -->report genertion) activities

methods
=======
   Iterable<T> findAll(Sort sort) // Only Sorting
   Page<T> findAll(Pageable pageable) // for pagination with /with our sorting
```

Iterable<T> findAll(Sort sort) // Only Sorting

```
//use repo
    itEntities=custRepo.findAll(Sort.by(asc?Direction.ASC:Direction.DESC,
                                        property));
```

single String

```
//use repo
    itEntities=custRepo.findAll(Sort.by(asc?Direction.ASC:Direction.DESC,
                                        properties));
```

var args (String)

Page<T> findAll(Pageable pageable) // for pagination with /with our sorting

For pagination

to give 0 based >=1

we need inputs (pageNo, pageSize) in the Pageable object.

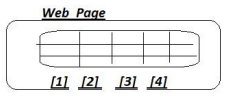
```
// pageNo,pageSize  PageRequest class implements Pageable
Pageable pageable=PageRequest.of(3,5);
```

pageNo, pagesize

20 -> 5,5,5,5

on

Web Page



Instead of displaying all records in a single page ,display them page by page .. that to based given pageNo and PageSize

We get output from findAll(Pageable obj) either in the form Page object/Slice obj haivng List<T> (list entities) other details like pageNo, total no.of pages , count of recods, nextpageInfo and etc..

eg::

```
Page<Customer> page=custRepo.findAll(pageable);
```

What is the diff b/w Slice and Page object in PagingAndSortingRepository?

Ans) Slice obj holds info about the current report page data info.. not about other report pages data..
Using this object we can get info about only current report page data.. by calling various getXxx()
but we can not get total no.of records , total no.of pages info .

Page obj holds info about the all the report pages data .. including current port pages data..
Using this object we can get info about current report page data.. by calling various getXxx()
and we can also get total no.of records , total no.of pages info .

Sliect<T>
↑ exstends
Page<T> (I)

note:: Page object is bit heavy object compare to Slice object becoz it containts multiple List objects /Sliect objs internally..

```
// Page page=custRepo.findAll(pageable);
Slice slice=custRepo.findAll(pageable);
//convert page object into Iterable obj (List collection)
//Iterable<Customer> itEntities=page.getContent();

System.out.println(slice.getNumber()+" "+slice.hasContent()+" "+slice.isEmpty()+" "+slice.isFirst()+"
"+slice.getNumberOfElements());
//System.out.println(page.getNumber()+" "+page.hasContent()+" "+page.isEmpty()+" "+page.isFirst()+"
"+page.getNumberOfElements()+" "+page.getTotalElements()+" "+page.getTotalPages());
```

JpaRepository (given based jpa specification)

=>Most of the methods inherited from CrudRepository,PagingAndSortingRepository and redefined as per JPA specification.. But also having some direct methods

Repository(I)
↑ extends
CrudRepository(I)
↑ extends
PagingSortingRepository(I)
↑ extends
JpaRepository(I)

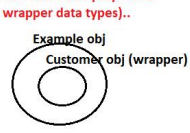
```
<S extends T> List<S> findAll(Example<S> example)
<S extends T> List<S> findAll(Example<S> example,  Sort sort)
void deleteAllInBatch(); -->Deletes batch of records by generating single delete query
void deleteInBatch(Iterable<T> entities)
T getOne(ID id)
void flush()  --> without waiting for TxManager to commit the
                  Tx , it will write changes to Db s/w..
```

<S extends T> List<S> findAll(Example<S> example)

Example obj wrapper object aroung given Object...It given in hibernate api havig similar behaviour of Optional (java 8)

Collects entity object from wrapper Example object and uses all non-null value properties int the SQL query preparation having and claluse (prefer taking wrapper data types)..

```
//Convert DTO to entity
Customer entity=new Customer();
BeanUtils.copyProperties(dto, entity);
//prepare Example object (Wrapper around Entity object)
Example  example=Example.of(entity);
//use repo
List<Customer> listEntities=custRepo.findAll(example);
```



---

what is difference among findAll() methods avaiable 3 Repositories?

=>findAll() fo CrudRepository does not support pagination and sorting
=>findAll() fo PagingSortingRepository supports pagination and sorting
(These two methods return Iterable(S extends T) obj
Does not allow Example objs as args..

=>findAll() fo JapRepository retunrs List<S extends T> colelction
|--->Here support is there for Sorting .. no support pagination
|--->Allows Example objs as arguments

JpaRepository (spring boot data jpa)

public void deleteAllInBatch() -->delets the records in a batch by generating single delete SQL query

```
@Override
public String removeAllCustomers() {
        boolean flag=true;
        //use repo
        if(custRepo.count()>=1) {
          custRepo.deleteAllInBatch();
          flag=true;
        }else {
              flag=false;
        }
        return flag?"All customers are deleted":" No customers found";
}
```

Why there is no update(-) and saveOrUpdate(-) in Spring data jpa?

Ans) since save(-) method internally persist() or merge() to perform save object or update object operation .. So there is no need of seperate update(-) method..

save(-) method calls persist(-)(JPA) if records already not available otherwise calls merge(-) (JPA) to update the existing record..

Internal Code of save(-) of data jpa

```
@Transactional
    @Override
    public <S extends T> S save(S entity) {

        if (entityInformation.isNew(entity)) {
            em.persist(entity);
            return entity;
        } else {
            return em.merge(entity);
        }
    }
```

Custom PErsistence logic using Spring data jpa

a) using findByXxx() methods declaration in Repostory interface
b)Using @Query methods (HQL/JPQL , SQL queries)

a) using findByXxx() methods declaration in Repostory interface

=>Converts findByxxx() abstract method declared in Repository into SQL Query dynamically at runtime

Syn::
<Return type> findBy<PropertyName> <Condition>(<params>);

=>suports only Select operations
=> For non-select operations go for @Query methods

=>Supports both
Entity Query Opeations (selecting all colum values)
and Scalar Query Operations (select specific col values aggragate results)
(Also called Projections)

Entity Queries (getting all column values by Condtion)

=>On static Properties Dependency Injections are not not possoble even autowiring also not possbile
=>we can get Bean class obj from Spring container by passing object type or reference type

```
//get Bean class object
custRepo=ctx.getBean(CustomerRepo.class);
```

=>Container generated proxy class for Data Repository internally becomes spring bean automatically..

```
/*=> SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE CADD=?
  =>if no condition is taken default condition is Where with " is " or "="
  if wrong property name is given in findByXxx method then we get
"Invalid derived query! No property cadd1 found for type Customer! Did you mean 'cadd'?
  //List<Customer> findByCadd(String addrs);*/

// SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTIMER WHERE CNAME=?
        //List<Customer>     findByCname(String cname);

      // SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTIMER WHERE BILLAMT> ?
        //List<Customer> findByBillAmtGreaterThan(double amount );
       // SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTIMER WHERE BILLAMT< ?
        //List<Customer> findByBillAmtLessThan(double amount );

        //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE CNAME like 'r%'
          // List<Customer> findByCnameLike(String initChars);

          //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE CNAME like 'r%'
          //List<Customer> findByCnameStartingWith(String initChars);

          //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE CNAME LIKE '%h'
      //  List<Customer> findByCnameEndingWith(String lastChars);

        //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE CNAME LIKE '%j%'
          //Iterable<Customer> findByCnameContaining(String chars);

         // SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE CADD IS NULL
          //Iterable<Customer> findByCaddIsNull();

      // SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE CADD IS NOT NULL
        //Iterable<Customer> findByCaddIsNotNull();
         Iterable<Customer> findByCaddNotNull();
```

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is,Equals	findByFirstname,findByFirstnameIs,findByFirstnameEquals	… where x.firstname = 1?
Between	findByStartDateBetween	… where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull	findByAgeIsNull	… where x.age is null
IsNotNull,NotNull	findByAge(Is)NotNull	… where x.age not null
Like	findByFirstnameLike	… where x.firstname like ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> age)	… where x.age not in ?1
True	findByActiveTrue()	… where x.active = true
False	findByActiveFalse()	… where x.active = false
IgnoreCase	findByFirstnameIgnoreCase	… where UPPER(x.firstame) = UPPER(?1)

```
//SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE BILLAMT>=4000 AND BILLAMT<=5000
        // Iterable<Customer>   findByBillAmtGreaterThanEqualAndBillAmtLessThanEqual(double min,double max);

      //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE BILLAMT BETWEEN(4000,6000)
        // Iterable<Customer>   findByBillAmtBetween(double min,double max);

        //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE CNAME='raja' OR CADD='hyd'
         //Iterable<Customer>   findByCnameEqualsOrCaddEquals(String name,String addrs);
        // Iterable<Customer>   findByCnameOrCadd(String name,String addrs);

        //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE BILLAMT BETWEEN(4000,5000) ORDER BY CNAME
         // Iterable<Customer>   findByBillAmtBetweenOrderByCnameDesc(double min,double max);

      //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE   CADD <> 'hyd'
      // SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE   CADD!='hyd'
        // Iterable<Customer>   findByCaddNot(String addrs);

         //SELECT CNO,CNAME,CADD,BILLAMT FROM CUSTOMER WHERE      CADD IN('vizag','hyd','delhi')
        //    Iterable<Customer>   findByCaddIn(Collection<String> cities);
```

note:: Taking findByXxx(-,-) with multiple proeprties and multi conditions makes method names very lengthy and complex... So go for @Query methods based custom query logics... i.e always use findByXxx() method for simple and single property conditions..

findByXxx(-,-) method for Scalar Operations Using Projections

if findByXxx(-) is returning single Record having all col values then we can take just <T> as the return type insteead of List<T> or Iternable<T>

```
Customer findByCname(String name);
```

findByXxx(-,-) method for Scalar Operations Using Projections

findByXxx(-) are abstract methods generating select SQL queries either giving all col values (Entities quries) or specific col values (scalar queries -Usinng Projectctons) by applying where clause condition

findByXxx methods with Projections
==========

PRojections are columns in SQL... getting records by specifying col names is called Projections concept...

(a) Static Projections (Always gives fixed specific col values)
(b) Dynamic Projections (we can get varying col calues)

(a) Static Projections *(scalar queries)*

#1 Take a interface as inner interfce repository interface and declare getter methods by choosing the properties of entity class

```
interface ResultsView1{
        Integer getCno();
        String  getCname();
    }
```

#2) Design findByXxx having the above TypeView interface as part of return type or as return type

```
//SELECT CNO,CNAME FROM CUSTOMER WHERE CADD=?
    List<ResultsView1> findByCadd(String addrs);
```

#3) Call the the above method in service class/client App to get the result and to process the result..

```
//invoike method
    List<ResultsView1> view1List=custRepo.findByCadd("hyd");
    for(ResultsView1 v:view1List) {
        System.out.println(v.getCno()+"  "+v.getCname()+"  "+v.getClass());
    }
```

type

note:: HEre we can not chnage view type dynamically in the return of findByXxx(-) method

---

Dynamic Priojections (SCalar opeations)

=>Allows to view type dynamically at runtime .. by designing the method with the support of java5 Generics..

step1) Take multiple view type intrfaces having declartion of getter methods...

```
interface ResultsView1 extends View{
        Integer getCno();
        String  getCname();
    }

interface ResultsView2 extends View{
        String getCname();
        double  getBillAmt();
    }

interface View{
    }
```

step2) //design the method with generics with dynamic Projections

```
// <T> List<T> findByCadd(String addrs,Class<T> clzz);
<T extends View> List<T> findByCadd(String addrs,Class<T> clzz);
```

step3) invoke methods in client App /service class/..

```
List<ResultsView1> view1List=custRepo.findByCadd("hyd",ResultsView1.class);
    for(ResultsView1 v:view1List) {
        System.out.println(v.getCno()+"  "+v.getCname());
    }
    System.out.println(".....");

    List<ResultsView2> view2List=custRepo.findByCadd("hyd",ResultsView2.class);
    for(ResultsView2 v:view2List) {
        System.out.println(v.getCname()+"  "+v.getBillAmt());
    }
```

Entity operations :: Getting all col values
scalar operations :: GEtting specific col values

Core Java recap
==========

```
class Person{
  ...
}

class  Customer exstends PErson{
  ..
}

class Employee extends  Person{
  ..
}

class  Student extends Person{
  ...
}
```

Design method method having ability to dislay any object data
==========

option1::) public void display(Object obj) --> Not good prattice
-->typcasting is required ,may give ClassCastException

option2::) public void display(Person obj) --> Not good prattice
-->typcasting is required ,may give ClassCastException

option3::) public <T> void display(Class<T> clazz) (jdk5 Generics)
*
Good prartic and avoids ClassCastException

public <T extends Person> void display (Class<T extends Person> clazz)
*
(More Good Pratice)

* --> is not return type of method .. It is Generic type declartion ..So that can be used through out method signature and definitation in return type, parameter types

Limitations of findByXxx() while wring custom PErsinstence logics in Spring data JPA

a) supports only select operations
b) Writing findByXxx(-) method with multiple properties and multiple conditions makes method name very big and complex
c) Working with Projections (Scalar Operations) bit complex.. (Especially taking view type interfaces.. for diff combinations we need to take diff view type interfaces..
d) Not So readable for prorgammers...
e) does not support aggragate operations.

conclusion:: use findByXxx(-) methods only for simple select operations with simple conditions..

To overcome these problems and to write Custom persitence logic in all possible angels go for @Query methods

---

@Query methods

=> Alows to write persitence logic using custom HQL/JPQL and Native SQL Queries
=> Support both select and non-select operations (except insert opearetion --->use ses.save(-) for insert operation)
=> Supports both single row and bulk operations with our choice conditions..
=> Query can have both postional(?1,?2,?3,...) and named parameters (:<var1> ,:<var2> ,:<max>) (named params are good)
=> Allows to invoke PL/sQL procedures and functions
=> Supports joins
=> allows both Entity and scalar query operations ....

Why @Query methods do not support single record insert operation?
=> Query based inseration can not use generators to generate the id value.. So use ses.save(-) for that...

Syntax::
========

In Repository interface..

```
@Query(" query with params' ?,?,?)
public <ReturnType> <method>(param1,param2,param3)
```

even palce named params

=>java method param values will be mapped with query positional params(?) automatically..

=>java method param values will be mapped with query name params(?) automatically if both names are mathing.. if not matching we can bind them using @Param annotation.

HQL/JPQL
========

=> Will be writen based on Entity class name and property names (not based on db table names and col names)
=> These queries are Db s/w independent ... becoz these object based SQL queries
=> These query par˙tial case -Sensitive i.e HQL/JPL keyworkds are not case-sensitive but entity class name, property names are case-sensitive
=> These queries Learning curve is very small...
=>These queries not support jdbc style positional params like ?,?,? but supports jpa style positional params ?1,?2,?3 becoz hibernate 5.2 onwords support for jdbc style postional params have beem removed..

Db table name
Enity class name
alias name
column name

```
SQL>  SELECT  * FROM  CUSTOMER
HQL/JPQL>  SELECT  cust  FROM com.nt.entity.Customer  cust
HQL/JPQL>    FROM   Customer  cust
HQL/JPQL>    FROM   Customer
```

```
SQL>  SELECT * FROM CUSTOMER  WHERE CNO>=?  AND CNO<=?
HQL/JPQL> FROM  Customer  WHERE  cno>=?1  AND  cno<=?2
```

Entity class name
property name

All HQL/JPQL Queries internally converted to SQL queries with positional params before sending them DB s/w.. Becoz DB s/w understands only SQL queries..

```
@Query("FROM  Customer WHERE  billAmt>=?1 and billAmt<=?2 ")
   Iterable<Customer>  getCustomersByBillAmtRange(double start, double end);
```

=>when multiple postional params.. we can change their order ... but there should not be any gap in the numbering .. we need take params in the method according to that changed order..
=>Giving index to more than 3 or 4 postional params is very complext.. to overcome this problem use named params (:<name>) (parameter with name..)

if u use jdbc stype plain positional parameters (?) in HQL/JPQL queries then we get java.lang.IllegalArgumentException: JDBC style parameters (?) are not supported for JPA queries.

```
//-------- select bulk operations with named params(Entity queries (all col values)) ----------
        @Query("FROM  Customer WHERE  cadd IN(:cityOne,:cityTwo,:cityThree)")
    Iterable<Customer>  getCustomersByCityNames(@Param("cityOne")String city1,
                                                @Param("cityTwo")String city2,
                                               @Param("cityThree")String city3);


        @Query("FROM Customer  cust WHERE cust.cname=:name")
        Iterable<Customer>  getCustomerByName(@Param("name")String name );
```

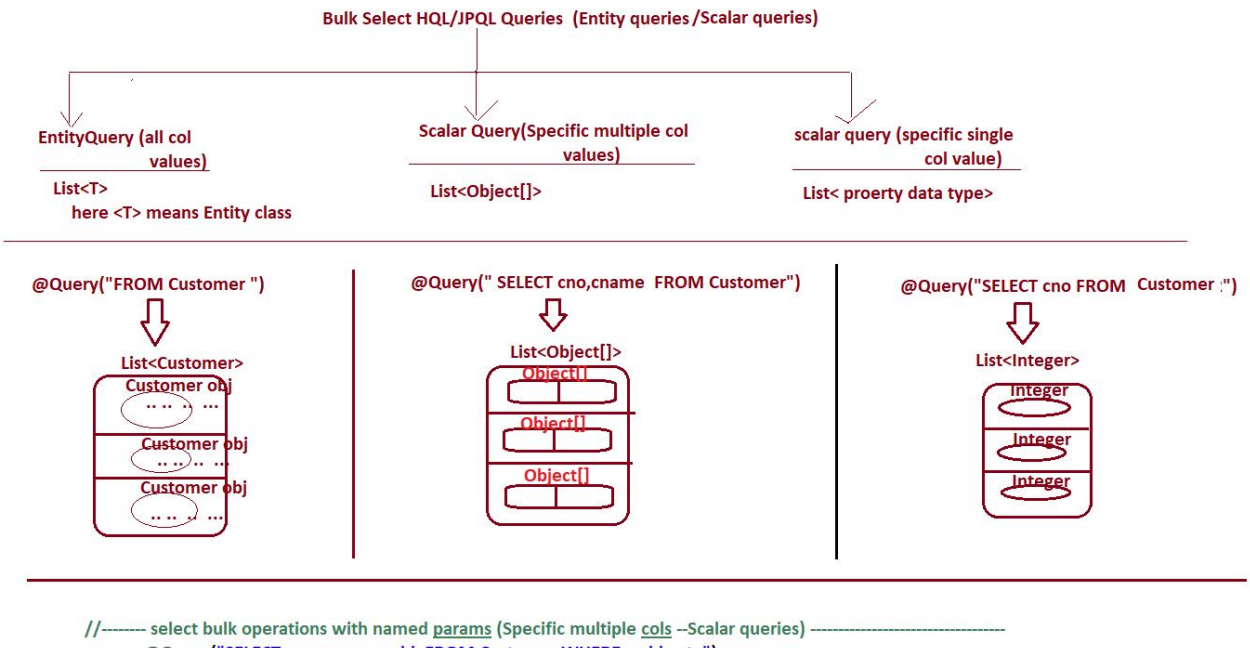
note:: @Param is used to java method param values to HQL/JPQL named param value...

note:: if named param name and java method param name are matching then no need of giving @Param ***

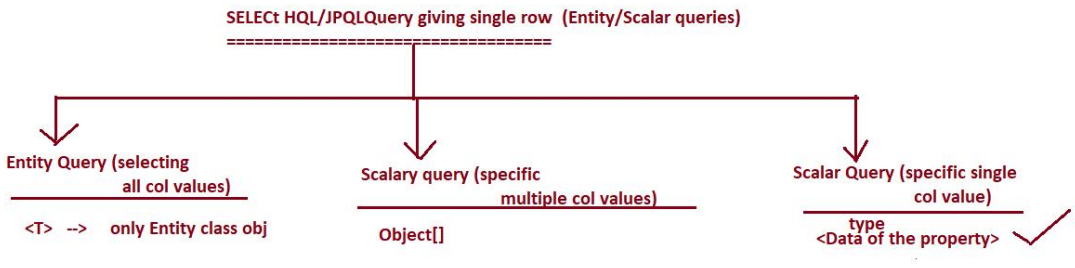
single

we can not place both postional and named parameters in @Query method JPQL/HQL query .. it throws org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'customerRepo': FactoryBean threw exception on object creation; nested exception is java.lang.IllegalArgumentException: Either use @Param on all parameters except Pageable and Sort typed once, or none at all!

```
        @Query("FROM Customer  WHERE cno>=?1 and cno<=:max")          //invalid
              Iterable<Customer>  getCustomersByCnoRange(Integer min ,@Param("max")Integer end);
```




```
//-------- select bulk operations with named params (Specific multiple cols --Scalar queries) ----------
        @Query("SELECT cno,cname,cadd  FROM Customer WHERE cadd=:cty")
         Iterable<Object[]>  getDataValuesByCity(@Param("cty")String city);


//-------- select bulk operations with named params (Specific single col values --Scalar queries) ----------
            @Query("SELECT billAmt FROM Customer WHERE cadd IN(:city1,:city2)")
            Iterable<Double>  getBillAmtByCities(@Param("city1")String city1,
                                                 @Param("city2")String city2);
```



```
@Query("FROM Customer  cust WHERE cust.cname=:name")
      Iterable<Customer>  getCustomerByName(String name );
```

if the name of the named parameter and the name of method param is matching then there is no need of placing @Param annotation as shown above..

For this we need to need extra setting for javac compiler at project level
right click on the project ---->properties ----> java compiler ----> [ ] select "Store information about method params" -> .. ..

```
//---------------------------------------SingleRow select operation using HQL/JPQL  (Entity query) ------------
    @Query("FROM Customer WHERE  cname=:name")  //assume "CNAME" col is having unique
    Customer   getCustomerByCname(String name);
   //---------------------------------------SingleRow select operation using HQL/JPQL  (Scalar query-selecting multiple specific cols) ------------

    @Query("SELECT cno,cname FROM Customer WHERE  cname=:name")
   Object  getDataValuesByCname(String name);

   //---------------------------------------SingleRow select operation using HQL/JPQL  (Scalar query-selecting single specific col) ------------
    @Query("SELECT billAmt FROM Customer WHERE  cname=:name")
    Double  getBillAmtByCname(String name);
```

Non-Select Operations (both bulk and single row operations)

=>Here we can take our choice conditions...
=> Only update and delete possible... for insert use repo.save(-) method..

=> use @Query, @Modify togather on the methods..

=> if service calss not taken we should @Transactional explicitly .. otherwise not required..
Add in the Repository interface..

1.To take advantage of generators
2. generay insert query does not need any condition
3. HQL/JPQL is not having any insert Query..

```
//-----------------------------update opeartion ------------------------------
        @Modifying
        @Query("UPDATE Customer SET  billAmt=billAmt+ :extraAmount WHERE cadd=:city")
        int  modifyCustomerBillAmtByCity(String city ,double extraAmount);

        //-----------------------------delete opeartion  ------------------------------
        @Modifying
        @Query("DELETE FROM Customer WHERE cadd IS NULL")
        int  deleteCustomersIfCaddIsNull();
```

Native SQL /Oginal SQL Queries

==>These DB s/w dependent queries ... so makes persistence logic as DB s/w dependnet.

==>Use this when HQL/JPQL does not support certain operation (like insert operation, calling PL/SQL procedure functio calling DB s/w specific aggregate functions like sysDate(oracle), now()(mysql) and etc..)

==>Allows both named ,positional params

- JPA style (?1,?2,..)
- jdbc style (?,?,?,..)

Executing Native SQL queries using @Query methods

```
//@Query(nativeQuery = true, value = "SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=?")
    //@Query(nativeQuery = true, value = "SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=?1")
    @Query(nativeQuery = true, value = "SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=:addrs")
    Iterable<Customer> getCustomerByAddrs(String addrs);
```

note:: native SQL supports jdbc style positional params(?) ,
jpa style positional params and
named parameters

```
@Query(nativeQuery = true, value = "SELECT SYSDate from dual")
    java.util.Date   getSysDate();
```

use Native SQL to perform those operations that are not possible with HQL/JPQL like
using DB specific aggrate functions , insert queries , calling PL/SQL procedures...

```
@Query(nativeQuery = true, value = "SELECT SYSDate from dual")
    java.util.Date   getSysDate();
```

note:: Native SQL queries based persistence logic is bad becoz it makes persistence logic
as DB s/w dependent persistence logic.

---

insert operation using native SQL query

```
@Transactional
  @Query(nativeQuery = true, value = "INSERT INTO CUSTOMER(CNO,BILL_AMT,CADD,CNAME) VALUES(HIBERNATE_SEQUENCE.NEXTVAL,?,?,?)")
  @Modifying
   int  insertCustomer(double billAmt,String addrs,String name);
```

Calling PL/SQL Procedures and functions In spring data jpa

=> PL/SQL procedure or function is like a java method to execute bunch of statements togather in
a single block
=> PL/SQL procedure does not return a value.. where function returns a value..
=> Industry uses more of PL/SQL procedures becoz we can results throuth out params...
=> Java method contains param name and type.. where as PL/SQL procedure or function params
will have param name, type and mode (IN,OUT,INOUT)

IN-->INPUT
OUT-->OUTPUT
INOUT -->INPUT and OUTPUT

y:=x*x ; y -->out param
x --> IN param

x:=x*x (x --> INOUT param)

in oracle PL/SQL
==========
= (for comparsion)
:= (for assignmnet)

note:: PL/SQL programming
is specific to each Db s/w

Instead of writing same persistence logic in multiple modules/Apps of a Project in the form of
SQL or HQL/JPQL queries , it recomaded to write only for 1 time as PL/SQL procedure or function and
use it multiple Apps or modules..

a) Authentication logic
b) Attendence calucation logic and etc..

Java Prjects 60 to 80% => SQL or HQL/JPQL based Queries
20 to 40% => PL/SQL procedures..

In spring DATA JPA we call PL/SQL procedures in 3 ways

a) Using @Query (Native SQL query approach) | we can call only in params ,no params procedures
b) Using @Procedure Approach
c) Usisng EntityManager (** Best) | We can call any procedure having in params ,
out params or no params

step1) create PL/SQL procedure having IN Params in mysql using Workbench

Workbench ---> go to ur ntsp612db ---> right clikc stored procedures ---> create procedure --->
name:: GET_CUSTOMER_BY_ADDRS -> type the code.... replacing null

```
SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=ADDRS;
```

also param Info IN ADDRS VARCHAR(10)

⇩

```
//generated code
DELIMITER $$                                  //auto generated
USE `ntsp612db`$$
CREATE PROCEDURE `GET_CUSTOMERS_BY_ADDRS` (IN ADDRS  VARCHAR(10))
BEGIN
   SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=ADDRS;
END$$

DELIMITER ;                                   //auto generated
```

step2) Prepare @Quey method in the Repository interface..

```
@Query(nativeQuery = true,value= "{call GET_CUSTOMERS_BY_ADDRS(?)}")
    Iterable<Customer>  fetchCustomerDataByAddrs(String addrs);
```

step3) invoke the method in the client App

```
// call PL/SQL procedure
    custRepo.fetchCustomerDataByAddrs("hyd").forEach(System.out::println);
```

Transaction Isolation Levels

I of ACID properties

=> Isolation level are given to apply locks at various levels and to avoid problems related sumulataenous or concurent access of Db s/w data...

Problems	solution (Isolation Levels)
(a) Dirty Read problem	Read Committed
(b) Non-Repetable Read Problem	Repeatable Read
(c) Panthom Read Problem	Serializable

Dirty Read Problem

User A	User B
a) User A begins the Tx and gets balance of the Account rs: 10000	
b) User A deposits rs: 5000 to Accnount bal=bal+amt (15000)	(c) User B withdrws rs: 12000 from account simultaenously bal=bal-amt (3000)
(d) User A , aborts the Tx , due that he gets 10,000 back to Account and UserA's operation becomes Dirty Read Operation..	User A,B are Joint Account Holders ->Assume Db s/w allowing UnCommited reads.

ReadCommitted Isolation level is solution for Dirty Read Problem where Db s/w is made to read only Committed Data and that to not in concurrent mode.

Non- Repeatable Read Problem

User A	User B
a) User A begins Tx and gets 10 records by issuing "Select SQL Query".	
	(b) User B , executes the update SQL Query which updates either all or some of the records came for User A's select SQL Query.
(c) At the end of Tx, User A re issues same Select SQL query to reuse the Data But he gets the modified data.. This make data as non-repetable data..	

Solution for this problem is "Repeatable read" Isolation level that means it makes DB s/w to apply write locks on Db s/w.

Panthom Read problem

USer A	user B
a) At the begining of Tx, user gets 10 records by issuing select Query with condition..	
	b) User B simulataenously inserts 4 records matching with condition of User A's select Query.
c) At the end of Tx , user A re issues same select Query but he ges 14 recrods (more than expected) ->Getting more recods than expected is called Panthom read problem.	

=>Serializable Isolation level solves this by applying read and write locks on DB s/w.

TL/PL Chooses the Transaction Isolation level based on the project need and Support of Isolation level with Undlerying DB s/w.

To apply Transaction Isolation levels from JDBC code

```
con.setTransactionIsolation(2/4/8);
        (or)
con.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED/
                            TRANSACTION_REPEATABLE_READ/
                            TRANSACTION_SERIALIZABLE);
```

To apply Transaction Isolation levels from Hibernate App

In hibernate.cfg.xml

```
<property name="hibernate.connection.isolation">2/4/8 </property>
```

To apply Transaction Isolation levels from spring App

```
@Transactional(propagation=REQUIRED, isolation= Isolation.SERIALIZABLE)
        (or)
<tx:advice id="txAdvice">
  <tx:attributes>
      <tx:method name="*" propatation="required"
                          isolation="serializable"/>
  </tx:attributes>
</tx:advice>
```

Executing Native SQL queries using @Query methods

```
//@Query(nativeQuery = true, value = "SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=?")
    //@Query(nativeQuery = true, value = "SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=?1")
    @Query(nativeQuery = true, value = "SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=:addrs")
    Iterable<Customer> getCustomerByAddrs(String addrs);
```

note:: native SQL supports jdbc style positional params(?) ,
jpa style positional params and
named parameters

```
@Query(nativeQuery = true, value = "SELECT SYSDate from dual")
    java.util.Date   getSysDate();
```

use Native SQL to perform those operations that are not possible with HQL/JPQL like
using DB specific aggrate functions , insert queries , calling PL/SQL procedures...

```
@Query(nativeQuery = true, value = "SELECT SYSDate from dual")
    java.util.Date   getSysDate();
```

note:: Native SQL queries based persistence logic is bad becoz it makes persistence logic
as DB s/w dependent persistence logic.

---

insert operation using native SQL query

```
@Transactional
  @Query(nativeQuery = true, value = "INSERT INTO CUSTOMER(CNO,BILL_AMT,CADD,CNAME) VALUES(HIBERNATE_SEQUENCE.NEXTVAL,?,?,?)")
  @Modifying
   int  insertCustomer(double billAmt,String addrs,String name);
```

Calling PL/SQL Procedures and functions In spring data jpa

=> PL/SQL procedure or function is like a java method to execute bunch of statements togather in
a single block
=> PL/SQL procedure does not return a value.. where function returns a value..
=> Industry uses more of PL/SQL procedures becoz we can results throuth out params...
=> Java method contains param name and type.. where as PL/SQL procedure or function params
will have param name, type and mode (IN,OUT,INOUT)

IN-->INPUT
OUT-->OUTPUT
INOUT -->INPUT and OUTPUT

y:=x*x ; y -->out param
x --> IN param

x:=x*x (x --> INOUT param)

in oracle PL/SQL
=
= (for comparsion)
:= (for assignmnet)

note:: PL/SQL programming
is specific to each Db s/w

Instead of writing same persistence logic in multiple modules/Apps of a Project in the form of
SQL or HQL/JPQL queries , it recomaded to write only for 1 time as PL/SQL procedure or function and
use it multiple Apps or modules..

a) Authentication logic
b) Attendence calucation logic and etc..

Java Prjects 60 to 80% => SQL or HQL/JPQL based Queries
20 to 40% => PL/SQL procedures..

In spring DATA JPA we call PL/SQL procedures in 3 ways

a) Using @Query (Native SQL query approach) | we can call only in params ,no params procedures
b) Using @Procedure Approach
c) Usisng EntityManager (** Best) | We can call any procedure having in params , out params or no params

step1) create PL/SQL procedure having IN Params in mysql using Workbench

Workbench ---> go to ur ntsp612db ---> right clikc stored procedures ---> create procedure --->
name:: GET_CUSTOMER_BY_ADDRS -> type the code.... replacing null

```
SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=ADDRS;
```

also param Info IN ADDRS VARCHAR(10)

⇩

```
//generated code
DELIMITER $$
USE `ntsp612db`$$ | //auto generated
CREATE PROCEDURE `GET_CUSTOMERS_BY_ADDRS` (IN ADDRS VARCHAR(10))
BEGIN
   SELECT CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=ADDRS;
END$$ |

DELIMITER ; | //auto generated
```

step2) Prepare @Quey method in the Repository interface..

```
@Query(nativeQuery = true,value= "{call GET_CUSTOMERS_BY_ADDRS(?)}")
    Iterable<Customer>  fetchCustomerDataByAddrs(String addrs);
```

step3) invoke the method in the client App

```
// call PL/SQL procedure
    custRepo.fetchCustomerDataByAddrs("hyd").forEach(System.out::println);
```

Calling PL/SQL procedure using EntityManager

=> EntityManager object manages entity obejects lifecyle and uses them for persistence operations
=>It is JPA object.. hold session,SessionFactory objs if the underlying ORM is hibernate
=> It is like HibernateTemplate class obj but it is common for all ORM frameworks...

note:: if add spring-boot-data-jpa-starter to libraries .. The EntityManager obj comes through
AutoConfiguration..

=>In service class we inject using @Autowired EntityManager manager;
=>In Client App we can get
EntityManager manager=ctx.getBean(EntityManager.class);

step1) make sure that above mysql PL/SQL is running property (having entity query with
In param)

step2) In client App write following code.. to call PL/SQL procedure usisng EntityManager.

note:: Using Entity Manager no restrication to
call any type of PL/SQL procedure..

note:: @Procedure we can not deal with Entity Query based PL/SQL procedures , we can
deal with Scalar query based that to single out param PL/SQL procedures..

Calling PL/SQL procedure from Mysql that is having Scalar query with IN,OUT PArams

a) Using @Query not possible
b) Using @Procedure possible
c) Using EntityManager possible (any thing is possible)

Using @Procedure

step1) make sure PL/SQL procedure is ready in mysql having scalar query with one OUT
param and one or more IN params..

```
DELIMITER $$
USE `ntsp612db`$$
CREATE PROCEDURE `GET_CUSTOMERS_COUNT_BY_ADDRS` (IN addrs varchar(10),out cnt int )
BEGIN
  SELECT COUNT(*) INTO cnt FROM CUSTOMER WHERE CADD=addrs;
END$$

DELIMITER ;
```

step2) Add the following mehtod inRepository Interface having @Procedure

```
/*@Procedure(procedureName = "GET_CUSTOMERS_COUNT_BY_ADDRS")
    int  fetchCustomersCountByAddrs(String addrs);  // method param values will become pl/sql procedures in params values
                                                    // Pl/SQL procdure out param value becomes method return value
                                                    //So procedure can only one out param..
    */
                    (or)
    @Procedure
    int  GET_CUSTOMERS_COUNT_BY_ADDRS(String addrs);
```

//Here PL/SQL procudure name is taken
as java method name So need of specifying
procedure name seperarely

@Procedure is not industry standard

step3) invoke method in client App by accessisng Repository object

```
//get Bean class object
  CustomerRepo custRepo=ctx.getBean(CustomerRepo.class);
  //invoke methods
  /*int count=custRepo.fetchCustomersCountByAddrs("hyd");
  System.out.println("hyd coustomer count::"+count);
  */
  int count=custRepo.GET_CUSTOMERS_COUNT_BY_ADDRS("hyd");
  System.out.println("hyd coustomer count::"+count);
```

Using EntityManager Approach for the above PL/SQL procedure

step1) make sure that above PL/SQL proceduring in stable
step2) Writing following code using EntityManager in client App

```
//get EntityManager object
  EntityManager manager=ctx.getBean(EntityManager.class);
  //create StoredProcedureQuery object
  StoredProcedureQuery procedure=manager.createStoredProcedureQuery("GET_CUSTOMERS_COUNT_BY_ADDRS");
  //register params
  procedure.registerStoredProcedureParameter(1, String.class, ParameterMode.IN);
  procedure.registerStoredProcedureParameter(2, Integer.class, ParameterMode.OUT);
  //set value to in Params
  procedure.setParameter(1,"hyd");
  //get results from out param
    int count=(int)procedure.getOutputParameterValue(2);
    System.out.println("hyd customers count::"+count);

   //close entity manager
    manager.close();
```

=>if result is List<T> then use getResultList() method
=>if reuslt isList<Object[]> then use getResultList() method
if result is in outparams hodling single values
then use getOutParameterValue(-)

Calling Pl/SQL Procedure of Oracle using EntityManager of Spring boot data jpa

step1) create PL/SQL procedure in Oracle

```
CREATE OR REPLACE PROCEDURE GET_CUSTOMERS_BY_ADDRS( ADDRS IN VARCHAR2 ,
                                                    DETAILS  OUT SYS_REFCURSOR) AS
BEGIN

 OPEN DETAILS FOR
   SELECT  CNO,CNAME,CADD,BILL_AMT FROM CUSTOMER WHERE CADD=ADDRS;

END ;                                ;
```

Cursor hodls this select query reuslt..

note:: Cursor in Memory variable of oracle Pl/SQL programming having capability to hold
bunch of records ... it is like JDBC ResultSet object
=>Sys_Refcusor is a built-in cursor data type in oracle PL/SQL programmer
=>Oracle PL/SQL programming does not allow any results with out OutParams So
->we can not use @Query to call PL/SQL procedures of oracle
->@PRocedure can not deal with entity Queries (select all col values) So we can not
it here.. So we we need to use only EntityManager option.

step2) make sure that u r application is pointing to oracle Db s/w..

step3) Use EntityManager in Client App as shown below..

Calling PL/SQL Procedure of oracle that is having Scalar query (Specific multiple col vlaues) using EntityManager

step1) create PL/SQL procedure in oracle using SQL developer

```
CREATE OR REPLACE PROCEDURE FETCH_CUSTOMERDETAILS_BY_ADDRS
(
  ADDRS IN VARCHAR2
, DETAILS OUT SYS_REFCURSOR
) AS
BEGIN
  OPEN DETAILS FOR
    SELECT CNAME,BILL_AMT FROM CUSTOMER WHERE CADD=ADDRS;

END FETCH_CUSTOMERDETAILS_BY_ADDRS;
```

step2) Write code in Client App Using EntityManager

```
//get EntityManager object  for Oracle PL/SQL procedure (SCALAR Query)
                EntityManager manager=ctx.getBean(EntityManager.class);
        // Create StoredProcedure Query object
                StoredProcedureQuery query=manager.createStoredProcedureQuery("FETCH_CUSTOMERDETAILS_BY_ADDRS");
        //register IN, OUT params
                query.registerStoredProcedureParameter(1, String.class, ParameterMode.IN);
                query.registerStoredProcedureParameter(2, Class.class, ParameterMode.REF_CURSOR);
                //set value IN param
                query.setParameter(1,"hyd");
        //gather results by calling PL/SQL procedure
                List<Object[]> list=query.getResultList();
                list.forEach(row->{
                        for(Object val:row) {
                                System.out.print(val+"  ");
                        }
                        System.out.println();
                });
         //close entity manager
          manager.close();
```

Calling PL/SQL procedure of Oracle that Performs Authentication using EntityManager

step1) create db table in oracle having users, passwords

```
CREATE TABLE "SYSTEM"."USERINFO"
  (    "UNAME" VARCHAR2(15 BYTE),
       "PWD" VARCHAR2(15 BYTE),
        PRIMARY KEY ("UNAME"));
```

step2) create PL/SQL Procedure having authtentication logic

```
CREATE OR REPLACE PROCEDURE P_AUTHENTICATION1
(
  UNAME IN VARCHAR2
, PASS IN VARCHAR2
, RESULT OUT VARCHAR2
) AS
   CNT NUMBER(4);
BEGIN
  SELECT COUNT(*) INTO CNT FROM USERINFO u WHERE u.UNAME=UNAME AND u.PWD=PASS;
    IF(CNT<>0) THEN
      RESULT:='VALID CREDENTIALS';
    ELSE
      RESULT:='INVALID CREDENTIALS';
    END IF;
  END P_AUTHENTICATION1;
```

step3) Write following code in Client app using EntityManager

```
//get EntityManager object  for Oracle PL/SQL procedure (SCALAR Query)
                EntityManager manager=ctx.getBean(EntityManager.class);
                // Create StoredProcedure Query object
                StoredProcedureQuery query=manager.createStoredProcedureQuery("P_AUTHENTICATION1");
                //register IN, OUT params
                query.registerStoredProcedureParameter(1, String.class, ParameterMode.IN);
                query.registerStoredProcedureParameter(2, String.class, ParameterMode.IN);
                query.registerStoredProcedureParameter(3, String.class, ParameterMode.OUT);
                //set IN param values
                 query.setParameter(1,"raja");
                 query.setParameter(2,"rani1");
                 //get reuslt from OUT param by calling PL/SQL procedure
                 String result=(String)query.getOutputParameterValue(3);
                System.out.println("result::"+result);
```

note:: While working with @Procedure we can have only 1 IN param and only OUT param.. So there are multiple limitations to use it.

=>PL/SQL Procedure does not return a value where as PL/SQL function returns a value
=> if PL/SQL procedure want to return 10 outputs we take 10 out params
=> if PL/SQL function want to return 10 outputs we take 9 out params and 1 return value.

note:: In Spring Data JPA there is not direct provision to call PL/SQL function But can add plain jdbc code by unwrapping Session, jdbc con,CallableStatement objects through Entity Manager...

In This Approach we need not to have Entity classes and repositories matching to the SQL queries of PL/SQL procedures or functions..

Example
=========

step1) keep PL/SQL function ready .. with out having any link u r repository and Entity class

```
CREATE OR REPLACE FUNCTION FX_GET_EMP_DETAILS_BY_DESG
(
  NO IN NUMBER
, DESG OUT VARCHAR2
, NAME OUT VARCHAR2
, DNO OUT NUMBER
) RETURN FLOAT AS
  BSAL FLOAT;
BEGIN
   SELECT ENAME,JOB,SAL,DEPTNO INTO NAME,DESG,BSAL,DNO FROM EMP WHERE EMPNO=NO;
   RETURN BSAL;
END FX_GET_EMP_DETAILS_BY_DESG;
```

step2) Write the Following code in service class using EntityManager...

```
@Service("custService")
@Transactional
public class CustomerServiceImpl implements CustomerService {
        @Autowired
  private EntityManager manager;

        @Override
        public void getEmpDetails(int no) {
                //get Session
                Session ses=manager.unwrap(Session.class);
                float sal=ses.doReturningWork(new ReturningWork<Float>() {
                        public Float execute(Connection con) throws SQLException {
                                //create CallableStatement obj
                                CallableStatement cs=con.prepareCall("{?= call FX_GET_EMP_DETAILS_BY_DESG(?,?,?,?)}");
                                //register Return, OUT params with JDBC types
                                cs.registerOutParameter(1,java.sql.Types.FLOAT );
                                cs.registerOutParameter(3,java.sql.Types.VARCHAR );
                                cs.registerOutParameter(4,java.sql.Types.VARCHAR );
                                cs.registerOutParameter(5,java.sql.Types.INTEGER );
                                //set value to IN param
                                cs.setInt(2,no);
                                //call PL/SQL function
                                cs.execute();
                                //gather results from OUT PARAMS and RETURN PARAM
                                System.out.println("emp desg::"+cs.getString(3));
                                System.out.println("emp name::"+cs.getString(4));
                                System.out.println("deptNo::"+cs.getInt(5));
                                return cs.getFloat(1);
                        }//execute(-)

                }//anonyomous inner class
                 ); //method call
                        System.out.println("emp salary ::"+sal);
        }
}
```