Hibernate
==========
=> pre requisites :: Core java + basic of JDBC + Basics of SQL + lots of common sense
=> Only Spring learning is great ... but learning spring with hibernate is super super great..
To learn and to use Spring ORM ,spring Data jpa modules we need Hibernate knowledge a lot
=> Here we are going learn both Hibernate and JPA
=> Hibernate 4.x /5.x and Intro to 6.x
=>duration :: 75 to 85 Sessions (6/7 days in a week)
=>12+ tools will be covered as common topics like log4j, maven,gradle, svn,git and etc...
=>Course fee : 2000
=> Admin person details :: Mr. srikanth.
phone :: 6302968665
=> Fb Group :: nataraajavaarena
(https://www.facebook.com/groups/nataraajavaarena)
=> daily 1.35 to 1.45 mins..
=> email id :: nataraajavaarena@gmail.com
=> BatchCode :: NTHB915
=> U can learn Spring ,hibernate, Adv.java courses parellel.

What is PErsistence?

=> The process of storing and managing data for long time is called persistence... We use support of SEcond memory devices like HDD, CD,DVD , Thumb drives and etc.. for persistence..
=> Storing data in the Application variables ,objects does not come under persistence..beccoz they allocate memory in the RAM that to only during the Application execution... Once the execution is over ... they will be vanished.. So data can not be used across the multiple executions of same App or diff Apps...
=>To overcome the above problem take the support persistence i.e write application data (both inputs and outputs) to Secondary Memory Devices like HDD and etc.. with the support of files and DB s/w..

Important terminologies
==========
a) Peritence store b) PersistentData c) PErsistence operations d) Persistence logic
e) Persistence technology /Framework/Tool

Persistence Store
==========
=> It is the place or store where data will be saved and managed for long time
eg:: Files , DB s/ws [oracle,mysql,postgreSQL,...]

Persistent Data
==========
=>The Data of Persistence store is called PErsistent Data...
eg: file info , DB tables and their records

PErsistence operations
==========
=> Insert,update,delete and select operations performed on Persistent Data ...
-> These are also called as CURD/CRUD/SCUD operations..

CURD/CRUD	SCUD
==========	=====
C-->create (Insert)	S-->Select
U-->Update (modify)	C-->Create
R ---->Read (select)	U-->Update
D---->Delete (remove)	D---->Delete

Persistence logic
==========
=> The logic that is written to perform CURD/CRUD/SCUD operations..
eg: to stream logics (like serialization ,DeSerialization),
jdbc code
hibernate code,
spring jdbc code,
spring orm code,
spring data code (hot cake)
and etc..

java learning
|--> language (Core Java)
|--> Technologies (adv java)
|-->frameworks (spring, hibernate, ...)

Persistence Technology/Framework
==========
=> The technology /framework/tool using which we develop persistence logics
JDBC (Technology)
hibernate (framework /Tool)
spring jdbc/Spring ORM/Spring Data (framework)

=>Persistence Technology/framework /Tool is given for developing persistence logics to perform Persistence operations on the persistent data of persistence stores...

Data Storage Technologies/software Vs DataAccessTechnologies/Frameworks

The s/w which store and manage data is called Data storage software
eg: Db s/w (oracle,mysql and etc..)

The s/ws using which we can access and manipulate the data of DataStorage software is called Data Access Technology/framework/software
eg: jdbc , hibernate, spring jdbc and etc....

Java app ---(I/O streams (like serialization , Deserialization))---> Flat Files (The file that natral to all OSs, languages) (text file, xml files..)

Java App ---(jdbc , o-r mapping)---> RDBMS DB s/w (oracle ,mysql and etc..)

Java App ---(Jdbc + SQL)/)---> OODBs/w OOB s/w (wakanda , Object Store ,OOABA and etc..) (Object Data base softwares)

Limitations with flat files as persistence stores
==========
(a) No Security
(b) No constraints
(c) No SQL support
(d) getting data with multiple conditions is very complex
(e) performing update and delete operations is very complex
(f) No realationships
(g) Merging and comparision of data is very complex...

To solve these use DB s/w s like RDBMS,OOB s/ws...

note:: We generally use Files as PErsistence stores only in Memory Critical Apps like mobile Apps, mobile games, Desktop games and etc...

OOB s/ws or OOOB s/ws
==========
=>These Db s/w can store software objects directly in Db s/ws as column values.. and so useful while working OOPs languages becoz in OOP we get every thing as objects...

public class Student{
private Int sno;
private String sname;
private String sadd;
private float avg;
//getters && fetters (methods)
...
...
...
}

Student st=new Student(101,"raja","hyd",56.55f);

st --> Student class obj
101
raja hyd
56.55

st --> OOB s/w or OODB s/w
Student (db table)

---
st

pros of OOB or OOOB s/ws
==========
=> In Object oriented Phgramming we get objects every where and we store them directly in Db s/w i.e there is no need converting simple values to objects and objects to simple values while doing persistence operations
=> All CURD opeations can be directly on the objects
=> No need of SQL ... We can these Db s/w as NOSQL Db s/w..

cons of OOB s/ws or OOOB s/ws
==========
=> Does not allows keep data/objects in relationships.
=> The language in which u r creating software object should have compitability with OOB s/w...
(There is tight coupling b/w Object oriented Db s/w and the language in which the software objects are created)
=> Against of Normalization rule1/form1 (Do not store multiple values in single col.. Always store multiple values in multiple cols)

=> These are not industry standard Db s/ws ...
and etc..

Different types of DB s/ws
==========
(a) HDBMS Db s/ws (hierarchal DBMS)
(b) NDBMS DB s/ws (Network DBMS)
(c) RDBMS Db s/ws (Relational DBMS)
(d) OODBMS DB s/ws (Object Oriented DBMS)

=> People had not shown any intreset on OOB s/w becoz of RDBMS DB s/ws popularity.. and more over they are very much habituated with RDBMS Db s/w like oracle,mysql and etc...

Java App ---(jdbc , (like hibernate) o-r mapping)---> RDBMS Db s/w (oracle ,mysql, postgre SQL and etc..)

Limitations of JDBC to develop PErsistence logic

(a) We develop JDBC Persistence logic using SQL Queries ... Since SQL Queries are DB s/w Dependent Queries we can say JDBC PErsistence logic is DB s/w dependent PErsistence logic and JDBC Code /persistence logic is not portable across the multiple DB s/ws..
note:: changing DB s/w in the middle of development or production or after development and before release is very complex... (jdbc persistence logic is not portable across the multiple DB s/ws)

(b) JDBC Persistence logic /code is having boilerplate code problem..
The code that repeates in multiple parts of App/Project either with no changes or with minor changes is called boilerplate code.

jdbc code in Java App
==========
=> Load jdbc driver class to activate JDBC driver s/w
=> Establish the connnection with Db s/w
=> create Statement object
common logics

=> Send and execute SQL Query in DB s/w
=> Gather results and process the results..
Application specific logics.

=>Handle the exceptions
=> close jdbc objs including jdbc connection
common logics
==> comming logics are nothing boilerplate code...

(c) JDBC Code throws SQLException which is checked exception..
|--->To Mandatory to handle. class
|---> Exception propagation to next class/caller is not possible if we catch and handle the exception.

public void m1){
try{
...
...//jdbc
...
}
catch(Exception e){
..
}
}

=>UnChecked exceptions are good becoz they are optional to catch and handle , they support exceptin propagration by default..

(d) in JDBC , for all problems we have Single SQLEception class... i.e we do not Detailed Exception Classes for Different Problems..

(e) Whilse closing jdbc objs including jdbc connection obj we need to analyze a alot...
-->closing jdbc objs at the end of try block (or) in the catch blocks or in both places is bad practice.. closing in finally block is good pratice

try{
...
... //jdbc code
...
}
catch(...){ ...}
catch(...) { ... }
finally{
try{
rs.close();
st.close();
con.close();
}
catch(...){ ...}
}

bad code.. becoz closing while said jdbc obj, if exception raised the control goes catch block and we are not giving fair chance to close other jdbc objs

finally{
try{
if(rs!=null)
rs.close();
}
catch(...){ ... }
try{
if(st!=null)
st.close();
}
catch(...){ ... }
try{
if(con!=null)
con.close();
}
catch(...){ ... }
}//finally

Good pratice to close jdbc objs

//if we use try with resource for jdbc code , the jdbc objs will be closed automatically at the end of try i.e we need not to close them manually .. using finally block..

try(Connection con=DriverManagr.getConnection(,,,);){
try(Statement st=con.createStatement()){
try(ResultSet rs=st.executeQuery("SELECT * FROM STUDENT ")){
... //logic to process the ResultSet object
...
}//try3
}//try2
}//try1
catch(...){ ....}
catch(...) { ....}

note:: doing TxMgmt by calling con.commit() , con.rollback() is bit complex in try with resource.

f) JDBC ResultSet objs are not Serializable objs i.e we can not send their data them over the network directly.. i.e we need convert ResultSet records to other values in order to send them over the network..
note:: The process converting obj's data into bits and bytes is called serialization... So these bits and bytes can be sent over the network or to file.. Serilization is possible only on serializable objs nothing but the objs whose classes are implementing java.io.Serializable()

g) Java is object oriented programming language.. but we can not use objects directly in jdbc persistence logic i.e we need to convert objects data into simple values in order use them in SQL queries as inputs becoz SQL queries can not take java objects directly as input values..

public class Student{
private int sno;
private String sname;
private String sadd;
//3 param constructor
...
//getters & getters
....
...
}

jdbc code
==========
Student st=new Student(101,"raja","hyd");

PreparedStatement ps=
con. prepare-Statement("INSERT INTO STUDENT VALUES(?,?,?)");

int no=st.getSno() ; String name=st.getSname();
String addrs=st.getSadd();
ps.setInt(1,no);
ps.setString(2,name);
ps.setString(3,addrs);

Here "st" object data is converted into simple values.... in order use in jdbc code..

h) There is no proper Transaction Management support
=>The process combining related operations into single unit and executing them by applying do everything or nothing principle is called Transaction Managment.
eg: transfermoneny with withdraw, deposite operation
Student registration with inserting record to student db table and inserting record to course db table.

two type Transaction Managments
(a) Local Transaction Management
--> HEre all operations take place on single DB s/w
eg: transfer Money operation b/w two accounts of same bank.
(b) Global Transaction Management
--> Here different operations take place on different DB s/ws..
eg:: transfer money operation b/w two accounts of two diff DB s/ws..
note: in jdbc there is good support Local Tx mgmt , But there is no proper support for Global Tx mgmt

i) JDBC supports only positional parameters (?) in the SQL query ... and it does not named parameters ... (name for each parameter) :<name> syntax of named parameter
insert into student values (?,?,?,?,?,?,?)
positional params
insert into student values(:no,:name,:addrs,:age,:avg,:m1,:m2,:m3)
named parameters

j) Strong SQL knowledge is required to work with JDBC code..

k) while developing persistence logics in JDBC we can not enjoy inheritence, polymorphism composition and etc.. features oops becoz jdbc does not allows to objs as input values in the SQL queries..

l) JDBC does not supporting versioning and timestamp features as built-in features.. i.e we need to write lots lines of code to implement those features...
Versioning :: keeps track of how many times the record is modified..
timestamping :: keeps track of when record is inserted and lastly modified (date,time)
and etc...

=>To solve most of these problems ... prefer writing persistence logic using o-r mapping env.. taking support hibernate,iBatis , toplink ,eclipse link and etc... ORM frameworks /Tools..
note :: spring ORM ,Spring data interally uses one or another ORM framework..

What is o-r mapping?

Short anwser :: It is all about mapping /linking classes with Db tables

Lengthy Answer :: The process of mapping java classes with DB tables , member variables of java classes with columns of DB tables and representing the records of DB tables with the objets of java classes having synchronization b/w them is called O-r mapping..
==>Here the word synchronization is used having english meaning .. i.e the modifications done in Objects will reflect to DB table records and vice-versa.

Student.java (Entity class/PErsistence class/Model class/Domain class)
==========

```
package com.nt.entity;
public class Student{
  private int sno;
  private String sname;
  private String sadd;
  private float avg;
  //settters && getters
   .....
   ....   4 setters
   ....   & 4 getters
}
```

o-r mapping cfgs (xml ,annotations)

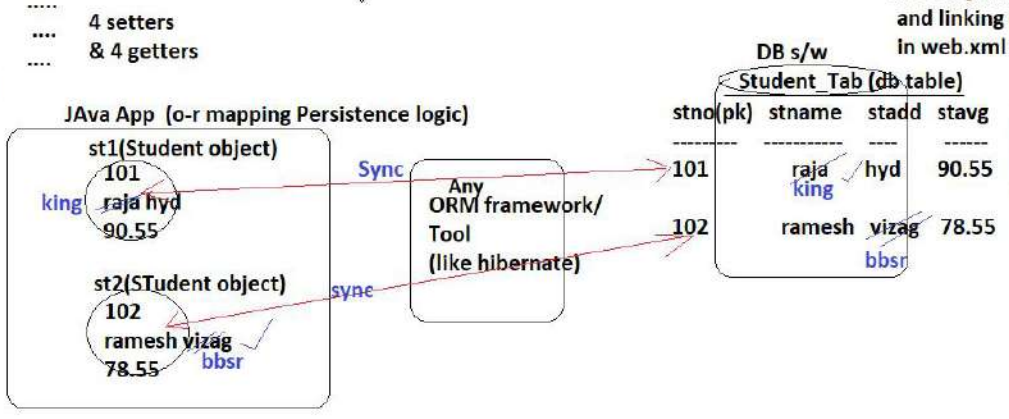
Student (java class) <-------> Student_Tab (db table)

java class properties: sno <----> stno; sname <--->stname; sadd <---> stadd; avg <---->stavg

(Db table cols)

What is Configuration?
==========
=> Providing info about about various comps like files/classes /code to underlying server/cotnainer/ jvm/framework and etc.. to give instructions is called configuration..

note1:: java class will be treated as servlet comp and linking with url based on its configuration done in web.xml file.. or using annotations..



-> Java class becomes Hibernate entity class and will be mapped with DB table based its configuration done hibernate cfg related xml file or using annotations..

Here we can perform all persistence operations through objects by persistence instructions to ORM framework like Hibernate in the form of objects with out using anySQL Queries
This makes o-r mapping persistnce logic Db s/w independent persistnce logic..
But the ORM framework like hibernate internally uses JDBC code +SQL Query to complete the Persistence operations..

In O-rMapping Persistence logic development , the objects of Entity classes like are actors playing different characters..
note: After deleting record representted by object.. the object will not be deleted more over we can use that object for other persitence operations on different records of same table.

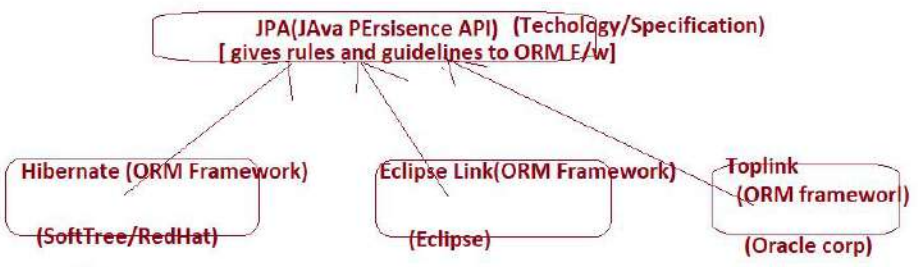
o-r mapping
==========
Saving object means writing object data to db table as record
updating object means modifying the db table record represented by the object
Loading object means selecting the record into the object from the DB table
Deleting object means deleting the record represented by the object..

There are multiple frameworks/Tools/Technologies to develop o-r mapping persistece logics they are called ORM framework/toos/Technologies

EJB Entity Bean (Technology) ---------->from Sun (oracle corp) (outdated)
Hibernate (Tool/Framework) ----------> SoftTree /Jboss/RedHat (1)
iBatis ----------> from apache (3)
Toplink ----------> from oracle copr
Eclipse link ----------> from eclipse (2)
OJB ----------> from apache
open JPA ----------> from Sun Ms/Oracle corp
and etc..

All java based ORM frameworks are given based on JPA specification rules and guidelines

JPA --->Java Persitence API



When to use JDBC and when to use Hibernate(ORM Framework/Tool)?

Offline App :: The app that runs with out human intervention (or) enduser interaction
eg:: census Application (Population count)

School teacher -----> home ----> gathers details on paper ---> submits to mandal/taluka head quaters ---> DEOs converts info to Excel sheets ---> sheets uploaded central GOVT App ---> takes excel sheets ---> validates them, corrects them , categorizem and stores in to Db s/w..

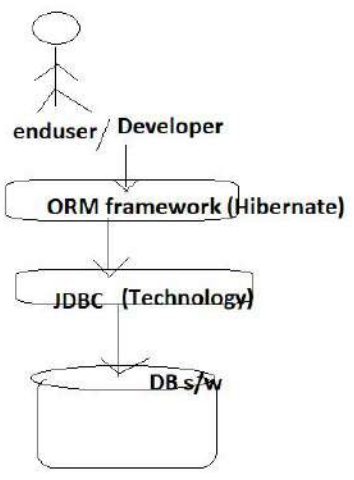
Online App :: The app that runs with human intervention (or) enduser interaction

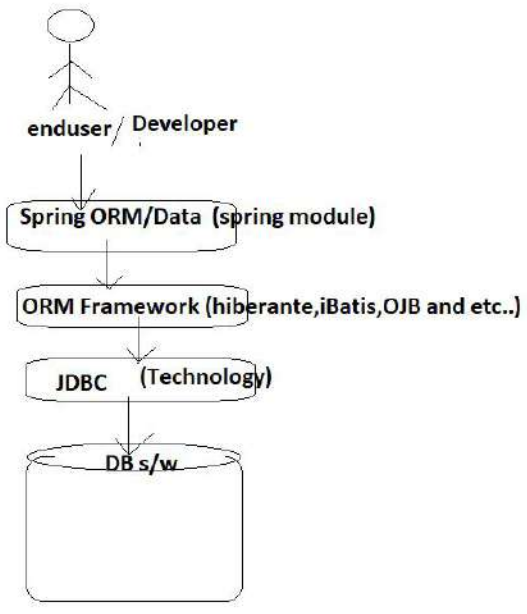
eg: flipkart.com , amazon.com , nareshit.com and etc...

=>Online or offline Apps when they are getting huge amount of data batch by batch for processing then use JDBC .. becoz it creates single ResultSet object to maintain huge amount of records.. if we use hibernate here huge no.of objects will be created to represent and process huge amount of records then Project may crash..
eg:: Census App ,Flipkart App

==>Online or offline Apps if we are getting little amount of data to process at a time then go for hiberante.. (ORM framework) becoz to represent small amount of records small no.of objects are sufficient and more ever we can get benifit of other features of ORM frameworks like DB portability and Time stamping ,versioning, caching and etc...
eg:: nareshit.com

note:: youtube data , Google data, fb data and etc.. are BigData.. they are beyond storing and processing capacity of DB s/w... So go for BigData frameorks like Hadoop ,spark anetc...





coclusion:
=========
=>Instead using plain jdbc ,prefer using spring jdbc that allows us to devleope jdbc style SQL Queries based DB s/w dependent PErsistence logic by avoiding boilerplate code of jdbc..
spring jdbc needs plain jdbc knowledge..

=> Instead of using plain ORM frameworks/Tools like hbiernate prefere using Spring ORM to develop Objects based Db s/w independent Persistence logic with out SQL Queries..
This also avoid boiler plate code relaed plain ORM framework programming.
spring ORM needs plain ORM tool/framework) like hibernate knowledge..

=> Instead using Spring JDBC, Spring ORM use spring Data which takes care of both commons logics (boilerplate code) and regularly used Application specific logics (nothing but CURD operations persistence logic)

Spring DAta sub modules
==========
=>Spring Data JDBC ---> generates common logics and specific logics by using jdbc internally (so jdbc knowledge is required here)
(SQL DB)
=>Spring Data JPA ----> generates common logics and specific logics by using ORM framework (hibernate) internally .
(so hibernate knowledge is required here)
(SQL DB)
=>spring Data MongoDB (NoSQL DB) ---> generates common logics and specifics required for interacting with MongoDB s/w.
=>Spring Data Neo4j
=>spring DAta Cassendra
and etc...

DB s/w
- SQL Db s/w (Structured Data) (oracle,mysql,postgreSQl and etc..)
- No SQL DB s/w (UnStructured Data) (MongoDB , Cassendra , neo4j and etc..)

How to choose Data storage Technology and DataAccess Technology in Java ?
(Persistence store) (Persistence technology/framework/tool)

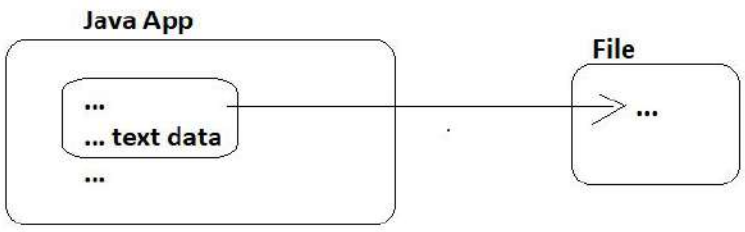
==========

Scenario1
========

=>App is small scale App and generating little (kb) amount of text data to persit

PErsistnece store :: files
Pesistence logic env :: IO Streams

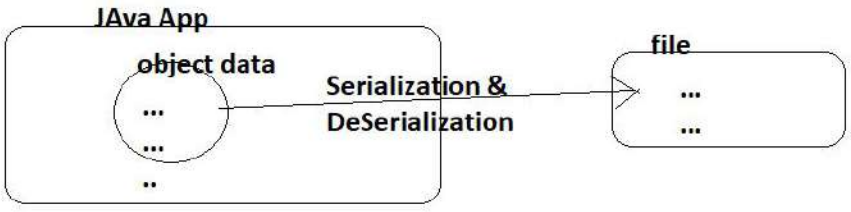
eg: Mobile Games, Mobile Apps, Desktop Games



Scenario2
=========

App is small scale App and generating objects based data (little data (kbs)) to persist.

Persistence store :: File
Persistence logic env.. :: IO Stream (serialization and Deserilization)

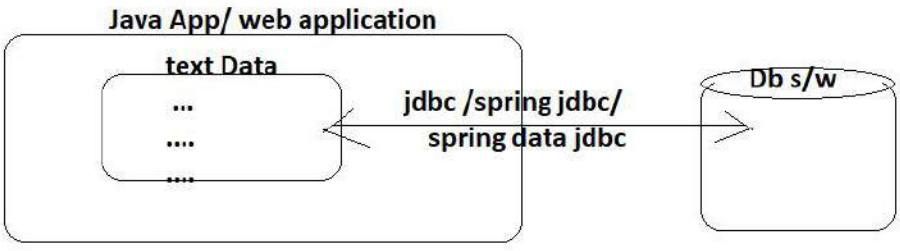


Scenario3:

App is medium scale App like web application and generating medium sized data .. (MB/GB) to persist as text data...

eg:: nareshit.com
matrimony websites

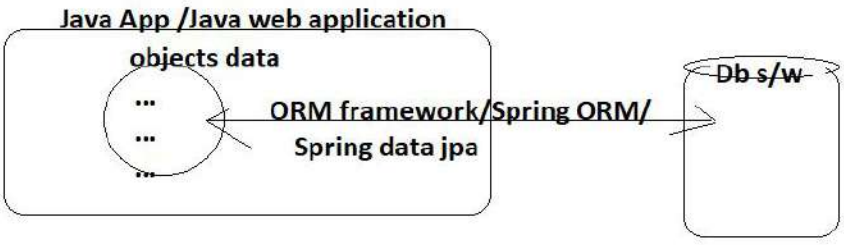
pesistence store :: RDBMS DB s/w
Persitence logic env :: jdbc /spring jdbc /spring data jdbc



scenario4:
=========

App is medium scale App like web application and generating medium sized (MB/GB) objects data based Data .. to persist

Persitence store :: RDBMS Db s/w
Peristence logic env :: ORM Framwork(Hibernate)/ Spring ORM/Spring Data jpa



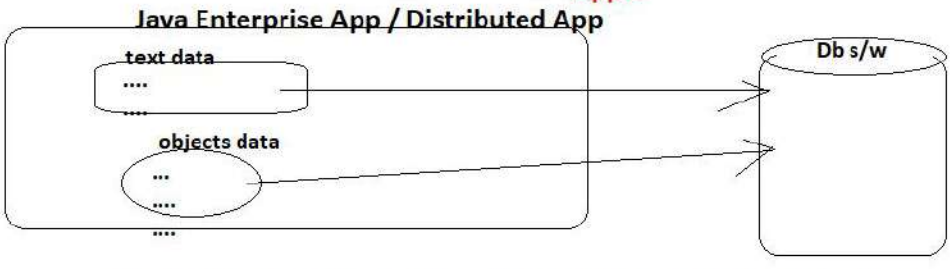
Scenario5:
========

App is enterprise App (complex App)/ Distrbuted App (App having different types remote clients) generating large amounts data (GB/TB) of data to persist in the form of text data.. or objects Data

Amazon.com
flipkart.com

These are complex app and also allows requests from diff types of remote clients like Android MobileApp , IOS mobile App,browser UI(html,Css) and etc.. So these are called distrobuted Apps or enterprise Apps.

PErsistence store :: RDBMS Db s/w
Persistence logic env :: jdbc/spring jdbc/sprjg data jdbc (text data)
hibernate/Spring ORM/Spring data jpa

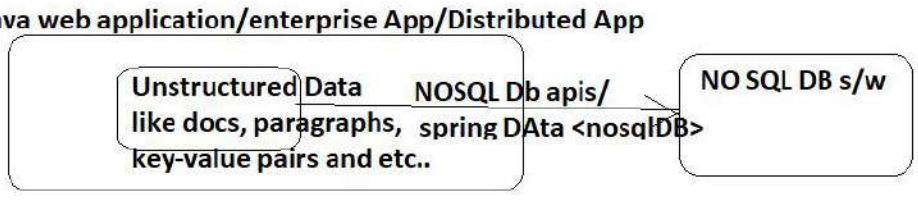


Scenario6

Application is medium scale or largeScale App (web application or enterprise App /Distributed App) generating unstructured Data .. to persist..

Persitence store :: NO SQL Db s/w is like MongoDB
Persistnece env :: use specific NO DB apis or spring data <NOSQL module >

note:: Unstructured Data means .. The data that changes dynamically some times 4 values , some times 10 values , some times 30 values.. for same employee type details



Scenario7:
==========

Data is huge amount of text data that is beyong processing and storage capacity given large scale Distributed or enterprise Apps having PB/XB/YB and etc.. to persist ..

eg:: fb data ,google data, youtube.data..

Persistence store :: BigData framework(Hadoop/spark)
PErsistence logic env :: BigData frameworks (....)

PB -->peta bytes
XB -->Xenota bytes
YB --->Yotta byte
and tc..

Employee Info
==========
name :: raja
addrs :: hyd
email : raja@rani.com
photo :: E:\images\raja.jpg
resume :: e:\resumes\raja.doc

note:: SQL DB ,NOSQLDB ,BigData softwares do not store audio,video ,images files directly in records.. just they maintain their locations as text data...

=>Instead of placing multiple logics in a single file /class .. it recomanded to place in multiple layers and make them talking with each other..
=>A layer is logical partiation of the application that represents specific logics..
=> A Layer can be one or more classes/files representing Cetatains logics..
eg:: Persistnece Layer/Data Access Layer , Service Layer /Business Layer ,presentation layer , Integration layer and etc..

=> Advatanages of layered Application development

(a) Multiple logics will be placed in multiple layers ,, This gives clean seperation b/w logics..
(b) The modifications done in one layer logics does not effect other layers..
(c) Maintence and Enhancement of the Project becomes easy..
(d) Parellel devleopment is possible , So the productivity is good
(doing more work in less time with out comprimising in quality gives good productivity)
(e) It is Industry standard
and etc..

=>The main logic of App that deals with calcuations, analyzation is called B.logic/Service Logic

=>CURD Operations on Persistence store are called Persistence logic

=>MVC Architecture is very popular to develope layered Applications in java...
Gives blueprint or plan or design to develope the applications,,

M---------->Model Layer (service/business layer (B.logic) + PErsistence Layer(Persistence logic) ----> It is like Accounts officer
|----> represents Data... DAO layer
V -------> view Layer ---> represents PResentation logic ---> It is like beautician

C -------->Controller Layer -----> represents Monitoring/Controlling logic ----> It is super viser

MVC is insdustry's defacto standard to develop java based Layerd Applications..

Combination of various Technologies /framewroks of java to develop MVC Applications
==========

jsp --------> Servlet ----------> Java class with java bean (jdbc) ----------> DB s/w
(v) (c) (M) (b.logic & Persistence logic)

The java class that contains b.logic is called service class

jsp ----------> servlet ----------> service class ----------> DAO class (jdbc) ----------> DB s/w
(v) (c) (m--->b.logic) m--> pesistence logic

The java class that contains persistence logic is called DAO calss

jsp ----------> servlet ----------> EJB session Bean ----------> EJB Entity Bean (ORM) ------> DB s/w
(v) (c) M---> b.logic M-->Persistence logic

Struts ----------> EJB Session Bean ----------> EJB entity Bean (ORM) --------> DB s/w
(v) (c) M---> b.logic M-->persistence logic

Struts ----------->EJB Session Bean ----------> DAO class (jdbc) (java class)----> Db s/w
(v) (c) M--->b.logic M--->persistence logic

Struts ----------> EJB SessionBean ----------> Hibernate (ORM) ----> Db s/w

Struts ----------> spring Core +spring AOP ----------> hiberante ----------> DB s/w
(v) (c) M---> b.logic M---> Persistence logic

flipkart.com
Spring MVC ----------> spring Core + spring AOP -----> spring jdbc/spring ORM/Spring data--> Db s/w /hibernate
(v) (c) M---> b.logic M---> Persistence logic
(Spring Rest)

External Services / Ditributed Comp/Service
WebService (SOAP/Rest)
(Paypal comp)

Different technologies /frameworks to develop Persisnce logics
==========

JDBC
Hibernate *
Toplink
EclipseLink *
JDO
spring jdbc *
spring orm *
spring data *
iBatis
EJB entity Beans
and etc..

Different Technologies and fremworks to develope service logics/b.logics
==========

java classes
spring core , spring JEE , spring AOP *
EJB SEssion Beans
and etc..

Different Technologies and frameworks to develop Exsternal services/ Distributed Comps
---

RMI
EJB (outdated)
CORBA

WebSerivce (latest) * SOAP :: Simple object Access Protocol

SOAP based WebServices (very less used)
---
jax-rpc
jax-ws
axis *
apache cxf and etc..

RESTfull webservices jax-rs (metro)
---
spring rest (good) *
rest easy
jersy

Different Technologies to devleop controller
==========
Servlet *
Servlet Filter

Differente technogies /tools to devleop view comps

->html ,jsp (*)
->css *
->bootstrap
->java script
-> angularjs
->angular *
->volocity
->themelfy
->ajax
-> Reactjs *
-> jquery
and etc..

diff frameworks to devleope view and controller logics togather
==========
Spring MVC (*)
Struts
JSF
ADF natarazjavaarena --->FB group
Webwork
and etc..

Hibernate
==========
type :: Java ORM framework/Tool or Implementation s/w of JPA Specification

JPA :: Java PErsistence API.

Vendor :: SoftTree / RedHat / JbossVendor
(A division of RedHat)

version :: 6.x (under development) (compitable with java8 or 11)
5.x--> 5.4.18 (stable version) (compitable with java 8 or 11)
(|-->Development based on jpa2.2 to specification)

JVM based languages are
scala, groovy ,kotlin ,java and etc..

creator :: Mr. Gavin king

Open Source

To download s/w :: download as zip file from www.hibernate.org website..
https://sourceforge.net/projects/hibernate/files/hibernate-orm/5.4.18.Final/hibernate-release-5.4.18.Final.zip/download

reference Books :: Persistence with Hibernate by Gavinking
--> Hidernate supplied ref docs +api docs + class room content..

To install s/w :: Extact zip file (hibernate-release-5.4.18.Final.zip)

=>use <hibernate_home>\changelog.txt file to know which version is released in which year..

=> <Hibernate_home>\documentation folder gives api docs and ref docs...

=><Hibernate_home>\lib folder gives libraries (jars)

=><Hibernate_home>\project folder gives
-> utility files ,classes
-> source code
-> sample Apps, projects
and etc..

Defination: non-invasive
Hibernate is an open source ,light weight , Portable and Integrated java based ORM framework or Tool that is given based on JPA Sepcification rules and guidelines to develope objects based o- r mapping persistence logics.

open source
==========
=>along with hibernate installation we get its source code.. and more over hibernate framework free s/w .. This hibernate as an Open source framework.
makes

Light weight
==========
Hibernate is ligtht weight becoz
(a) Hibernate f/w is given as zip file having less size (In mbs)
(b) To develope and execute Hibernate Persistence logic we just need
JDK/JVM and hibernate libraries(jars) , i.e there is no need of arranging the heavy weight servers / containers

Portable
(c) By creating object for "org.hibernate.cfg.Configuration" class we can activate /bootstrap hibernate any where..
==========
=> Using hibernate we can develop persistence logic as objects based persistence logic with out using any SQL Queries support.. This makes hibernate persistence logics DB s/w independent persistence logics and also portable across the multiple DB s/ws...
note:: This feature helps to change the DB s/w in the middle of the development and also in the middle of the production.

What is the difference b/w inavise programming and non-invasive programming?

Invasive Programming
==========
=> HEre the classes of App development are tighly coupled with underlying framework/ Technology apis.. I.e The App classes should implement /extend from underlying framework /Technology api interfaces/classes..
=>Here we can not move App classes to other frameworks /technology for execution..

eg:: struts 1.x (framework) ,Servlet (Technology)

joining in the company with bond

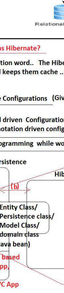
non-invasive Programming
==========
=>HEre the calsses of App development are loosely coupled with underlying framework/ Technology apis ie. App classes do not implement/extend from underlying framework or Technology api interfaces/classes...
=> Here can move App classes to other frameworks /Technologies for execution becoz they are developed as ordinary /simple java classes..

eg:: hibernate (all versions) , spring all versions , Struts 2.x and etc...

joining in the company with out bond

Integratable
==========
=>we can place hibernate persistence logic in different types of java applications to make them interacting with Db s/w like we can place hibernate persistence logic in standalone JAva app, in web applications, in distributed Apps and etc... This makes hibernate persistence logic as easily integratable logic with diff java technologies and frameworks and also in different types of Apps..

High level architecture of Hibernate
==========



To Common features of hibernate that Implemented based JPA Specification use JPA based programming..

To use direct features of hibernate which are not there in JPA specification then use Hibernate Native API...

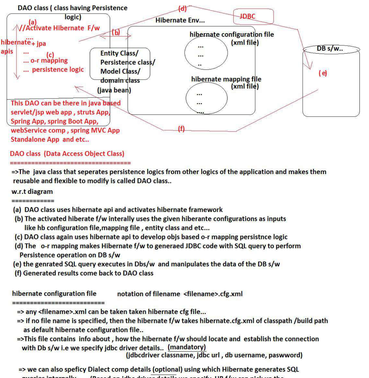
Why the Hibernate is named as Hibernate?

It is given based on hibernation word.. The Hibernate Programming represents DB table record through objects and keeps them cache .. looks like hibernation.. So this framework is called as hibernate.

Hibernate Configurations (Giving instructions/information to underlying container /Server /Framework/JVM)

1. xml driven Configurations (old --> used only in maintenance projects)
2. annotation driven configurations (Latest --> used in all projects)

Components in Hibernate Programming while working with Xml driven cfgs



DAO class (Data Access Object Class)
==========
=>The Java class that seperates persistence logics from other logics of the application and makes them reusable and flexible to modify is called DAO class..

w.r.t diagram
==========
(a) DAO class uses hibernate api and activates hibernate framework
(b) The activated hibernate f/w internally uses the given hiberante configurations as inputs like hb configuration file,mapping file , entity class and etc..
(c) DAO class again uses hibernate api to develop objs based o-r mapping persistnce logic
(d) The o-r mapping makes Hibernate f/w to generaed JDBC code with SQL query to perform Persistence operation on DB s/w
(e) the genrated SQL query executes in Dbs/w and manipulates the data of the DB s/w
(f) Generated results come back to DAO class

hibernate configuration file notation of filename <filename>.cfg.xml
==========
=> any <filename>.xml can be taken taken hibernate cfg file...
=> if no file name is specified, then the hibernate f/w takes hibernate.cfg.xml of classpath /build path as default hibernate configuration file...
=>This file contains info about , how the hibernate f/w should locate and establish the connection with Db s/w i.e we specify jdbc driver details... (mandatory)
(jdbcdriver classname, jdbc url , db username, password)

=> we can also specfiy Dialect comp details (optional) using which Hibernate generates SQL queries internally ... (Based on jdbc driver details we specify HB f/w can pick up the Dialect comp details dynamically..so it is optional to specify)

=> This file to use to give common instructions to hibernate f/w like enabling autoCommit, enabling caches , enabling SQL query show and etc.. (optional)

=> Hibernate cfg file generally contains
a) jdbc properties (jdbc driver details) (mandatory)
b) hibernate properties (like dialect info, showing sql, formatting sql, auto generation DDL and etc..)
c) Hibernate Mapping file names..
(mandatory)
(optional)

=> we given same info by using hibernate.properties file... but that as become legacy..(old)

=> This Hibernate configuration file information can be given as key value pairs where keys are fixed and values will be changed based on DB s/w , jdbc driver and hibernate s/w names and versions we are using..

eg::

```
## Oracle
hibernate.dialect     org.hibernate.dialect.Oracle8iDialect
hibernate.dialect     org.hibernate.dialect.Oracle9iDialect
hibernate.dialect     org.hibernate.dialect.Oracle10gDialect
```

we use one of these but optional

```
hibernate.connection.driver_class  oracle.jdbc.driver.OracleDriver
hibernate.connection.username  ora
hibernate.connection.password  ora
hibernate.connection.url jdbc:oracle:thin:@localhost:1521:orcl
```

jdbc properties (mandatory)

(collection from <Hibernate_home>\project\etc\hibernate.properties file)

=>We develop this configuration file on 1 per DB s/w basis.. If our App is talking with 1 Db s/w .. then we take 1 hibernate cfg file... otherwise we take multiple configuration file..

note:: dialect comp is internal part of hiberante f/w having capability to Generate SQL Queries Internally based on the DB s/w we use..

In hibernate.cfg.xml (Partial Code)
==========

```
<property name="hibernate.connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
<property name="hibernate.connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
<property name="hibernate.connection.username">system</property>
<property name="connection.password">manager</property>

 ....
 .....

<mapping resource="com/nt/entity/Employee.hbm.xml"/>
```

hibernate mapping file name

Entity Class /Model class/Persistence Class/Domain class
==========
=> It is just bean class whose objects holds persistable Data (to be inserted to DB) or Persistent data (colleted from DB)
=> It will be taken on 1 per Db table basis
=> while writing objects based o-r mapping persistence logic .. we use this objects to represent DB table records and to perform Persistence operations
=> These are non-invasive classes .. that means these classes need not implement or extend from hibernate api interfaces/classes.
=> We can design these classes using all java oop features like inheritence ,composition polymorphism and etc..

DB table in Oracle
==========

```
Employee
|---->eid (n)
|---->name (vc2)
|---->esalary (n)
|---->eaddrs (vc2)
```

```
//Entity class
public class Employee  implements Serializable{
                                  Recomanded.. (optional)
  //bean properties
  private int eid;
  private String ename;
  private float esalary;
  private String eaddrs;

  //setters && getters

  public void setEid(int eid){
    this.eid=eid;
  }
  public int getEid(){
    return eid;
  }
  ...
  ...
  ...
  ...
}
```

mapping
note:: when we are Entity classes with DB tables ,there must be compitability b/w Entity classes properties count and type with Db tables colx type and count.

hibernate mapping file
==========
=> any <filename>.xml can be taken as hibernate mapping file.. but every file mapping name must be specified in hibernate cfg file using <mapping resource="---"> tag.

=> This file contains diff o-r mapping cfgs ... like mapping Entity classes with DB tables and mapping properties of Entity classes with the colums of Db tables...
=> Recomanded notation of this file name is <filename>.hbm.xml
=> There is no default file name for hibernate mapping file..So Every file name must be specified in configuration file..
=> we generally take these mapping files on 1 per Db table basis.

In Employee.hbm.xml (sample mapping file) (partial Code)
==========

```
<class name="pkg<.Employee"  table="EMPLOYEE">
  <id name="eid"  column="EID"/> //for PK colum related property
  <property name="ename"  column="ENAME"/>
  <property name="esalary"  column="ESALARY"/>
  .....
</class>
```

for other columns related properties cfg...

Entity classes ---> 1 per Db table basis
HB Mapping files ---> 1 per Db table basis
HB Cfg files ---> 1 per Db s/w basis

Naming Conventions
==========

```
Oracle DB s/w
=============
Employee
Student
```

Db tables

HB configuration file name :: hibernate.cfg.xml
Entity classes names :: Employee ,Student
mapping file names :: Employee.hbm.xml
Student.hbm.xml

```
Employee emp=new Employee();
emp.setId(1001);
emp.setName("Raja");
emp.setEaddr("hyd");
emp.setSalary(9000);
```

```
SF$;save(emp);
```

gives save object persistence instruction to Hiberate f/w. (insertint record)

hibernate session obj is no way related Servlet Session object...
hibernate Session object is con++ (i.e con object + xml files info)

Features of Hibernate framework

==========

=> Allows to devleop Objects based DB s/w indenpendent o-r mapping persistence logic with out any SQL Queries.

=> Avoid jdbc boilerPlate code...

=> Supports both Xml driven , Annotation driven cfgs..

=> Hibernate Light weight framework.. we can bootstrap/activate HB by just creating object for "Configuration class".

=> The entity classes of hibernate are non-invasive i.e they can moved and used in other frameworks

=> All exeptions in Hibernate are unchecked exceptions.. So no need of catching and handling method.. more over.. they automatically exception Propagration..

=> Supports both Local and Global Transactions.. and we can change from Local to Global and vice-versa .. with out toughing hibernate persistnce logic.. by just modifying xml file.

=>Supports Versinoning and timestamping..

=> allows use all java features while developing entity classes... like inheritence ,composition, polymorphism , association and etc...

=> Auto Schema Generation (DB table creation)

=> supports both single row and bu lk row operations

|--->HQL/JPQL HQL --->Hibernate Query Language
JPQL ---> Java Persistence Query Language | DB s/w independent

|

|--->NativeSQL (Original SQL queries)

|--->Criteria API /QBC (Query By Criteria)

=>Allows to call PL/SQL procedures and functions... (from hibernate 3.x)

=>Supports Lazy Loading .. (Delays the hittin process of DB s/w)

=>Gives generators(12+) to generate PK col values /Identity values dynamically

=> Gives callback interfaces to write plain jdbc code.. by getting jdbc objs..

and etc...

How does ORM framework /hiberante performs synchronization b/w Objects and DB table rows?

```
//Entity class
public class Employee{
   private int eno;
   private String ename;
   private String eadd;
   private float salary;
   //setters & gettrs
      ...
      ..
   }
```

properties of Entity class

Employee.hbm.xml (hibernate mapping file)

pkg.Empoyee(class) <====> EMPLOYEE (db table)

Id property

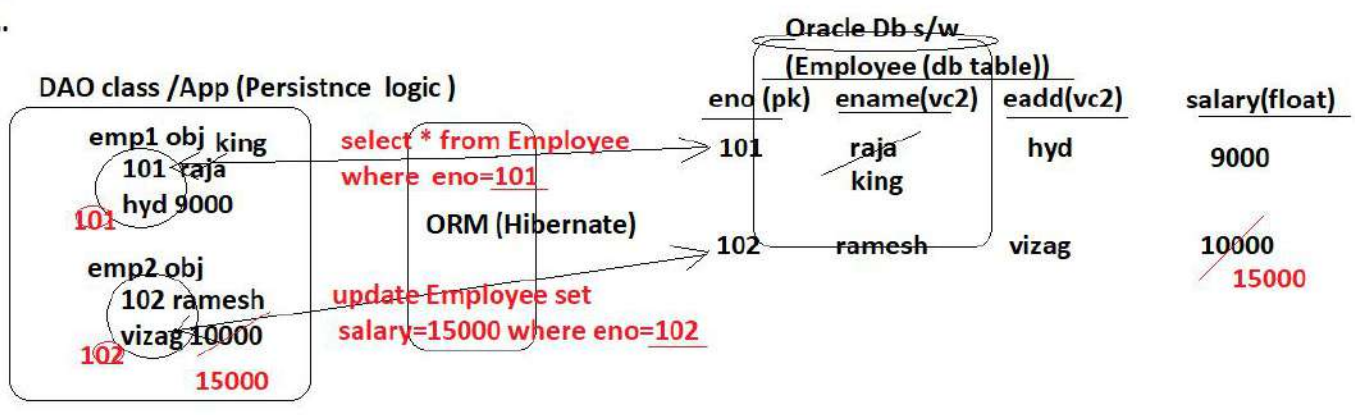
eno <------->ENO

ename <---->ENAME

eadd <---------> EADD

salary <----------> SALARY

Cols of Db table



note:: ORM framework (Hibernate) uses Each object id value as the criteria value to perform sync b/w Object and Db table row.. i.e it uses id value as criteria value while generating select ,update queries of synchroization. But choosing and configuring one property of Entity class as id property/identity field in the mapping file is the resposibility of the Programmer...

JVM identifies the object using its hashCode.. where HB indentities the Entity class object using its id value (Id field/id property value).. similarly DB table identifies the record either using rownum or using pk col value..

How we need to analyze while configuration propertie(s) as Id Propertie(s)/Id Field(s) in Hibernate Mapping file from Entity class?

=> if PK constraint is applied on one column of the DB table then it is called singularPK

=> if PK constraint is applied on multiple columns togather belonging to the DB table then it is called Composite PK

=> if we cfg single property of entity class as id field/property then it is called singular id field /property.

=> if we cfg multiple properties of entity class togather as id field/property then it is called composite id field /property.

if Db table is having singular pk ... then take that pk colum related property of Entity class and configure it as singular ID field using <id> in mapping file

if Db table is having composite PK then take that pk column related multiple properties of Entity class and configure them as composited ID field/property using <composite-id> of mapping file...

if Db table is given with out PK colums , unique key cols then better to reject those db tables to use in Hibernate..

Is It mandatory to have PK column in DB table while using that db table in our HibernateApp?

Ans) No, but it is recomanded to place... to avoid synchroization issuses...

sample Employee.hbm.xml

==========

```
<hibernate-mapping>
   <class name="pkg.Employee" table="EMPLOYEE">
     <id name="eno" column="ENO"/>      Singular Id Field cfg..
     <property name="ename" column="ENAME"/>
     <property name="eadd" column="EADD"/>
     <property name="salary" column="SALARY"/>
   </class>
</hibernate-mapping>
```

3 important Objects of Hibernate PErsistence logic
==========

(a) Configuration object
(b) SessionFactory object
(c) Session object

Keeping these 3 objects ready is called Boostraping of hibernate (Activating hibernate)

Configuration obj
==========

=> By creating object for this concrete class .. we activate Hibernate f/w by collecting Hibernate jars /libraries from the classpath or build path..

we should add <hibernate_home>\lib\required folder jars

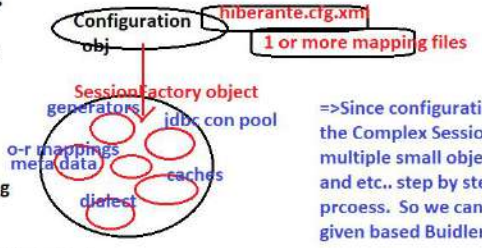
=> Takes hiberante configuration file name and location as input value and also take hibernate mapping file names and locations through hibernate configuration file name..
=> This class is given based on builder Design Pattern...

=>Using this object we can create HB SessionFactory object...
=> if no hibernate cfg file name is supplied .. then it will take hibernate.cfg.xml file from classpath (src folder of ecplise project) as default hibernate cfg file name..

```
// Activate Hibernate f/w
Configuration cfg=new Configuration();
//Supply HB cfg , mapping file names..
cfg.configure();  -->Takes hibernate.cfg.xml of classpath  as cfg file name
cfg.configure("com/nt/cfgs/mycfg.xml"); -->Takes given mycfg.xml as
                                                hibernate cfg file name
```

SessionFactory object
==========

=> Designed based on Factory Desing Pattern providing abstraction towards Session object creation.
=> The configuration object internally reads the entries of configuration file, mapping files and creates different services like jdbc con pooling, caches, dialect , Transaction Manager, o-r mapping metadata, and etc.. and uses them in the creation of SessionFactory object step by step by defining process. So we say Configuration is given based on buidler desing pattern..



=>Since configuration object is creating the Complex SessionFactory object by using multiple small objects like dailect, con pool and etc.. step by step by defining certain prcoess. So we can say "Configuration" is given based Buidler Desing pattern..

=> It is heavy weight object becoz it cotains multiple small objects reprsenting different comps and services..
=> It is Long lived in the application..
=> It is immutable object (i.e once data is set to this object we can not modify that data)
=> Immutable objects Thread Safe objects ,So the SessionFactoy object is also thread safe.

In java local variables , synchorinized objects and immutable objects are thread safe..

```
SessionFactory factory=cfg.buildSessionFactory();
```

creates SessionFactory object , first by creating multiple objects and uses them in SessionFactory object creration..(builder DP)

Fcatory PAttern says to client.. u ask object , i will object .. but u do not worry about how i am creating object...
Factory Pattern provides abstraction on object creation process...

=>SessionFactory object means it is the object of java class that implements org.hibernate.SessioFactory(I)
=> SessionFactory , Configuration objects we generally on 1 per DB s/w basis in our Apps..

Session object
==========

=> it is con++ object i.e it contains jdbc con object that is collected jdbc con pool represented by SessionFactory object and lots o-r mapping MetaData Info (++ means o-r maping metadata info)

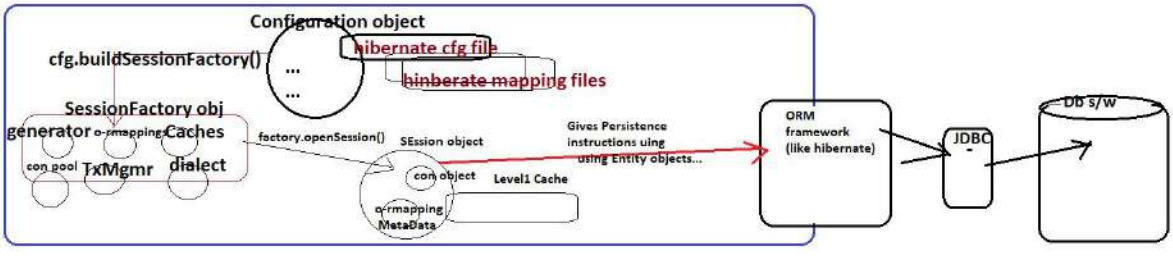
=> Created through SessionFactory object
=> It is base object for programmer to give persistnece instructions to Hibernate F/w using entity class objects...

=> It is mutable object , So it is not thread safe by default...
=> It is short lived object in App .. using SessionFactory object we create session objects
=>We geneally take Session object either 1 per each Persistence operation or 1 per related Persistence operaitons..

=> It is light weight object...
=> It comes with one built-in Level1 cache.. having capability to hold Entity class objects..

=>SEssion object means .. It is the object of the class that implements org.hibernate.Session(I)

```
Session ses=factory.openSession();
```



Hiberante Persistence operations

Single row operations
==========
(allows to manipulate
1 record at a time)

ses.save(-)
ses.persist(-) | for saving object (insert)

ses.load(-,-)
ses.get(-,-) | for loading object (select)

ses.update(-) | for updating the object

ses.delete(-) | for deleting object

ses.saveOrUpdate(-,-) | for saving /upating object
ses.merge(-,-) | for merging object

ses.refresh(-) | For reloading the object

and etc..

Bulk operations
==========
-->HQL /JPQL
->Native SQL
->Criteria API/QBC

HQL:: Hibernate Query language
JPQL :: Java Persistence Query Language
QBC :: Query By Criteria

note:: In hiberate or o-r mapping programming
-> saving object means writing object's data to db table as record
-> loading object means selecting record from Db table to Entity class object
-> updating object means updating the record represented by the object
-> deleting object means deleting the record represented by the object

Procedure to develop First Hibernate App performing save object operation

step1) keep the s/w setup ready
->Hibernate latest version :: (5.4.18)
-> Jdk (1.8+)
-> Oracle (11g +)
->Eclipse (2019-12) (Extract zip file / use installer)
eclipse-jee-2019-12-R-win32-x86_64.zip
->SQL Developer (for oracle)
(GUI Db tool)

STS
myeclipse
netbeans
IntelliJ
JCreator
JDeveloper
and etc..
popular one is :: Eclipse with STS plugin

step2) create DB table "Product" in oracle Db s/w using SQL Developer tool

Product
|---->pid (n) (singular pk)
|--->pname (vc2)
|--->price (float)
|--->qty (float)

```
CREATE TABLE "SYSTEM"."PRODUCT"
  (    "PID" NUMBER(5,0) NOT NULL ENABLE,
       "PNAME" VARCHAR2(20 BYTE),
       "PRICE" FLOAT(126),
       "QTY" FLOAT(126),
        CONSTRAINT "PRODUCT_PK" PRIMARY KEY ("PID"))
```

step3) Launche eclipse IDE by workspace space folder..
The folder where projects will be saved...

step4) create Java Project in Eclipse IDE..
File > new > project > java project > name:: HBProj1-SaveObject > finish..

step5) Add Libraries(jars) to the build path /classpath of the Project..
=> all jar files collected from <HB_home>\lib\required folder
=> ojdbc6/7/8.jar

Right click on project > buildpath > configure build path >Libraries tab >
add external jars > browser and select jar files..

note:: if App is using third party apis then those apis related both main and dependent jar files must be added to CLASSPATH/Buid path..

step6) Create packages in the Project and develop the source code...

HBProj1-SavingObject
|---->src
|--->com.nt.entity
|---->Product.java (Entity class)
|---->Product.hbm.xml (Hb mapping file)
|--->com.nt.cfgs
|--->hibernate.cfg.xml (HB cfg file)
|--->com.nt.test
|--->SaveObjectTest.java (Client App having Hibernate based o-r mapping persistence logic)

ctrl++ /-- ---> for increasing or decreasing instant font.

=>Hiberante supports disk caching ... through the serialization of Entity class objects i,e writes Entity class objects data to files of Harddisk using the concept of Serialization. but Serialization is possible only on Serializable objects.. To make objects as Serializable we should make the class of the objects implementing java.io.Serilizable(I) (Marker Interface-Empty interface)

note:: All Entity classes should be developed as java bean classes .. implementing java.io.Serializable(I).. Programmer/Hibernate uses setter methods to set/modify data of Bean objects and getter methods to read data from bean objects..

ctrl+shift+L :: To get all short cut keys...

step7) Run the the Application
Go to client App ctrl+F11
(or)
Right click on Client app source code --> run as -->java App.

Hiberante is giving DTD/XSD rules for both hb cfg file, hb mapping file to make all developers working with same tags, attributes in the development hibernate cfg file , hibernate mappig files..

DTD :: Document type definitation
XSD :: Xml schema definitation (latest) | These are Building block of xml files..

Collect DTD/XSD rules import statements for creating HB cfg file or HB mapping file from sample cfg files , mapping files of <Hibernate_home>\project\ folder..

```
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
```
For Hibernate Mapping file

```
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
```
for hibernate cfg file

Can we develop single xml file having both hb cfg and hb mapping info?

Ans) No .. Becoz Hibernate f/w is expecting DB connectivity details from HB cfg file and o-r mapping details of diff calsses and db tables from Hb mapping files whose file names are specified through HB cfg file.. So we can not mixup both..

note:: we can collect sample mapping fiels and same hibernate cfg from <Hibernate_home>\project folder...
note:: we can get the fixed property names of hibernate cfg file from <HB_home>\project\etc\hibernate.properties..

=>All non-select operations in hibernate must be executed as transactional statements.. So that we can commit or rollback thoese operations..

```
<property name="hibernate.show_sql">true</property>
```

makes hibernate f/w to show the generated SQL query as log msg

Hibernate: insert into PRODUCT (PNAME, PRICE, QTY, PID) values (?, ?, ?, ?)

The word "hibernate" is optional in hibernate cfg file properties..

```
<property name="format_sql">true</property>
```

To show the generated SQL Query as the formated SQL query.. this property must be used in combination with "show_sql"

note:: Hibernate internally uses JDBC PrpearedSatement object by making the generated SQL Queries as pre-compiled SQL Queries to executes the SQL queries in DB s/w.

```
<property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
```

Writing this property in hibernate cfg file is optional from java 8 onwords.. becoz java 8 given based jdbc4 version having support for autoloading of jdbc driver class...

It is always recomanded (in a way mandatory) to execute non-select (insert,update,delete) persistence operations in hibernate as transaction statements i.e as statements which gives ability to commit or rollback data changes..

ses.save(prod) is not actually generating insert SQL Query directly on its own to insert the record.. It actually gives save object persistence instruction to HB f/w .. by giving entity object with data.. Hibernate actucally finishers that persistence instruction when tx.commit() is called by generating the neccessary jdbc code + insert SQL query dyamically as required for the underlying DB s/w, Till that time the given Entity class object will maintained in Level1 cache of Session object

---

=>Xml file/doc that satisfies syntax rules is called well-formed Xml document..

=> xml file/doc that satisfies the imported XSD/DTD rules is called valid Xml document.

=> Xml parser is a software program/ app that can check wheather the xml doc valid or not , wellformed or not. It can also read and process Xml doc /file..

eg: SAX parser , DOM Parser , DOM4J Parser and etc..

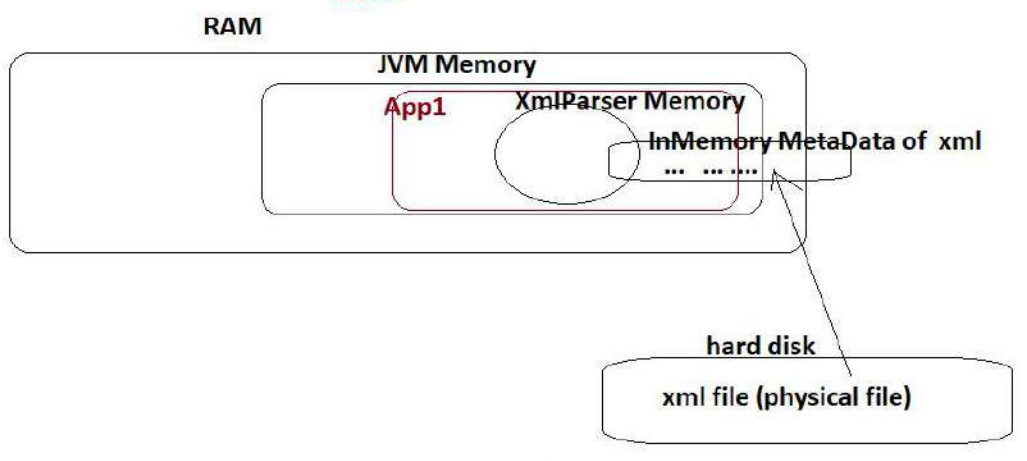
xml syntax rules

->tags are case-sensitive
->every open tag should having closing tag
->tags should nested properly
-> attributes values must in quotes..
and etc..

SAX :: Simple API for Xml Processing
DOM :: Document Object model
DOM4J :: DOM for Java (Hibernate internally uses this xml parser)

=> Xml parser loads xml files ,checks well-formed or not, valid or not , reads xml file content and creates In Memory MEtaData of xml file in the memory where XmlParser/ or its App is running (JVM memory of RAM) for the faster reusability of xml file data in the execution of the Application

what is the use equals(),hashCode() methods overrding in our class?



ctrl+shift+o -->for pkg imports
ctrl+shift +c ---> for enabling or disaling single line comments
ctrl+shift+/ --> for enabling multi-line comments on selected code
ctrl+shift+\ --> for disabling multi-line comments on selected code

Flow of first Hibernate Application
==========

```
Configuration cfg=new Configuration();
```

==>BootStraps/Activates the Hibernate f/w based on the jar files that are added to CLASSPATH env.. variale.. /Buildpath (eclipse)

```
cfg.configure("com/nt/cfgs/hibernate.cfg.xml");
```

Here configure(-) method loads the given hiberante cfg file .. ,checks wheather that file is well-formed or not , valid or not and also creates InMemory MeData.. and even stores same info in Configuration class object..
note:: configure(-) does not attempt to load mapping files..

This configure(-) method allows the programmer to take hibernate cfg file name and location accoding to our choice.. or requirement..

```
cfg.configure("com/nt/cfgs/mycfg.xml");
```

note:: by default the "src" folder every eclipse will be placed in classpath or buildpath..

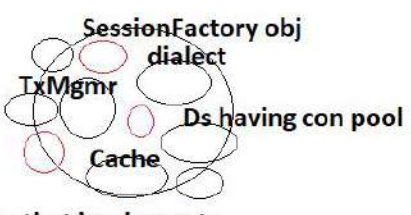
```
cfg.confgure();
```

Takes hibernate.cfg.xml file from classpath (like src folder in Eclipse Project) as default hibernate cfg file name.. (But it is bad practice).

```
SessionFactory factory=cfg.buildSessionFactory();
```

(runs on builder Design Pattern)

the buildSessionFactory() method performs
(a) From Configuration object info , it will gather the names and lcations of hibernate mapping files .. loads them , check them wheather they are well formed, valid or not and creates InMemoryMetaData for those xml files..
(b) Based info that is collected Configuration file, mapping files multiples services related objects will be created like DataSource pointing jdbc con pool , dialect , Transaction manager ,generators, caches and etc..
(c) All all above objects HB SessionFactory will be created as complex and heavy object..and returns that object..



=> SessionFactory object meas it is the object of java class that implements org.hibernate.SessionFactory(-). Since this class name chanages hibernate version to version ..we always refer by using the common "SessionFactory(I)" ref variable...

```
System.out.println("sessionFactory object class name::"+factory.getClass());
```

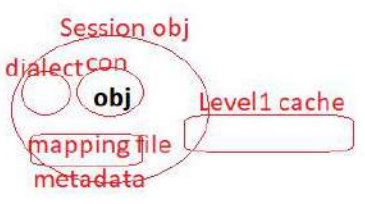
sessionFactory object class name::class org.hibernate.internal.SessionFactoryImpl

---

```
Session ses=factory.openSession();
```

con++ (based on Factory design pattern)

the factory.openSession() method is given based on factory desing pattern performing following operation
=>collects con object throguh Datasource object of SessionFactory from its jdbc con pool , also collects mapping file metaData and dialect.. to create and SEssion object having all these objects..



SEssionFactory holds huge no.of object representing huge no.of services.. where as session holds less objects representing lesss services..

SEssion object it is the object of java class that implements org.hibernate.Session(I) and this class name chanages version to version of hibernate.. So we refer Session object with org.hibernate.SEssion(I) variable (common variable)

```
System.out.println("session object class name::"+ses.getClass());
```

session object class name::class org.hibernate.internal.SessionImpl

org.hibernate.Session(I)
↑ extends
org.hibernate.engine.spi.SessionImplementor(I)
↑ implements
org.hibernate.internal.SessionImpl (c)

org.hibernate.SessionFactory(I)
↑ extends
org.hibernate.engine.spi.SessionFactoryImplementator(I)
↑ implements
org.hibernate.internalSessionFactoryImpl(c)

note:: Web applications's HttSession object is no way related to HB session object..

```
Transaction tx=ses.beginTransaction();
```

=>It Begins the TRansaction in the mode /code that is choosen in HB configuration file for the " hibernate.transaction .factory_class" property .. The possible values are "JDBC" (default) ,JTA and etc..
=>if "JDBC" is choosen then it uses Auto con.setCommit(false) method to begin the Transaction internally.. if JTA is choose then uses ut.begin() method to begin the Transaction
|-->"ut" -->UserTransaction object.

JTA -->Java Transaction API

note:: Hibernate is providing unified model for Transaction Management (begin Tx, commit Tx ,rollback Tx) for if we switch JDBC to JTA or JTA to JDBC then we need not modify the code of Hibernate Tx mgmt.. we just modify "hibernate.transaciton.factory_class" property value in hibernate cfg file..

In hibernate.cfg.xml

```
<property name="hibernate.transaction.factory_class">jdbc</property>
```

(default) jdbc / jta

```
Transactiontx=ses.beginTransaction();   //internally calls con.setAutoCommit(false) to begin the Tx
System.out.println("tx objecct class name::"+tx.getClass());
```

tx objecct class name::class org.hibernate.engine.transaction.internal.TransactionImpl

---

```
ses.save(prod)
```

Gives save object persistence instruction to Hibernate framework , but Hibernate f/w fullfills that persistene instruction by generating the SQL query only when tx.commit() method is called, till that time the given Entity object will be maintained in Level1 Cache of session object..

```
tx.commit();
```

=>Takes all the persistence instructions that given through Session object in a TRansaction and completes them by generating and executing the jdbc code + SQL queries in Db s/w.. and also commits the query execution results like calls con.commit() in case of "JDBC" and ut.commit() in case of JTA.

```
tx.rollback();
```

Rollbacks the query execution results..or cancels the query execution for persistence instructions that are given in a Transaction. internally uses con.rollback() in case of "JDBC" or use ut.rollback() incase of JTA.

```
ses.close();
```

=>closes the session i.e
-> closes the connection with Db s/w and returns that con object back to jdbc con pool
-> Vanishes the Dialect, o-r mapping metadata
-> empties and closes the Level1 Cache of "session obj".

```
factory.close()
```

=>closes the SEssionFactory by doing
(a) releases data source by cleaning up the jdbc con pool
(b) releases cache (Level2 caches)
(c) vanishes in the Memory MetaData of configuration fiel
(d) vanishes all the objcts that associated with "SessionFactory object".
(e) Deactives the hibernate f/w..

What is the difference b/w SessionFactory object and Session object?

SessionFActory object	Session object
(a) Heavy weight object containing multiple other objects like DS, dialect, TxFactory,generators and etc..	(a) Light weight object (con++)
(b) Immutable object	(b) Mutable object
(c) Thread safe object by default	(c) Not Thread safe object by default
(d) Generally we take one Object per DB s/w basis	(d) Generally we take on 1 per Persistence operation or 1 per related peristence operations basis
(e) Created by using Configuration object.buidSessionFactory() method which is designed based on Builder DP	(e)Created by using factory.openSEssion() which is designed based on Factory Pattern.
(f) Long Lived object of the Application	(f) Short lived object of the Application.
(g) It is the object of java class that implements org.hibernate.SessionFactory(I)	(g) It is the object of java class that implements org.hibernate.Session(I)
(h) Maintains configurable Level2 Cache	(h) Maintain built-in Level1 Cache

ses.save(-) method
==========

This method gives persistence instruction to save object and also returns either assigned or generated identity value back to the App as the return value...

note:: if no generator is configured ..then the value assinged to "id" property will be returned as identity value.. otherwise "the" genereator generated value will be returned as the identity value..
The generators are "increment" ,"sequence","hilo" and etc..

In Client App
==========

```
int idval=(int)ses.save(prod);
S.o.p("generated id vlaue ::"+idval);
```

signature of the ses.save(-) method
---

public Serializable save(Object obj)

This allows to pass any Entity class object as the argument value..

Allows to return any Wrapper class obj/String related objects having id value..

note:: All wrapper classes and String are Serializable

In Product.hbm.xml
==========

```
<id name="pid" column="PID">
    <generator class="increment"/>
</id>
```

uses maxval(id) +1 formulae to generate the id value..dynamically..

---

ses.persist(-) method
==========

JPA :: Java Persistence API.

=> Gives instruction to HB f/w to persist/save object ..
=> does not allow to work with generators
=> return type is void.. i.e can not return the identity value..
=> this method is given by JPA specification.. and implemented in hibernate

public void persist(Object obj)

sample code
==========

```
try{
  tx=ses.beginTransaction();
     ses.persist(prod);
     flag=true;
  }
catch(Exception e){
     flag=false;
  }
finally{
   ...
   ... //same old code..
   ...
}
```

What is the difference between save(-) and persist(-) methods?

save(-)	persist(-)
(a) Can return the generated id value value as Serializable wrapper object	(a) Can not retrun the id value...
(b) return type is Serializable..	(b) return type is void
(c) Can work with generators to generate the value to id /identity property dynamically	(c) can not work with generators..
(d) It is direct method of Hibernate f/w	(d) It is gathered an implemented by collecting from JPA Specification
(e) Can not give guarantee for this method availability in all ORM frameworks	(e) Available in multiple ORM frameworks...
signature :: pubic Serializable save(Object obj)	signature :: pubic void persist(Object obj)

The SQLDialect generates insert,update,delete,select select queries as pre-generated queries for all the Db tables that are specified in all the mapping files..having id column as the criteria column.. all these operations will take create in the Process of creating SessionFactory object... so when call ses.save(-) ,ses.get(-) ,ses.update(-) and ses.delete(-) methods these pre-generaed queries will be used to complete the persistence operations... But the problem with these pre-generated queries is .. they involve all the cols.. To generate dyanamic SQL queries(involves only required cols) the we need to go for
dynamic-insert="true" , dynamic-update="true" .. in <class> tag of mapping files...

```
prod=new Product();
prod.setPname("chair");
int idval=ses.save(prod);
```

The dynamic SQL query is
insert into Product(pid,pname) values(?,?)

a dynmic-insert="true" is taken

```
prod=new Product();
prod.setPname("chair");
prod.setPrice(9000); prod.setQty(3);
```

the pregenrated query is
insert Product (pname qty,price,pid) values(?,?,?,?)

Why there is no "dynamic-delete" , dynamic-select attributes in <class> tag?

Ans) In the delete query col names are not required.. only condition is required.. So there is no need of dynamic-delete ...
=>delete from product where pid=?

ses.get(-,-),ses.load(-,-) can get whole record into Entty class object i.e they can not get specific col values.. So dynamic -select is not required.. in single row operations...

note:: But by using HQL,NativeSQL, Criteria API we can go for Dynamic selects..

note:: It is good pratice to devleop Entity clasess with wrapper data types.. not with simple data types...

SQL Dialects /Dialects / Hiberante dialects
==========

=>It is imp comp of Hibernate f/w ... that is capable generating both pre-generated/static and Dynamic SQL Queries...

=> Dialect are useful

a) To generated optimized SQL Queries as required for the underlying DB s/w..

b) To assign Sensible,intelligent default values to the configuration properties which are not specified in hiberante cfg file..

=> In Most of cases , hibernate automatically picks up the dialect... based jdbc properties u supply..

=> All Dialects are hibernate api supplied pre-defined classes extending from org.hibernate.dialect.Dialect(c)

=>These dialect class names will change based on DB s/w and and its version we use...

In hibernate.cfg.xml

```
<property name="dialect">org.hibernate.dialect.Oracle10gDialect</property>
```

oracle10g/11g ----------> Oracle10gDialect
oracle12c ----------> Oracle12cDialect
oracle8i ---------->Oracle8iDialect
oracle9i ---------->Oracle9iDialect
mysql5 ---------->Mysql5Dialect.
and etc..

note:: Dialect is very useful and quite mandatory when we have installed multiple versions of same dB s/ws in a single computer...

Q) Can we specifcy higher or lower version Dialect to current DB s/w.. in Hiberante cfg file?
Ans) Possible , but not recomanded...

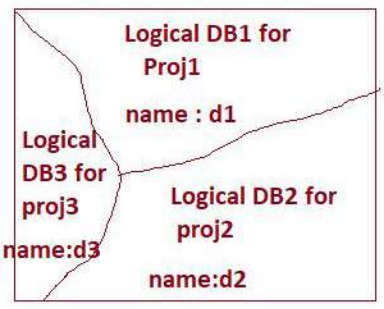
Q) Can we specifcy different DB s/w.. Dialect with out worrying about underlying DB s/w in Hiberante cfg file?
Ans) Possible , but not recomanded...

Mysql
======

type :: Db s/w
version:: 8.x
vendor :: devx/sun Ms/oracle corp
default port no:: 3306
admin username :: root
password :: root (choose during installation)
To download s/w :: https://www.mysql.com/downloads/
Gives Myworkbench as GUI DB tool.. (mysql-installer-community-8.0.16.0.ext)
(built-in tool)

MariaDB DB s/w is copy of MysqlDB s/w..

Physcial DB s/w vs Logical DB



The Logical Partiation of Physical Db s/w is called Logical DB.. It is generally on 1 per Project basis... So Every Project related Db tables, PL/SQL procedures/functions, views and etc.. will be created in one Logical DB...

=>In Oracle DB s/w, every Logical DB is identified with its sid.. (xe,orcl and etc..)
=>In Mysql DB s/w, every logical DB is identified with its db name

Procedure to create Logical DB in mysql Db s/w having DB table using mysql workbench?

a) create connection by giving connectio name , db username, db pwd details

b) create logical DB (NTHB914DB)

c) create Db table

right click on logical DB (NTHB914DB) -->create table ---> give db table name, col names and datatypes...

To interact mysql DB s/w ,we need connector/j JDBC driver that comes in the form mysql-connector-java-<ver>.jar file .... and the details are

a) driver class name :: com.mysql.cj.jdbc.Driver

b) jdbc url :: jdbc:mysql:///<LogicalDB> (for local mysql) eg:: jdbc:mysql:///nthb914db
jdbc:mysql://<host>:<portni>/<logical Db> (for remote mysql) eg: jdbc:mysql://localhost:3306/NTHB914DB

c) jar file :: mysql-connector-java-<ver>.jar
(download seperately from internet)

To convert HB First App working for Mysql
==========

(a) add mysql-connector-java-<ver>.jar file to build by downlaoding from internet

(b) Do following changes in hibernate.cfg.xml file and run the Client App

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
  <hibernate-configuration>
    <session-factory>
      <!-- connection properties -->

      <property name="connection.url">jdbc:mysql:///NTHB914DB</property>
      <property name="connection.username">root</property>
      <property name="connection.password">root</property>

      <!-- hibernate properties -->
       <property name="show_sql">true</property>
       <property name="format_sql">true</property>
       <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>

      <!-- Mapping file names -->
       <mapping resource="com/nt/entity/Product.hbm.xml"/>
    </session-factory>
  </hibernate-configuration>
```

java.lang.Class (pre-defined class)

==========

The object of java.lang.Class holds class name, interface name, annotation name, enum name in a running java App... as the data of the Object...

Class c=new Class() (x--> not possible) (compile time error)
(Becoz class is having This only one private 0-param constructor)

other ways of creating objects for java.lang.Class

==========

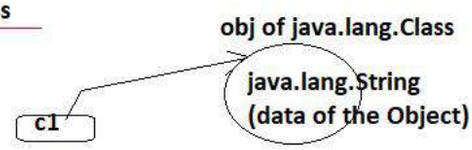
a) Using Class.forName(-) method

Class c=Class.forName("java.util.Date");

b) Using getClass() method of java.lang.Object class

String s=new String("ok");

Class c1=s.getClass();



c) Using "class" property

Class c2= System.class;

"length" , "class" and etc.. are the built-in properties of every java class that are generated by java compiler dynamically during the compilation process..

Proxy DesignPattern

==========

Proxy ----> means dummy that looks like orginal.. but not useful to addtional functionalaties on tempoary basis... Proxy internally uses orginal Object if needed.

usecase:: => Bank should allow any amount of money to withdraw (orginal/real)
=>During demonitisation it should allows only 4000 though customer asks to withdraw more than 4000..(Use Proxy for this)

Proxy Design Patterns add extra functionalities dynamically at runtime by internally taking the support of real object/functionalaties..

The components of Proxy Design pattern

==========

(a) Component Interface
(b) concrete Component class (real class) (implements Component interface)
(c) Proxy class (implements Comp interface and also contains component class obj) (HAS -A relation)
(d) Factory class (To return either Real or Proxy object based on the need)

HBProj3-Proxy DP
|-->
src
|-----> com.nt.component
|---->BankService.java
|---->BankServiceImpl.java
|---->com.nt.proxy
|---->BankServiceProxy.java
|---->com.nt.factory
|------> BankServiceFactory.java
|---->com.nt.test
|------>BankEmployee.java (Client App)

Java Stream api (java 8 feature) is good to go through elements of arrays or collections either sequentially or parellely with out using loops and very useful to perform aggregate operations...

note:: we can generate proxy classes as dynamic InMemomory classes using CGLIB ,JDK Libraries....

HibernateUtil class

==>Utility classes are helper classes that are required in multiple parts of project representing boilerplate code...

=>Instead of the logic to create Configuration , SessionFactory, Session objs in every Client App or in every method of multiple DAO classes. It recomanded to place only for 1 time in seperate HibernateUtil class ... (HibernateUtil.java)

In this HibernateUtil.java

-->we should create SessionFactory ,Configuration objects 1 per Db s/w in the entire Application (static block)

--> we should create Session object either on 1 per each Persistence operation (single row operations) or 1 per related Persistence operations basis.

(usecase: transferMoney operation (static method)
(withdraw+deposite operations)

-> Logic to close given Session object (static method)

-> logic to close SessionFactory object (static method)

note:: java8 interface can have static methods .. but can not have static block , So Do not HibernateUtil.java as java8 interface..

note:: Factory class return one of the several related classes object ... where as Utility class return diff objects by calling different methods... At max factory class contains only static/not-nonstatic factory method.. where as utility classes contain multiple static factory methods ...returning different objects..

example for factory classes	examples for Utility classes
SessionFactory	Collections
BeanFactory	Arrays
CarFactory	HibernateUtil
	IOUtil
	BeansUtil
	StringUtils

note:: Since the entire App is ready to use Single SessionFactory ,Configuration objects per Db s/w basis it is better to create them in the static block of HibernateUtil class , So that they will be created automatically the moment HibernateUtil is loaded...

HibernateUtil.java (We take it on 1 per DB s/w basis)

```
public class HibernateUtil{
  private static SessionFactory factory;
  static {
   Configuration cfg=null;
    try{
    //Bootstrap hibernate
     cfg=new Configuration();
    // input hb cfg file
     cfg.configure("com/nt/cfgs/hibernate.cfg.xml");
    //create SessionFactory object
     factory=cfg.buildSessionFactory();
    }
   catch(HiberanteException he){
        he.printStackTrace();
    }
    catch(Exception e){
      e.printStackTrace();
   }
}

 public  static Session getSession(){
     if(factory!=null)
        retrun factory.openSession();
     else
       return null;        }
 public static void closeSession(Session ses){
       if(ses!=null)
            ses.close();
  }
  public  static void  closeSsionFactory(){
      if(factory!=null)
                factory.close();
    }
}
```

note:: Since the static block can not propagate the exceptions.. So we catch handle exception. alteast to write exception related messages log files...

=>try with resource
=>inner classes
=> java 8 features
- ->forEach method
- -> LAMDA expression
- -> Functional interfaces
- ->Streaming api
- ->java8 interface
- ->Optional API
- -> method reference

=> java8 date and time

In ClientApp /DAO class

```
Session ses=HibernateUtil.getSession();

   ...
   ....
   ....

HibernateUtil.closeSession(ses);
HibernateUtil.closeSessionFactory();
```

Loading an Object
==========

=>selecting a record from DB table and storing into an Entity class class object....

For this we can use

a) ses.get(-,-) ---> Performs eager Loading (No Proxy design Pattern)

gets a record db table and stores into an Entity class obj irrespective of whether we use that object/record or not..

b) ses.load(-,-) ---> Performs lazy(default)/eager Loading (Uses Proxy Design Pattern)

Signatures
==========

```
public <T> get( Class<T> clazz, Serializable id)
public <T> load( Class<T> clazz, Serializable id)
```

*--> We need to pass the Entity class name as the object of java.lang.Class.
like Student.class, Customer.class, Employee.class | gives objects of java.lang.Class having entity class names.

Exmple on ses.get(-,-) method
==========

```
InsurnacePolicy policy=
      ses.get(InsurnacePolicy.class,1001L);
              *1                     *2
```

*1-->Passing java.lang.Class obj having InsurancePolicy as the Entity class name
*2 --> Passing 1001L which will be internally converted to java.lang.Long wrapper class obj (serializable obj) using autoboxing concept..

note: All Wrapper classes are Serializable by default
note:: The Process of converting simple values to objs automatically is called auto boxing and reverse is called auto unboxing.

```
if(policy==null)
  S.o.p("record not found");
else{
  S.o.p(policy)
}
```

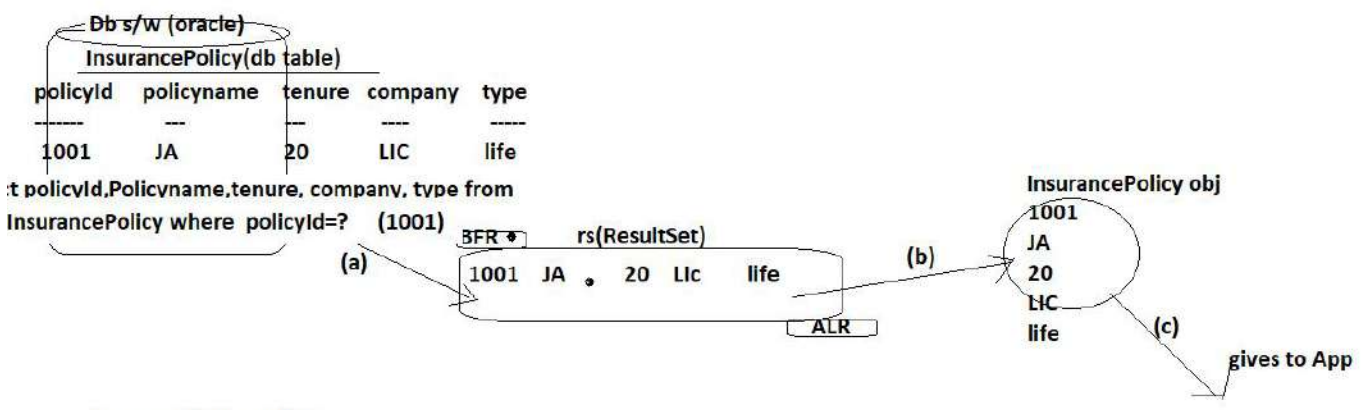
//while working ses.get(-,-) we can check wheather record is available or not.

InsurancePolicy (db table)
|--->policyId (n) (11) (pk)
|-->policyName (vc2)
|-->tenure (n) (10)
|-->company (vc2)
|-->type (vc2)

```
//Entity class
public class InsurancePolicy{
  private long policyId;
  private String policyName;
  private int tenure;
  private String company;
  private String type;
  //setters && getters
   .....
   ....      //toString
  }          ...
```

//Generated SQL Query is ::
Select policyId,Policyname,tenure, company, type from InsurancePolicy where policyId=? (1001)

note: set.get(-,-) hits the Db s/w the moment it is called.. and tries to get the record to store in to given Entity class obj by creating object for Entity class..




```
InsurnacePolicy policy=
      ses.get(InsurnacePolicy.class,1001L);
              *1                     *2
```

Internal code
==========

```
//create PreparedStatment object
PreparedStatement ps=con.prpeareStatement("Select policyId,Policyname,tenure, company, type from InsurancePolicy where policyId=?")
                                          pre-geneated query by SQL Dialect while creating SessionFactory object
//set values to query params
ps.setLong(1,1001);
//execute Query
ResultSet rs=ps.executeQuery();
// Create given Entiy class object..
InsurancePolicy policy =(InsurancePolicy) (InsurancePolicy.class).newInstance();
//map ResultSet object record to Entity class object
  if(rs.next()){
     policy.setPolicyId(rs.getLong(1));
     policy.setPolicyName(rs.getString(2));
     policy.setPolicyTenure(rs.getInt(3));
      ...
      ...   return policy;
   }
  else{
   return null ;
  }
```

Procedure to develop hibernate App using Gradle in Eclipse
==========

step1) make sure that Buildship plugin is installed in Eclipse IDE
step2) Create Gradle Project ...

File -> project --->gradle -->Gralde Project ---> Project Name :: HBProj4-LoadingObjects

->select gradle wrapper ---> select Auto Sync ---> ....

step3) develop build.grdle as shown below by gahering dependencies (jars) info from mvnrepository.com

build.gradle
==========

```
plugins {
  id 'application'
}
repositories {
  jcenter()
}
dependencies {
  // https://mvnrepository.com/artifact/org.hibernate/hibernate-core
implementation group: 'org.hibernate', name: 'hibernate-core', version: '5.4.18.Final'
  // https://mvnrepository.com/artifact/com.oracle.ojdbc/ojdbc8
implementation group: 'com.oracle.ojdbc', name: 'ojdbc8', version: '19.3.0.0'
}
```

collect from mvnrepository.com

step4) Add packages and develope source code

We should cfg id proeprty/id field using <id> tag ...
irrespective of undelrying Db table is having PK col nor..

To get Hibernate configuration file properties list
==========
https://docs.jboss.org/hibernate/core/3.3/reference/en/html/session-configuration.html#configuration-hibernatejdbc

note:: we can execute select persistence oepreations in hibernate as non-transactional statements...becoz they do not effect/modify Db table records.. more over they just read/select the records...

go to code and press f3 or ctrl+click :: for getting soruce code

```
HBProj4
 |------->src/main/java
          |--->com.nt.cfgs
                |--->hibernate.cfg.xml
          |--->com.nt.entity
                |---->InsurancePolicy.java
                |--->InsurancePolicy.hbm.xml
          |--->com.nt.utility
                |---->HibernateUtil.java
          |--->com.nt.test
                |---->LoadingObjectTest.java

          >build.gradle
```

step5) Run the Application....

Right clikc on Project --->run as ---> java Application... (ctrl+f11)

Running using gradle
---

a) copy com.nt.cfgs, com.nt.entity packages to src/main/resources folder
b) perform gradle refersh
right click on Project --->Gradle ---> refresh project..
c) Go GradleTasks tab ---> go to ur App --> expand application -->run ---> execute the task..

note:: set.get(-,-) does not throw any exception if the record is not found ..... it just returns null ...

Working with ses.load(-,-)
==========

=>can perform by lazy(default) and eager loading of object

=> if lazy="false" in <class> tag then it performs eager loading , if lazy="true" (default) in <class> tag then it performs lazy loading

=> Internally uses Proxy Design Pattern while performing lazy loading..

=> when ses.load(-,-) is called by haivng lazy="true" (default) then it first returns having proxy object(dummy object) having id value as the sub class object of Entity class by generating Proxy class as InMemory class using javaassist libraries. when call non-identifier methods on the proxy object the it hits the Db s/w , gets the record ,keeps in real Entity object that is created inside the proxy obj.. So that we can access record values through proxy object..

=>All single row operations in Hibernate takes place by taking the identity values as the criteria/condition values..

=>if u want to use other than identity value as the criteria value then go for HQL/NativeSQL/Criteria API

=> Dynamic select, delete queries generation for single row operations using single row method calls like ses.get(-.-)/load(-,-) and ses.delete(-) not possible.

signature::

public <T> T load(Class <T> clazz,Serializable id) throws HibernateException

<class .... lazy="true" />

(default is true)

(a)

InsurnacePolicy policy=ses.load(InsurancePolicy.class, 9001);

(b) generates InMemoryProxy class as the subclass of Entity class using javaassist libraries having ref variable pointing to Entity class(real class) , creates object for Proxy class having given id value..

Proxy obj (InsurancePolicy$$Proxy)

InsurancePolicy real :: null

9001 (id)

policy

(c) policy.getPolicyName(); (non-identifier method)

(called on proxy obj)

internall calls getPolicyName() on real obj

(h) o/p JA

(i) policy.getCompany()

internally calls getComplay on real obj

LIC

(o/p)

(d)

uses the pre-generated SQL select Query

select policyId,policyName,policytype, company, tenure where policyId=?

(9001)

oracle Db s/w

policyId	policyName	policyType	company	tenure
9001	JA	life	LIC	20

(e)

BFR rs(ResultSet)

9001 JA life LIC 20

ALR

(f)

real object

(InsurnacePolicy obj)

9001

JA

life

LIC 20

(g)

ses.load(-,-) method makes the getter methods of proxy class throwing org.hibernate.ObjectNotFoundException: No row with the given identifier exists: [com.nt.entity.InsurancePolicy#nnn] if record not found load from Db s/w

Q) why there is no ObjectNotFoundException in the signature of ses.load(-,-) method...

Ans) ses.load(-,-) is not throwing that exception, the InMemory proxy class getter methods are throwing that exception, So it will not be there in signature of ses.load(-,-)

The InMemory Proxy class
==========

interfaces related to javaassit

```
public InsurancePolicy$Proxy extends InsurancePolicy implements ..... {
    private InsurancePolicy real;
    private long id;

    public InsurancePolicy$Proxy() {
     //intialize id with second argument value
     that is collected from ses.load(-,-) method call..
      ... ....
    }
```

Q) Who generates Proxy class when ses.load(-,-) is called?

Ans) The JavaAssist jar file of Hibernate libraries...

```
    public long getPolicyId(){
      return id;
    }

    public String getPolicyName(){
      if(real == null){
        try{
         // jdbc code to get record
            from Db s/w by taking id value as the criteria value
            PreparedStatement ps=con.prepareStatement (" select policyId,policyName,policytype,
                                                         company, tenure where policyId=?");
            ps.setInt(1,id);
            ResultSet rs=ps.executeQuery();      //hitting the DB
            if(rs.next()){
                real=new InsurancePolicy(); //creating real obj inside the proxy object
                real.setPolicyId(rs.getLong(1));
                real.setPolicyName(rs.getString(2));
                real.setPolicyType(rs.getString(3));
                 ....
               }
             else
               throw new ObjectNotFoundException(" No row with the given identifier exists:.....");
               }
            }//try
            catch(SQLException se){
      throw new HibernateException(se.getMessage());
            }
      }//if
      return real.getPolicyName();
    }

    public String getPolicyType(){
        ....
        ....
        .... //same as above
        .....
      return real.getPolicyType();
    }

    ....
    ..... //other getter methods
    ....
    ....
    ...
    public void setPolicyId(long policyId){
      if(real!=null)
        real.setPolicyId(policyId);
    public void setPolicyName(String policyName){
      if(real!=null)
        real.setPolicyName(policyName);
    ....
    .... //setter methods..
    ....
}
```

=>when ses.get(-,-) is called only 1 object for entity class will be created.. to store the selected record.
=>when ses.load(-,-) is called total two objects will be created . (1. proxy object 2.real entity object)

Performing eager loading by using ses.load(-,-) method
==========

In <class> of mapping file take lazy="false"
In InsurancePolicy.hbm.xml file

```
<class name="pkg.InsurancePolicy" table="....." lazy="false">
    ....
    ....
</class>
```

In Client App

```
InsurancePolicy policy=ses.load(InsurancePolicy.class, 9001L);
S.o.p(policy)
```

=>Here no proxy class will be created.. it behaves like ses.get(-,-) method.. but thorws ObjectNotFoundException if the record is not available..where as ses.get(-,-) returns null if the record is not avaiable...

What happens if we take tthe entity class as final class or its getter methods as final methods while working with ses.load(-,-) method by having lazy="true"?

Ans) Performs eager loading and no proxy class will be generated.. it behaves like ses.get(-,-) method..becoz the proxy class can not be generated as the sub class of final entity class.. or the final methods of entity class can not be overridden in the proxy class (sub class of entity class)

=>In this situation if the give id based record is not avaiable.. to load then exception will be raised that is ObjectNotFoundException

Q) by lazy="true" in <class> can we perform eager loading by using ses.load(-,-) method

Ans) Take Entity class as final or take entity class getter methods as final methods..

note:: ses.get(-,-) always performs eager loading .. irrepsetive of "lazy" attribute values taken in <class> tag.. of mapping file..

Q) How can we perform lazy loading using ses.load(-,-)method even though entity class is final class?

Ans) Make Hibernate F/w's javaassist libraries generating Proxy class not as the sub class of entity class .. make it as the implementation class of Proxy Interface.. i.e Proxy class should as the implementation class of given Interface (proxy interface)

step1) create ProxyInteface having the decl of getter, setter methods

IInsurancePolicy.java
==========

(This is called proxy interface becoz the javaassist libraries are generating proxy class based on this proxy interface).

```
public interface IInsurancePolicy{
    public Long getPolicyId();
    public void setPolicyId(Long policyId) ;
    public String getPolicyName();
    public void setPolicyName(String policyName);
    public String getPolicyType();
    public void setPolicyType(String policyType);
    public String getCompany();
    public void setCompany(String company) ;
    public Integer getTenure();
    public void setTenure(Integer tenure) ;
}
```

step2) make Entity class implementing Proxy Interface

InsurancePolicy.java
==========

```
public class InsurancePolicy implements Serializable,IInsurancePolicy{
    private Long policyId;
    private String policyName;
    private String policyType;
    private String company;
    private Integer tenure;

    public InsurancePolicy() {
        System.out.println("InsurancePolicy:: 0-param consructor"+this.getClass());
    }
   //getters & setters
    ....
    .....
   //toString()
    ......
}
```

note:: here we can Entity class as the final class..becoz proxy class comes as the impl class of Proxy interfaces.. not as the sub class of entity class..

step3) cfg Proxy Interface in mapping file...

In InsurancePolicy.hbm.xml

```
<class name="com.nt.entity.InsurancePolicy"
        table="INSURANCEPOLICY" lazy="true" proxy="com.nt.entity.IInsurancePolicy">
    ....
    .....
</class>
```

step4) In client App refer Proxy object with Proxy Interface ref variable.. (becoz we can not refer with entity class ref variable as Proxy class is not coming as the sub class of entity class)

In Client App

```
//Load object
IInsurancePolicy policy= ses.load(InsurancePolicy.class,9001L);
```

Proxy interface ref variable..

Generated Proxy class
==========

```
public class InsurancePolicy$Proxy implements IInsurancePolicy, HibernateProxy {
    private IInsurancePolicy real;
    private long id;

  public InsurancePolicy$Proxy (){
      //initialize id
        ...
  }
  //getter methods implementation.. as seen previous proxy class
   ...
   ....
   ....
   ....
}
```

What is the different b/w set.get(-,-) and ses.load(-,-) methods?

ses.get(-,-)	ses.load(-,-)
=>Performs eager Loading of object	=> Performs lazy loading of object by default can also perform eager loading
=>Does not gernate proxy class	=> generates the proxy class
=> Returns null, if the record not avalable to load	=> throws ObjectNotFoundException indirectly..
=> suitable to verify wheather record is available or not	=> not suitable
=>hits the DB s/w the moment get(-,-) method is called	=> first returns proxy object.. and hits the DB s/w only when non-identifier methods are called on proxy object
=> Creates only one object for entity class	=> creates two objects 1 proxy object + 1 entity obj(real)
=> no way related to lazy attribute of <class> tag	=> lazy="true"(default) performs lazy loading lazy="false" performs eager loading...
=> no way related to proxy attribute of <class> tag	=> if we specify proxy interface name in the "proxy" attribute , the proxy class will be generated based on proxy interface, This allows to take our Entity class as final class..
=> use ses.get(-,-) in standalone Apps where single layer will be there having all the logics and having the guarantee that the loaded record/object will be used immediately	=> prefer using ses.load(-,-) method in layered Apps.. the record asked in DAO class is actually will be displayed either in the Client app/in the view comp (jsp) , So DAO gives Proxy object to Client App/view comp by calling ses.load(-,-) and the real hit to Db s/w takes place when non-identifier methods are called on proxy object from client app/view comp .

try with resource
==========

=> alternate to try , catch blocks with finally
=> Introduced from java7
=> on Io streams, we can apply from java7 itself
=> On JDBC objects, we can apply from java8
=> On Hibernate objects, we can apply from Hiberante 5.x

if we open/create any stream /jdbc/ hibernate or other obj using try with resource .. there is no need of closing it seperately..by taking finally block.. it will be closed automatically at the end of try block.. but the objects should be auto closeable objects..

The objects whose classes are implementing java.lang.AutoCloseable(I) are called AutoCloseable objects...
from java7 onwords, all IO stream objects,
from java8 onwords , all jdbc objects,
from hibernate5 onwords , Session,SesssionFactory objs are auto closable..

<u>try with reosuce syntax</u>

```
try(res){
  ...
  ...
} // here res will be closed automtically
```

Updating object (Modifying the record of the Db table)
==========

=> use ses.update(-) for this..

public void update(Object obj)

3 different approaches of updating the object
==========

Approach1) Upading total object../total record based .. (Full object modification)

```
//prepare Entity object having new values in other properties and having old id value for identity property
 Produdct p=new Product();
 p.setPid(1001) ; //should be existing value
 p.setPname("table1"); p.setPrice(9000); p.setQty(10);
try{
 tx=ses.beginTransaction();
     ses.update(p);
    flag=true;
 }
 catch(HibernateException he){
   he.printStackTrace();
    flag=false;
 }
 finally{
   if(flag){
      tx.commit();
         S.o.p("objected updated);
    }
    else{
      tx.rollback();
      s.o.p("Object is not updated");
      }
   }
```

must be new values if want to modify.. otherwise u should rember and pass old values.

note:: Approach1 good for modifying complete/total object i.e not good for modifying partial object.

=> Using Approach1 , we can not verity wheather record is updated or not.. becoz update(-) return type is void .. more over it generates update SQL Query directly with out generating any select query.. tx.commit() throws exception for ses.update(-) method , if record is not found..

Approach2) Load and modify object (Good for partial modification of the object)
==========

```
try{
   //Load object
   Product   p=ses.get(Product.class,1001);
   if(p!=null){
     tx=ses.beginTransaction();
     //modify partial object
       p.setPrice(10000);
       ses.update(p);
       flag=true;
 }
 else{
   S.o.p("record not found");
      return;
   }
 }//try
catch(HiberanteException he){
 he.printStackTrace();
   flag=false;
 }
 finally{
     if(flag){
        tx.commit();
          S.o.p("object updated");
        }
     else{
          tx.rollback();
          s.o.p("object not updated");
        }
   ...
 }
```

Approach2 is good for partial moidifcation of object becoz the loaded obj maintains old values.. that u do not want to modify

Approch3: Load object and modify object with out calling ses.update(-) method
========= (Good for Partial modification of the Object)

note:: if we modify the loaded object in a Transaction ... the modifications will be synched with Db table record when call tx.commit() method automatically.. So there is no need calling ses.update(-) method seperately..

```
try{
   //Load object
   Product   p=ses.get(Product.class,1001);
   if(p!=null){
     tx=ses.beginTransaction();
       //modify partial object
         p.setPrice(10000);
         flag=true;
 }
 else{
   S.o.p("record not found");
      return;
   }
 }//try
catch(HiberanteException he){
 he.printStackTrace();
   flag=false;
 }
 finally{
     if(flag){
        tx.commit();
          S.o.p("object updated");
        }
     else{
          tx.rollback();
          s.o.p("object not updated");
        }
   ...
 }
```

Approach3 is good for partial moidifcation of object becoz the loaded obj maintains old values.. that u do not want to modify

if entity class name matches with db table name and entity class properties names match with db table column names then there is no need of specifying db table name and column names in the hibernate mapping file...

```
<class name="com.nt.entity.BankAccount" >
    <id name="acno"/>
    <property name="holderName"/>
    <property name="balance"/>
</class>
```

Db table
========
BankAccount
|--->acno (n) (pk)
|--->holderName (vc2)
|---->balance (n)

=>In Approach1 .. the update SQL query will be generated irrepspective of modifications in DB table record required or not with resepect to Entity class object data.

=>In Approach2,3 .. the update SQL query will be generated only when DB table record needs the modification with respect to object data..

How can u show the synchronization Entity obj to DB table row?

Ans) if we modify the loaded object in a Transaction .. then the modifications will reflect to Db table record when tx.commit() method is called..

eg:: refer approach3 code of "updating object".

How can we show synchronization from DB table row to Entity object?

Ans) First load the object.. (select the record into Entity class object) and call ses.refresh(-) method.. having entity object as the argument value.. to get chanages done in Db table row to Entity Object.

```
try {
//Load object for partial moidification of the object
account=ses.get(BankAccount.class,1001L);
if(account!=null) {
        System.out.println(account);

        System.out.println("modify 1001 record in DB table from SQL prompt/developer");
        try {
         Thread.sleep(40000);  //modify db table record using SQL prompt or SQL developer
        }
        catch(Exception e) {
                e.printStackTrace();
        }

        ses.refresh(account);  //Db table row to Entity object sync
        System.out.println(account);
}
else {
     System.out.println("record /object not found");
}
    }//try
```

Deleting object

=>Deleting the record represented by Entity class object...
=> Deletes the record based id value of the given entity object..
=> use ses.delete(obj) method
public void delete(Object obj)

Approach1 (Direct deletion..)
==========
(directly call ses.delete(-) haivng entity object. In that entity object we can place only id value)

```
Transaction tx=null;
boolean flag=false;
try{
 tx=ses.beginTransaction();
   Product p=new Product();
    p.setPid(1001);
    ses.delete(p);
    flag=true;
 }
 catch(HibernateException he){
    he.printStackTrace();
     flag=false;
  }
 finally{
    if(flag){
       tx.commit();
         S.o.p("Object deleted");
     }
    else{
      tx.rollback();
       s.o.p("object not deleted");
     }
   }
```

In this Approach1 we can not find wheather record is deleted or not.. becoz here we are attempting the deletion of record with out loading of record.

note: Though we are calling ses.delete(-) directly it first genertes 1 select query to check wheather record is avaiable or not internally.. if avalilable then it generates delete SQL query to delete the record otherwise delete query will not be generated..

Approach2 (Load and delete object) (recomanded)
==========

```
Product p=null;
Transaction tx=null;
boolean flag=false;
try{
   tx=ses.beginTransaction();
  //Load object
    p=ses.get(Product.class,1001);
     if(p!=null){
        ses.delete(p);
         flag=true;
     }
    else{
     s.o.p("record not found to delete");
      return;
     }
   }//try
catch(HibernateException he){
   he.printStackTrace();
    flag=false;
  }
finally{
    if(flag){
       tx.commit();
         S.o.p("Object deleted");
     }
    else{
      tx.rollback();
       s.o.p("object not deleted");
     }
   }
```

note:: In any DB s/w we can not go for partial deletion of the record.. but we can have partial updation of the object...

note:: here select query is generated becoz select.get(-,-) and delete query will be generated becoz of ses.delete(-) method.

---

if u want see hibernate f/w generating dB tables dynamically then better to sepcify type,lenght , not-null,unique and etcc.. details in mapping file.. along with the reqular property names, col names.. as shown below.

```
<hibernate-mapping>
  <class name="com.nt.entity.Project" table="PROJECT">
    <id name="projId" column="PROJID" type="long" length="11" />
    <property name="projName" column="PROJNAME" type="string" length="15" not-null="true" unique="true"/>
    <property name="teamsize" column="TEAMSIZE" type="int" />
  </class>
</hibernate-mapping>
```

gives <id> automatically pk constraint to db table col

---

if u r working with existing db table... just colnames and prpperty names sufficeint in mapping file as shown below..

```
<hibernate-mapping>
  <class name="com.nt.entity.Project" table="PROJECT">
    <id name="projId" column="PROJID" />
    <property name="projName" column="PROJNAME" />
    <property name="teamsize" column="TEAMSIZE" />
  </class>
</hibernate-mapping>
```

saving or updating object (using ses.saveOrUpdate(-) method)

=> if object/record is already available then it will update the record otherwise it will insert/create new record /object..

signature ::
public void saveOrUpdate(Object obj)

usecase:: membershop registration/updation

To make our gradle project working with java latest version (other than default 8 version)

step1) make sure that latest version (jdk 14) is installed in u r computer
step2) add the following two entries in build.gradle file..
sourceCompatibility = "14"
targetCompatibility = "14"
step3) bind java /jdk latest version with eclipse Project..
right click project --->build path ---> cfg build path---> libraries tab--->
module path --> edit ---> add-->Installed JRES ---> add ---> select java14/jdk14
installation folder --> ....
step4) Change Project facade java version to 14 .. using project properites....
step5) Perform Gradle Refersh on Project once..

Version1 (With out UnSaved Value) *(Best practice)*
==========

```
//prepare object
        Memership member=new Membership();
        member.setMid(9999999999L);
        member.setName("rajesh");
        member.setAddrs("hyd");
        member.setRewardPoints(17L);
        try {
            //begin Tx
            tx=ses.beginTransaction();
              ses.saveOrUpdate(member);
              flag=true;
        }//try
        catch(HibernateException he) {
            flag=false;
            he.printStackTrace();
        }
        finally {
            //perform tx mgmt
            if(flag) {
                tx.commit();
                System.out.println("Object is saved or updated");
            }
            else {
                tx.rollback();
                System.out.println("Object is not saved or updated");
            }
            //close objs
            HibernateUtil.closeSession(ses);
            HibernateUtil.closeSessionFactory();
        }
```

=>This method generates select and insert/update sql queries
=> if the given object id value based record is already available and updation is not required then it just generates sleect query (no update query)

version2 (with unsaved-value)
=======

In mapping file
==========

```
<id name="mid" column="MID" unsaved-value="8889">
    <generator class="increment"/>
</id>
```

Client app code
==========
same as version1 code

=> if the unsaved-value is matching with Entity object id proerty value then it performs direct insert operation either using generator (like increment) generated value as the id value or id property value as the id value(if no generator is cfg).

=> if the unsaved-value is not matching with Entity object id proerty value then it performs direct update operation by taking given Entity object id value as the criteria value/condition value (if record is not avaiable to update then exception will be raised)

note:: In both casses "no-select Query will be generated -- only direct insert/update query will be generated)
note:: unsaved-value is use-ful to guide hibernate to generate update/insert query directly with out checking the record availability using select query..

means this attribute will not be saved in Db table at any cost.. but will be used by hiberanate to generate insert/update query dynamically with out generating select SQL query while working with ses.saveOrUpdate(-) method

note:: Working version2 is bad practice becoz changing unsaved-value attribute value in mapping file based on Entity object id value is very complex..

understanding ses.merge(-)
==========

=> It can be used two angles
version1) as alternate ses.saveOrUpdate(-) method
versno2) To merge given object data to already loaded same object data..

signature :: public Object merge(Object obj)

version1) as alternate ses.saveOrUpdate(-) method (It is same ses.saveOrUpate(-) version1)

```
member=new Membership();
  member.setMid(4L);
  member.setName("rajesh");
  member.setAddrs("hyd1");
  member.setRewardPoints(43L);
  try {
      //begin Tx
      tx=ses.beginTransaction();
       m1= (Membership) ses.merge(member);
       System.out.println(m1);
       flag=true;
  }//try
  catch(HibernateException he) {
      flag=false;
      he.printStackTrace();
  }
  finally {
      //perform tx mgmt
       if(flag) {
           tx.commit();
           System.out.println("Object is saved or updated");
       }
       else {
           tx.rollback();
           System.out.println("Object is not saved or updated");
       }
       //close objs
       HibernateUtil.closeSession(ses);
       HibernateUtil.closeSessionFactory();
  }
```

=>ses.merge(-) method is no way related to "unsaved-value" cfg in the <id> tag..

=> In this version1 .. if id property value of Entity object based record is not available in Db table then it perform insert opertion otherwise it performs update operation by taking same id value as the criteria value

=>In this version1, ses.merge(-) returns new Object representing the new to be inserted or existing record to be updated when tx.commit() is executed...

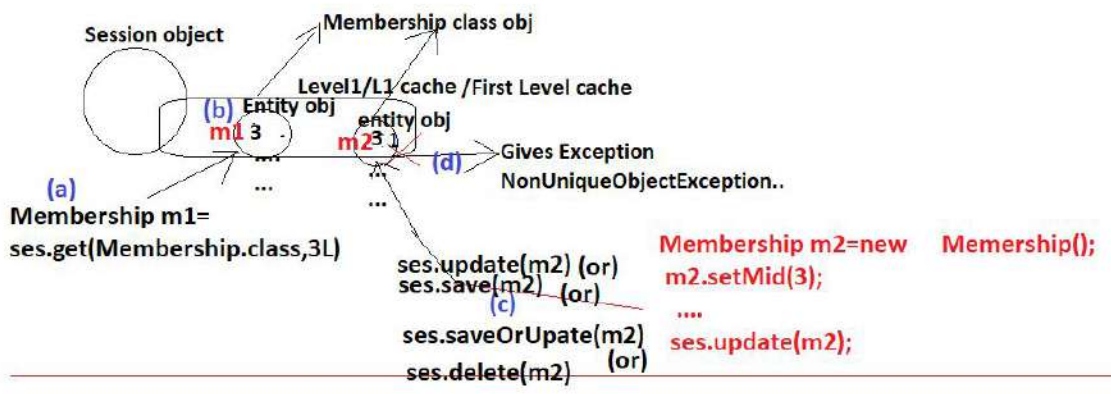
=> In this version1 , It uses select query + insert/update query to complete the task..

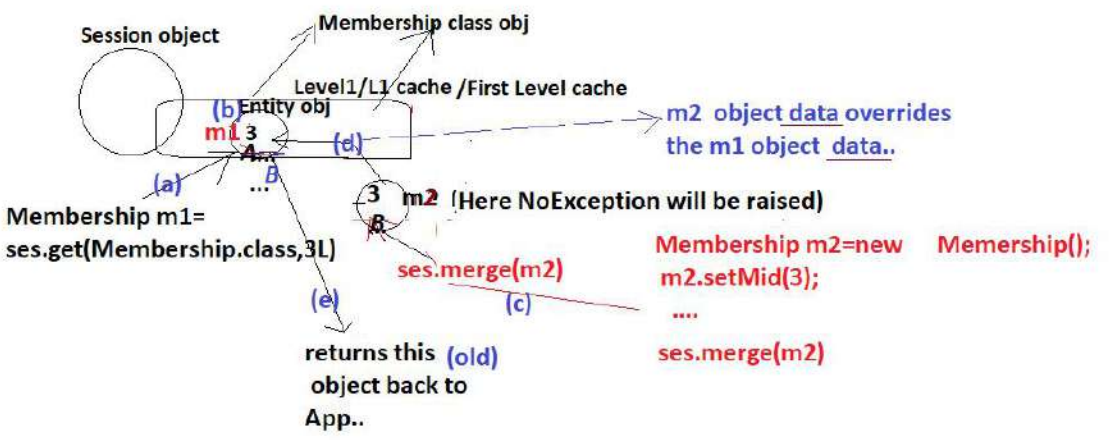
---

version2 (For merging existing object data with given new object data)

=>when we call ses.load(-,-)/ses.get(-,-) methods the entity object that is created having loaded record will be maintainer Level1 cache for reusability...
=> when we call ses.save(-)/ses.persist(-)/ ses.update(-)/ ses.delete(-)/ses.saveOrUpdate(-) methods with an entity object then that object will be placed in Level1 cache until tx.commit() method is called..

if Session object Level1 is already having Entity object with certain Id value.. the placing another object of same Entity class with same id value by using ses.update(-) /ses.save(-)/ses.persist(-) / ses.delete(-) /ses.saveOrUpdate(-) throws "NonUniqueObjectException" or javax.persistence.EntityExistsException (incase of ses.delete(-))
becoz Level1 cache of Session object allows us to keep only unique objects..






```
//Load object
member=ses.get(Membership.class,3L);
System.out.println(member);
try {
    tx=ses.beginTransaction();
    member1=new Membership();
    member1.setMid(3L);
    member1.setAddrs("vizag");
    member1.setName("suresh1");
    member1.setRewardPoints(47L);
    //ses.update(member1); //throws NonUniqueObjectException
    //ses.save(member1); //throws NonUniqueObjectException
    //ses.delete(member1);//throws EntityExistsException
    //ses.saveOrUpdate(member1);  //throws throws NonUniqueObjectException
    member2=(Membership) ses.merge(member1);
    System.out.println(member2);
    System.out.println(member.hashCode()+" "+member1.hashCode()+" "+member2.hashCode());
    flag=true;
}
catch(Exception e) {
    e.printStackTrace();
    flag=false;
}
finally {
    if(flag) {
        tx.commit();
        System.out.println("Object updated");
    }
    else {
        tx.rollback();
        System.out.println("Object not updated");
    }
    HibernateUtil.closeSession(ses);
    HibernateUtil.closeSessionFactory();
}
```

usecase:: if ordinary passeger reserves VIP seat in flight.. the moment VIP books same seat then Oridary passenger details will be overridden with VIP details and seat will be confirmed for VIP.

what is the diff b/w ses.update(-) and ses.merge(-)?

ses.update(-)	ses.merge(-)
(a) performs only update object operation	(a) can perform insert/update object operation based on the availability of the record.
b) when FirstLevel cache of Session is already having Entity object of certertain class, if we try put another Entity object of same class with same id value to the same FirstLevel cache by calling ses.update(-) method then we will get NonUniqueException object	(b) if we call ses.merge(-) for the same situation then the new object data will override/merge with already available entity object data of First Level cache with out throwing any exception.
c) update(-) return type is void , So does not return any thing	c) merge(-) return type is java.lang.Object i.e means it returns old object having overriden values of new object data (already available objcet in cache)
d) Directly generates update SQL query	d) generates select query + insert/update query when it used alternate to ses.saveOrUpdate(-) otherwise it just generates update query to modify already avaiable object data with new object data...
e) Generates update SQL query irrespective of updatation is required or not	e) Generates update SQL query only when updation is required..
f) if the record is not avaiable to update then throws exception.. (assume ses.update(-) called is directly)	f) if the record is not avaliable to update/merge then it will will perform insert operation..

Can we perform non-select opreations without in hibernate using TxMgmt code?

Ans) yes, possible upto hibernate 4.x using ses.flush() .. which will sync DB s/w with all Persistence instructions that are there with session object.. i.e all pending persistence instructions that there in session obj will be completed in Db s/w. but it is not a recomanded process..

```
//prepare object
        member=new Membership();
        member.setMid(56L);
        member.setName("rajesh");
        member.setAddrs("hyd1");
        member.setRewardPoints(43L);
        try {
            idVal=(long) ses.save(member);
            System.out.println("Generated id value::"+idVal);
            ses.flush();
        }//try
        catch(HibernateException he) {
            he.printStackTrace();
        }
        finally {
            //close objs
            HibernateUtil.closeSession(ses);
            HibernateUtil.closeSessionFactory();
        }
```

note:: from hiberante 5.x the TxMgmt is made mandatory for non-select Persistnece operations..

F3:: To get Source code
F4:: To see hierarchy of given class/interface/....
ctrl+shift+T :: to open source code any class .. which is not there in the current editor

---

Dynamic Schema Generation/ hbm2ddl.auto property

=>This feature makes hibernate f/w to genrate DB tables dynamically (schema) based on the info given in mapping files... To use feature it is recomanded to specify type, length , not-null, unique and etc.. attribute in mapping file..

=>It is very useful if change Db s/w in the middle of development or production.. to generate DB tables in new DB s/w dynamically though we do not know much about the SQL part of new DB s/w..

=> It is very useful the Project DB s/w is changing

development (h2) / mysql	to testing (mysql)	to UAT/pilot (oracle xe)	to Production oracle(EE)
deveopers in S/w comapny	(testers in the s/w company)	Client org people after realeasing and before using project in Production	when project in live for the client org customers

jdbc, hibernate (infact ORMs) are given to work with only SQL Db s/ws... (RDBMS Db s/w)

The possible vlaues for "hbm2 ddl.auto" property (hibernate cfg file)

a) create
b) update (best)
c) validate (default)
d) create-drop

<property name="hbm2ddl.auto">create </property>

=>Always creates new Db tables based on mapping file info .. if Db tables are already available they will be deleted..
=>Internally uses SchemaExport tool..
=> if u r having long running apps with out shutdow.. like Railway ticket reservation App then use "create".. Once the App is restarated we will loose old data.. becoz it creates new db table by dropping existing db tables . (railway ticket booking app erase the data of bookings for every 3 years)

=>It is recomanded to place "length" , type(hiberante type),unique, non-null and etc.. attributes in <property> tags of mapping file (hbm file) to generate Schema (db tables, sequences and etc..) more effectively.. The column name specified in <id> tag automtically becomes pk colum.. so there is no need of placing unique="true" , not-null="true" attributes in <id> tag.

=> If mapping file does not contain col names , table name .. then Entity class property names will be taken as col names and entity class name will be taken as db table name..

=>Hibernate data types are bridge data types b/w java data types and underlying Db s/w data types

java data type	hibernate data type	oracle data type (SQL data type)
int	int	number(-) ,int
short	short	number(-)
float	float	float, number(-,-)
java.lang.String	string	varchar, varchar2
java.sql.Date	date	date
.	.	.
.	.	.
.	.	.

456677.66
scale precision

note ::"length" attribute kept in mapping file will work effectively only for String/text type colums.. not for numberic data type cols.. They have done this to keep java data type range values while setting data to numberic data type proeprties of Entity class objects...

<property name="hbm2ddl.auto">validate</property>

=>Verifies wheather Db table is there according mapping file configurations.. or not if not it will throw "org.hibernate.tool.schema.spi.SchemaManagementException: Schema-validation: "

==> As part validiation/verification it will check Db table, cols availability, data types matching , col count matching (for less cols no error.. for more cols error will be generated) and etc...

==> This option does not attempt to create or alter or drop the db tables..

<property name="hbm2ddl.auto">update</property>

=> Internally uses SchemaUpdate tool
=> creates the Db table if the db table is not already available
if Db table already available matching mapping file info then it uses that db table
if Db table already available not matching mapping file info then it can alter db table
(This altering is possible only by adding new cols to the db table for new proerties cfg in the mapping file i.e it can not drop exisitng cols , can not modify existing col data types)

=> This most reguraly used in real time.

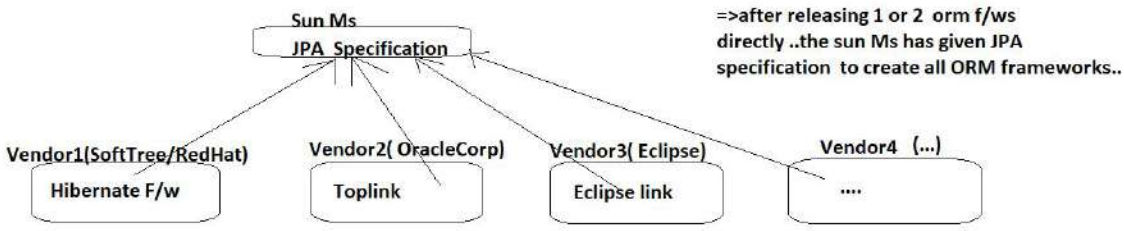
=>Banking Apps use this option (especially useful for linking adharNo with Bankaccount by adding adharNo col to existing db table)

<property name="hbm2ddl.auto">create-drop </property>

=> Internally uses SchemaExport tool ....
=> Creates db tables based on mapping files info .. when the SessionFactory object is created ... and delets the same db table when session factory is destroyed...
=>This is very useful while testing projects , while giving demo of the project and also in UAT .. becoz in all these places .. Db tables and its data is required on temporary basis..

JPA (Java Persistence API)

JPA is s/w specification that provides bunch of rules and guidelines to devleop ORM s/ws or tools or frameworks...



=>after releasing 1 or 2 orm f/ws directly ..the sun Ms has given JPA specification to create all ORM frameworks..

We can develop hibernate apps in two ways

a) xml driven configurations (Both hibernate cfgs and mapping will be done by xml files like hibernate cfg file, hibernate mapping file

b) annotation driven configurations (We should we use mapping annotations in Entity class by replacing hibernate mapping file. Still we need to use hibernate cfg file as xml file)

Why Hibernate cfg file (xml file) is not replaced annotations?

Ans) cfg file contains connecton properties , hibernate properites holding jdbc driver details , dialect details and other details which will be changed time to time or env.. env.. Using annotations we can not flexiblity of modification becoz they are statements and should be written in java code... So they are still continuing hibernate cfg file as the xml file..

Why they are giving mapping annotations as alternate to HB mapping file?

Ans) Once the DB table designing is frozen, it will not changed much.. more over Hibernate can generate Schemas (db table and etc..) dynaically, So flexiblity of modification is not required that much.. For this they have replaced mapping file with annotations.

How can override Mapping Annotation cfgs done in Entity classes with out touching its source code?

Ans) Use Hibernate mapping file .. to override mapping annotaiton cfgs..

note:: xml files are far better when compare to properties files /yml files.. becoz xml file gives hierarchy of data supports, xsd/dtd rules support and etc..

note:: Performing code about code (metadata operations) is always good through annotations when compare to xml files..

While writing annotaitons give following prioirty

a) JPA Mapping annotations (given by Sun Ms) (eg:: @Table, @Column ,@Id and etc..)
b) Hibernate Mapping annotations (given by hibernate) (@Filter , @FilterDef and etc..)
c) Java Config annotations (given by jse, jee) (@Resource , @PreDestroy ,@PostConstruct ...)
d) Third party Annotations

note:: we can link hbm files to hb cfg file using "resource" of <mapping> tag
eg:: <mapping resource="com/nt/entity/InsuracePolicy.hbm.xml">
we can link Entity classes with mapping annotations to HB file using "class" attribute
<mapping > tag in Cfg file

<mapping class="com.nt.entity.InsurancePolicy"/>

Basic JPA Mapping Annotations

==========

@Entity ---> To mark given java class as model class/entity class/Persistence class (mandatory) | <class>
@Table ---> To map entity class with Db table (optional)
@Id ----> To mark Entity class property as Id property (<id> tag) (mandatory)
@Column ---> To map each property with Db table column (optional)

=>we can add mapping annotations either on Entity class properties or on getter methods (better) of Entity class

```
@Entity
@Table(name="PRODUCT")
public class Product implement Serializable{
 private int pid;
 private String pname;
 private float qty;
 private float price;

 //write setters
  .....
   ...
 //write getters

 @Id     @Column(name="PID")
 public int getPid(){
   return pid;
 }

 @Column(name="PNAME")
 public String getPname(){
   return pname;
 }

 @Column(name="PRICE")
 public float getPrice(){
  return price;
 }

 @Column(name="QTY")
 public float getQty(){
  return qty;
 }
```

we can not use lombok api here

=>Gives good performence becoz reading annotaiton from public getter methods is easy..

```
@Entity
@Table(name="PRODUCT")
public class Product implements Seriliazable {
   @Id
  @Column(name="PID")
   private int pid;
@ Column(name="PNAME")
   private String pname;
@ Column(name="QTY")
   private Float qty;
@Column(name="PRICE")
   private Float price ;
   //setters && getters
   .....
    ....
}
```

=>if entity class name is matching Db table name then @Table is optional .. Similarly if Property names are matching with Col names then @Column is optional..

We can use lombok api..

=>Gives bit slow performence.. becoz accesing annotations from private properties /member variables of class needs lots of reflection api code like setAccessible(true)

Why they are not allowing annotations on the setter methods of Entity class?

Ans) some times we can avoid setter method in the entity class by taking parameterized constructor to set data to object.. So adding annotations on the top setter methods is bit risky.. For this reason ,they are allowing only on getter methods..

JPA mapping annotations based entity class becomes portable across the multiple ORM Framworks becoz all ORM frameworks given based JPA Specification..

=>ctrl+shift+x :: To selected content as upper case content
=>ctrl+shift+y :: To selected content as lowercase content

more addtions
==========
@Type (HB) --> To specify hibernate data type
@Transient --> To make certain proeprty not parcipating persistnece operations

```
@Column(name="COMPANY",length = 20,unique=true, nullable=false)
 @Type(type="string")
 @Transient
 public String getCompany() {
        return company;
 }
```

note:: It is not recomanded to apply few annotations at field level and few other annotations at getter metthods level...

note:: applying @Transient on top of @Id Property is always invalid combo.. becoz pk col can not have null values...

To enable dynamic insert ,dynamic update based insert ,update SQL query generatation as alternate dynamic-insert="true" and "dynamic-update="true"' of <class> tag we can use @DynamicInsert , @DynamicUpdate annotatons (HB)

```
@Entity
@DynamicInsert(true) or @DynamicInsert(value=true)
@DynamicUpdate(true)
@Table(name="PROJECT")
public class Project implements Serializable {
 ....
  ...
}
```

What is diff b/w @Transient and @DynamicInsert /@DynamicUpdate annotations?

Ans) @Transient makes Hibernate f/w not participate certain property of entity class in Persistnece operations.. through that Property is filled up with data ... This useful to keep certain property and its column not participating in all Persistence operations.. on temporary basis

@DynamicInsert/Update makes Hiberante Dialect to generate SQL query dynamically only by involving those properties whose values are not null.

@Transient is applicable at filed , method level .. i.e we can make our choice properties as transient properties
@DynamicInsert/Update is applicate only at class level.. i.e we developer can not decide properties of dynamic query generation.. (they will be decided based on the null values we set to Entity class object)

@Transient makes Dialect not to involve its property pre-genrated SQL queries given by Dialect where as @DynamicInsert/Update makes Dialect to generate SQL queries dynamically Query..

@Transient stops all persistence operations(CURD) on that Property... where as @DynamicXxx annotations are there only for dynamic insert,update queries generation..

note:: @Transeint proeprties do not participate in Dynamic query generation..

How many types of annoations in hibernate programming? (Based on Persistence activities)

a) Persistable annotations
=>@Table , @Column , @Id ,@Entity and etc..
b) Non -Persistable Annotations
=>@Transient , @Filter ,@FilterDefs and etc..

(Author specific categorization)

=>In mapping file we use lazy="true" or lazy="false" of <class>
to make ses.load (-,-) to perform lazy loading or eager loading respectively..
=> The above effect we can get by using @Proxy annotation of HB ..

@Proxy (lazy="true") --> makes ses.load(-,-) to perform lazy loading (default)
@Proxy (lazy="false") --> makes ses.load(-,-) to perform eager loading

```
@Entity
@Proxy(lazy = false)
public class Project implements Serializable {
   ...
   ...
}
```

In the above setup , if we take entity class as final class .. then it perform eager Loading though we take @Proxy(lazy=true) becoz Proxy class generation is failed as the sub class of Entity class as we have taken entity class as the final class.. To overcome this Problem we can make Hibernate to generate Proxy class as the impl class of ProxyInterface by specifying the interface name in "proxyClass" attribute of @Proxy ..

IProject.java
==========

```
package com.nt.entity;
public interface IProject {
      public String getProjName() ;
      public void setProjName(String projName) ;
      public Integer getTeamSize() ;
      public void setTeamSize(Integer teamSize);
      public String getCompany();
      public void setCompany(String company) ;
      public void setProjId(Long projId) ;
      public Long getProjId();
}
```

```
@Entity
@Proxy(lazy = true,proxyClass =IProject.class )
public final class Project implements Serializable,IProject {
   ....
   ....
}
```

note:: There no annotation support to bring the effect of "unsaved-value" attribute of <id> which we use while working with ses.saveOrUpdate(-) method.

Surrogate key colum and natural key column
==========

=>The cols in Db table whose values are unique and can be used to indentify and access records in called candidate key colum..

PersonInfo (Db table)
==========

->aadharNo (candiate key col)
-> name
-> addrs
-> dlNo (candiate key col)
-> DOB
->gender
->panNo (candiate key col)
->voterId (candiate key col)
->PassportNo (candiate key col)

Natural key colum
==========

=>The candiate key colum which is having outside business meaning and can be used to indentity the record inside the Db s/w and outside the Db s/w is called natural key column .these values must be collected from enduser..
eg: panNo , voterId , passportNo, DLNo , aadharNo and etc./

Surrogate key column
==========

The candidate key column whose values does not have business meaning and generated by the underlying App/Project or DB s/w dynamically is called Surrogate key column .. These values will not be collected from endusers...

eg1: sequence in Oracle to student number or serial no
eg2: autoincrement col in mysql
eg3:: Generators in hibernate... (like increment, identity , hild and etc..)

Limitations of taking natural key column as PK column
==========

a) values are very lengthy , So more memory is required
b) Since values are having business meaning .. if they are changed becoz of outside world policies then distrub main and dependent db tables and relevant java code .. (This is very costly)
c) these values are expected from Enduser.. if enduser fails to give them record insertion fails..

Avantages of taking surrogate key column as PK column
==========

a) values are very small ,So takes less memory
b) values do not have business meaning of outside world.. So these values will not change for any reason and will not distrub other db tables and relevant java code..
c) Not expected from enduser.. So record inserations are possible with the limited data given by enduser.

note:: Always take surrogate key column as the PK column..

Generators in Hibernate
==========

=>These are different algorithms given Hibernate and JPA using different DB concepts to the generate the values to the Id Property of Entity class obj/PK col of Db table dynamically.. Indirectly it is taking Surrogate key colum as PK column... i.e vlaues to PK col will be inserted dynamically with out any business meaning..

3 types of Generators in hibernate
(a) Hibernate supplied generators (Specific Hiberante F/w) (12)
i) Using xml driven cfgs (<generator> in mapping file)
ii) Using annotation driven cfgs (@GeneratedValue ,@GenericGenerator)

(b) JPA Generators (common for all ORM f/ws) (4)
(@GeneratedValue and other annotations)

(c) Custom Generators

Hibernate Supplied generators
==========
assigned,increment,identity , sequence, hilo, seqhilo ,native , foreign , select ,guid,uuid,sequence-identity
(defualt) (removed from HB 5.x)

While choosing any Generator make sure
(a) The value and length of generator generated value is compitable to store in Id property
(b) underlying Db s/w supports the generator related DB concepts..

All generators are HB supplied java classes implementing org.hiberante.id.IdentifierGenerator (I) like IncrementGenertor,SequenceGenerator and etc... but we can use either their class names or their short names.. while cfg them in <generator class="....."/>
shortname or classname of generator

<generator class="increment"/> is equal to
<generator class="org.hibernate.id.IncrementGenerator"/>

(a) Hibernate supplied generators (Specific Hiberante F/w) (12)

assigned
=======
=> this is default generator
=> makes programmer to set value to id property before calling ses.save(-) method
=> It makes natural key col as pk col becoz it is expecting id value either from enduser or from programmer so Not recomanded to use..
=> allows to take any type of property as id property..
=> Works with all Db s/ws..

```
eg:: <id name="projId" column="PROJID">
        <generator class="assigned"/>
     </id>
```
(or)
```
eg:: <id name="projId" column="PROJID">
        <generator class="org.hibernate.id.Assigned"/>
     </id>
```

usecases:: To choose fancy mobile numbers, vehice registration numbers and etc.

---

increment

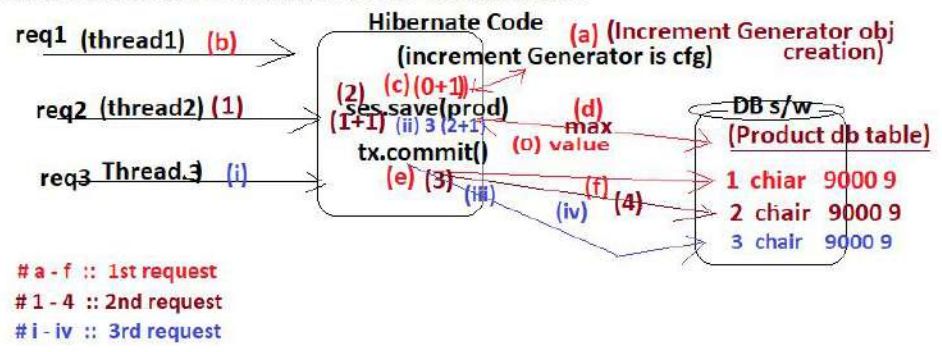
=> use max val+1 formulae to generate id value of type int ,short,long ...
=> Works with All Db s/ws
=> if Db table is empty then it generates 1 as the identity value..
=> Suitable in Multi-threaded env.. but not suitable in clustered Env...
(running the same App by keeping in multiple computers having load-balance)

```
<id name="pid" colum="PID">
   <generator="increment"/>
</id>
```

=>The Objects for configured Generator classes will be created while preparing SessionFactory object ... i.e for every Generator class only object will be created.. with repsect to each mapping file/entity class..

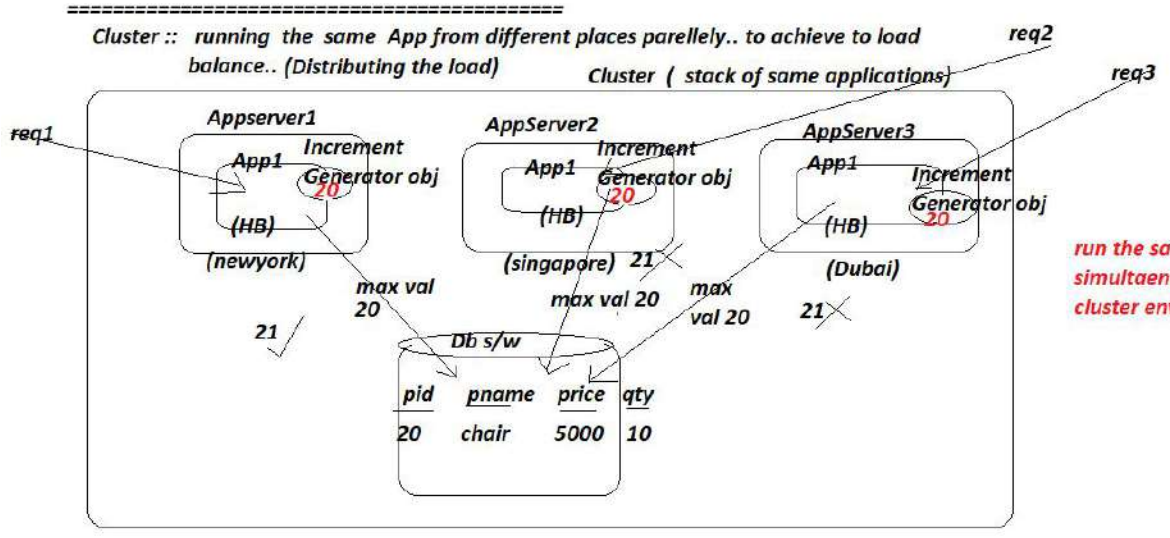
=>The Increment generator hits the Db s/w only for 1 time to get Max value from Db table and increments by 1 for first ses.save(-) method call.. for next ses.save(-) method calls with the same execution of the App it uses previous val+1 to get the identity value..

=> Increment Generator is suitable in multi-threaded env..



Increment Generator is not suitable in clustered env..
==========

Cluster :: running the same App from different places parellely.. to achieve to load balance.. (Distributing the load)



run the same client app for multiple times simultaenously.. to feel the problem in cluster env...

note::: do not use "increment" generator in real projects especially in Clustered env.. take support of "indentity" or "sequence" generators which works nicesly in clustered env..

---

3) identity generator
==========
=> Generates the id values of type int or long or short by using identity col support given by underlying Db s/w like mysql, DB2, MS SQL Server, Sybase and HypersonicSQL and etc...
=> This generator can be used only in that Db s/w that supports identity cols..
=> The col whose value will comes based on <prvious value>+1 formulae when we insert values to other columns of the Db table.
=> This generator does not work in oracle,postgresql DB s/ws..as they do not support identity cols...

note:: In Mysql DB s/w the "autoincrement" constraint makes the col as identity column..
=> This generator makes DB s/w to generate the id value..
=> It is suitable for multithreaded env.. and clustered env...

In mysql DB s/w setup
==========
->add autoincrement constraint to pid column product (use mysql Work bench)
->open properties of the product table and select AI checkbox for pid column
-> in Mapping file

```
<id name="pid" column="PID" type="int">
    <generator class="identity"/>
</id>
```

=>oracle 12c onwords identity columns support is given , so we can use "identity" generator from oracle 12c onwords..

What is the difference b/w increment and identity generators?

increment	identity
a) Works with all Db s/ws	a) works only in that Db s/w which supports identity cols like mysql .. (does not work with oracle (upto 11g) and postgreSQL)
b) uses max val+1 to generate first id value later uses preval+1 to generate next id values..	b) uses previous val +1 to generate the id value
c) makes Hiberante to generate and assign id value	c) makes underlying Db s/w to geneate id value , HB collects that id value and assigs to id property.
d) first generates id value and uses that in the insertion of the record	d) while inserting the record id value is generated and that will be assigned to id property..
(e) considers deleted values while generating new id value using max val+1 formulae	e) ignores the deleted id values while generated new id vlaue becoz it uses previous val +1 as the formulae..
(f) can not be used in Clustred env.. of Application development (Distributed Env..)	(g) can be used...
(g) In the execution of same App, this generator hits DB s/w only once to get the max pk col value and uses it for multiple records insertion by using max val+1 (for first id value geneation) and previous val+1 (for next id value generation)	g) Hits the DB s/w to get each identity value while inserting each record..

sequence generator
==========

=> uses a sequence in DB2, PostgreSQL, Oracle, SAP DB, McKoi or a generator in Interbase. The returned identifier is of type long, short or int

=>upto oracle 11g there is no support for identity cols.. similary mysql does not support "sequences"

=> if no sequence name is specified it generates one default sequence "hibernate_sequence" having logic start with 1 increment by 1. (hbm2ddl.auto property must set with update/create")

```
<id name="pid" column="PID">
   <generator class="sequence"/>
</id>
```

change the setup to oracle..

=>This gernator makes HB to insert sequence generated value as the PK col value while inserting record.

=> It is suitable for multi-threded env.. also in clustered env....

note:: Realtime does not use "increment" they use either "sequence" or identity based on capabilities of underlying Db s/w..

note:: sequnece, identity,increment generators are useful to generate serialno, billNo, TxId and etc.. id values without any business meaning..

To work with custom sequence
==========

step1) create custom sequence in oracle

```
SQL>CREATE SEQUENCE "SYSTEM"."PID_SEQ" MINVALUE 100 MAXVALUE 3000
INCREMENT BY 10 START WITH 100;
```

step2) cfg sequence generator in mapping file by specifying the custom sequence name

```
<id name="pid" column="PID" type="int">
    <generator class="sequence">
        <param name="sequence">PID_SEQ</param>
    </generator>
</id>
```

This is param name to supply
the custom sequence name
in some versions(5.0/1/2/3 it is "sequence_name"

hilo
====

uses a hi/lo algorithm to efficiently generate identifiers of type long, short or int, using given a table name and column value (by default hibernate_unique_key(default table name) and next_hi(default col name) respectively) as the source of hi values and max_lo param value as the source of low value. The hi/lo algorithm generates identifiers that are unique only for a particular database.

source of hi value :: given db table col value
source of low value :: max_lo param value..

In mapping file
==========

```
<id name="pid"  column="PID">
   <generator class="hilo">
         <param name="table">HI_TAB </param>
         <param name="column">HI_COL </param>
         <param name="max_lo">5 </param>
   </generator>
</id>
```

source of low value

DB table in any Dbs/w
==========

HI_TAB (user-defined db table)

HI_COL

~~100~~ (source of hi value)
~~101~~
102

(For each record insertion the source of hi value in Db table will be incremented by 1)

formuale
==========

<hi value> * (max_lo param val+1)

100 * (5+1) = 600
101 * (5+1) = 606
102 * (5+1) =612

(or)

(hi val* max_lo param val) + hi val

100 * 5 + 100 =600
101 * 5 + 101 = 606
102 * 5 + 102 =612

=>Hilo generator is removed from hibernate 5.x (becoz of its complexitity) so work with 4.x
=> works with all DB s/ws...
=> Suitable for multithreaded env.. and also for clustred env..

if we do not specify hi value related custom db table name and col name then hibernate dialect dynamically generates default table hiberante_unique_key ,column next_hi as the source of high values

```
<id name="pid"  column="PID"  type="int">
      <generator class="hilo">
         <param name="max_lo">5</param>
      </generator>
   </id>
```

hibernate_unique_key (db table)
==========

next_hi

~~1~~ ~~2~~ 3

<hi value> * (max_lo param val+1)

1
1 * 5+1 = 6
2 * 5+1 = 12
3 * 5+1 =18

(hi val* max_lo param val) + hi val

1
(1 * 5) +1 = 6
(2 * 5) + 2 = 12
(3 * 5) +3 =18

```
<id name="pid"  column="PID"  type="int">
      <generator class="hilo"/>
   </id>
```

hibernate_unique_key (db table)
==========

next_hi

~~1~~ ~~2~~ 3

<hi value> * (max_lo param val+1)

1
1* (32767 +1) = 32768
2* (32767 +1) = 65536

(hi val* max_lo param val) + hi val

1
(1 * 32767) +1 = 32768
(2 * 32767) +2 = 65536

the defaul value of max_lo param value is :: 32767

In realtime "hilo" generator is used for batch insertion where <"max_lo" param>+1 acts as batch size.. This is useful group ticket reservation to generate batch sequence of ticketIds to passengers..

Mapping file
==========

```
<id name="pid"  column="PID"  type="int">
      <generator class="hilo">
         <param name="table">HI_TAB</param>
      <param name="column">HI_COL</param>
         <param name="max_lo">5</param>
      </generator>
   </id>
```

DB table in any Dbs/w
==========

HI_TAB (user-defined db table)

HI_COL

~~100~~ (source of hi value)
~~101~~
102

```
try {
    tx=ses.beginTransaction();   //internally calls  con.setAutoCommit(false) to begin the Tx
            //save object
          for(int i=1;i<=10;++i) {
                prod=new Product();
                    //prod.setPid(9010);
                     prod.setPname("chair1");
                     prod.setPrice(40000.0f);
                     prod.setQty(80.0f);
           idVal=(int)ses.save(prod);
           System.out.println("Generated id values ::"+idVal);
           flag=true;
          }
      }
      catch(HibernateException he) {
            he.printStackTrace();
            flag=false;
      }
      finally {
            //commit or rollback  Tx
            if(flag==true) {
                  tx.commit();  //internally calls con.commit()
         System.out.println("Object is saved");
            }
            else {
                tx.rollback(); //internally calls  con.rollback()
                 System.out.println("Object is not saved");
            }

            //close  session object
            HibernateUtil.closeSession(ses);
            //close SessionFactroy
            HibernateUtil.closeSessionFactory();
      }//finally  */
```

idvalues :: 600
601
602
603
604
605

here gives 6 insert queries

---

id values :: 606
607
608
609

here gives 4 insert queries

note:: for better results use "seqhilo" in the place "hilo" generator which is not removed in 5.x

seqhilo
=======

=>Uses the hilo algorithm to gnerate the id values of type int ,long or short by taking sequence generated value as the source of high value and max_lo param value as the source of low value..
=> If no sequence is specified then it takes hibernate_sequence as the default sequence name.
=> This generator works only in the Db s/ws who support sequences.. like oracle ...
=> Suitable for multithreaded env.. and also suitable for clustered env..
=> It works in all versions of hibernate f/w..

eg1::

```
<id name="pid" column="PID" type="int">
    <generator class="seqhilo">
      <param name="max_lo">10</param>
    </generator>
  </id>
```

uses hiberate_sequnece generated value as the source of hi value..

formuale (same as hilo)
==========

sequence generatd hi value * (max_lo param value +1)

1 * (10 +1) =11
2 * (10+1) =22
3 * (10+1) =33

(<sequence generaed hi value>* <max_lo param value>) + <sequence generated hi value>

(1 * 10) + 1 = 11
(2 * 10) + 2 = 22
(3 * 10) + 3= 33

---

eg2:: Using custom sequence ...

SQL>CREATE SEQUENCE "SYSTEM"."PID_SEQ"
MINVALUE 10 MAXVALUE 1000 INCREMENT BY 1

```
<id name="pid" column="PID" type="int">
    <generator class="seqhilo">
      <param name="sequence">pid_seq</param>
      <param name="max_lo">10</param>
    </generator>
  </id>
```

sequence generatd hi value * (max_lo param value +1)

10 * (10+1) =110
11 * (10+1) =121
12 * (10+1) =132

(<sequence generaed hi value>* <max_lo param value>) + <sequence generat

(10 * 10) +10 = 110
(11 * 10) +11= 121
(12 * 10) +12= 132

eg3:: seqhilo is very useful for Batch record insertion...

In Mapping file
==========

```
<id name="pid" column="PID" type="int">
    <generator class="seqhilo">
      <param name="sequence">pid_seq</param>
      <param name="max_lo">10</param>
    </generator>
  </id>
```

In client app
==========

```
try {
tx=ses.beginTransaction();   //internally calls con.setAutoCommit(false) to
begin the Tx
      //save object
    for(int i=1;i<=20;++i) {
        prod=new Product();
      //prod.setPid(9010);
      prod.setPname("chair1");
      prod.setPrice(40000.0f);
      prod.setQty(80.0f);
  idVal=(int)ses.save(prod);
      System.out.println("Generated id values ::"+idVal);
      flag=true;
    }
}
```

native (Generator)

=>Picks up the hilo, identity or sequence generator based on the capabilities of underlying Db s/w
=> In case of oracle , it uses sequence generator
=> in case of mysql , it uses indentity generator
=> This generator is not having any class name..
=>if any Db s/w, does not support sequences or identity cols there it will take support of "hilo" generator.
=>we can use in multi-threaded, clustered env...

#for oracle

```
<id name="pid" column="PID" type="int">
    <generator class="native"/>
</id>
```

uses the default sequence hibernate_sequence

#for oracle to use custom sequence

```
<id name="pid" column="PID" type="int">
    <generator class="native">
        <param name="sequence_name">pid_seq</param>
    </generator>
  </id>
```

---

mysql Db s/w (internally uses identity generator)

```
<id name="pid" column="PID" type="int">
    <generator class="native"/>
  </id> -
```

---

uuild (universal unique id)

=>uses a 128-bit UUID algorithm to generate identifiers of type string that are unique within a network (the IP address is used). The UUID is encoded as a string of 32 hexadecimal digits in length.

this generated id value by using
a) IP address of computer
b) system date and time
c) current Thread/process info
and etc..

=>Works in multiple in Db s/ws...
=>Hiberante generates this values and gives to Db s/w for insertion...
=> suitable for multi-threaded and clustred env..

binary --> base2 (0,1)
octal ---> base8 (0 to 7)
decimal --> base10 (0 to 9)
hexadecimal -->base16 (0 to 9 and A to F)

```
<id name="pid" column="PID" type="string" length="35">
    <generator class="uuid"/>
  </id>
```

note:: take pid as the String property in Entity class..

---

guid
====

=>uses a database-generated GUID string on MS SQL Server and MySQL.
=>same as uuid .. but it is DB s/w generated id value..
=> It is Db s/w generated id value ...
=> works with all Db s/w...

```
<id name="pid" column="PID" type="string" length="40">
    <generator class="guid"/>
  </id>
```

guid,uuid generators are useful to gernated TxIds ... in real applications...

select
======

retrieves a primary key, assigned by a database trigger, by selecting the row by some unique key and retrieving the primary key value.

=>Works only in the Db s/ws which supports trigger like oracle...
=> suitable for multi-threaded and clustrered env...

Example
==========

#1) create sequence , trigger in oracle DB s/w...

```
CREATE OR REPLACE TRIGGER PID_TRIGGER BEFORE INSERT ON PRODUCT For EACH ROW
BEGIN
  SELECT PID_SEQ.NEXTVAL INTO :new.PID FROM DUAL;

END;
```

In SQL plus
==========

SQL> ed ---> type the above code ... file menu -->save --> exit
SQL> /
(trigger created)

IN SQL Developer
==========

Right click trigger ---> new --->
name :: pid_trigger
base type :: table
base object :: product
timing :: before
events :: insert (select it) --> ok..
and modify the code as shown above..

#2) cfg "select" generator in mapping file...

```
<id name="pid" column="PID">
    <generator class="select">
      <param name="key">pname</param>
    </generator>
</id>
```

The trigger inserted value into pk column will be retrieved by hibernate by taking this given canidate key column ."pname" (unique key column)
hibernate uses the recived pk col values as the id value of entity object

---

foreign
==========

sequence-identity
==========

will be discussed in future (In Assoication mapping)

Custom Id Generator in hibernate
==========

Since all pre-defined generator classes are implementation classes of IdentifierGenerator(I)... we can develope our custom Generator by the implementing the same interface...

Custom Id Generator to get Random number as Id value
==========

step1) keep any application ready...

step2) develop Custom Generator class by implementing org.hb.id.IdentifierGenerator(I)

RandomIdGeneator.java
==========

```
public class RandomIdGenerator implements IdentifierGenerator {
  public RandomIdGenerator() {
        System.out.println("RandomIdGenerator:: 0-param constructor");
    }
    @Override
    public Serializable generate(SharedSessionContractImplementor session, Object object) throws HibernateException {
  System.out.println("RandomIdGenerator.generate(-,-)");
        int idVal=0;
        idVal=new Random().nextInt(100000);
        return idVal;
    }

}
```

step3) cfg custom generator in mapping file..

```
<id name="pid" column="PID" type="int">
      <generator class="com.nt.generators.RandomIdGenerator"/>
</id>
```

note1:: custom generators are specific to each project
note2:: No short names fro custom generators..

Assignment
==========

Generate id values like this using Custom Generators
NIT001,NIT002 ..... NIT009,NIT010, ...

Annotation driven Generators
- Hibernate Generators: @GenericGenerator (HB) + @GeneratedValue(JPA)
- JPA Generators: @GeneratedValue (JPA)

Hiberante Generators cfg using Annotations
==========

we need to use @Id , @GenericGenerator , @GenerateValue annotations..
- @Id, @GenericGenerator: (HB annotation) — To cfg hiberate generators
- @GenerateValue: (JPA Annotation) — To instruct hibernate that this @Id property will have generator's generated value as the id value.

increment
==========

#1) keep app ready having annotations setup..

#2) Add the the following annotations on the top of @Id property.

```
@Id
@GenericGenerator(name="gen1",strategy ="increment")
@GeneratedValue(generator = "gen1")
private Integer pid;
```

logical name
HB generator name

identity
==========

#1) change to mysql Db s/w
#2) Take pk col as identity column (auto increment)
#3) cfg identity generator

```
@Id
@GenericGenerator(name="gen1",strategy ="identity")
@GeneratedValue(generator = "gen1")
private Integer pid;
```

sequence
==========

#1) change to oracle DB s/w..
#2) cfg sequence generator with/with out custom sequence name..

eg1::

```
@Id
@GenericGenerator(name="gen1",strategy ="sequence")
@GeneratedValue(generator = "gen1")
private Integer pid;
```

=>creates sequence with the name "gen1" having logic start with 1 increment by 1

---

eg2::

```
@Id
@GenericGenerator(name="gen1",strategy ="sequence",
                  parameters =@Parameter(name="sequence_name",value="PID_SEQ") )
@GeneratedValue(generator = "gen1")
private Integer pid;
```

(HB annotation alternate to <param>
=>uses custom sequence PID_SEQ

note:: some hibernate specific generators will not work annotation driven env.. becoz their parameters not recognized properly like "hilo","seqhilo" and etc.. So prefer using JPA Generators in Annotation driven hibernate Programming...

JPA Generators
==========

=>These are given by Sun Ms's JPA specification..
=> Will work in all ORM frameworks..
=>we can specify generator name directly in @GenerateValue (JPA)
=>4 generators are given
a) identity b) sequence c) table d) auto

a) identity
==========

=>uses the indentity columns of underlying Db s/w like mysql

eg::
#1) change to mysql Db s/w taking pk col as identity col
#2) cfg JPA Indentity Generator

eg1::

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Integer pid;
```

---

b) sequence
==========

=>Generates the id value either using default sequence (hiberante_sequence) or using the configured custom sequence.

#1) change to oracle Db s/w
#2) cfg JPA sequence generator

eg1::

```
@Id
@GeneratedValue(strategy = GenerationType.SEQUENCE)
private Integer pid;
```

uses the default sequence hibernate_sequence

output:: 1,2,3,4,...

---

eg2::

```
@Id
@SequenceGenerator(name ="gen1",
                   sequenceName = "JPA_PID_SEQ",
                   initialValue = 1000,  (like start with)
                   allocationSize =10 )  (like increment by)
@GeneratedValue(generator = "gen1",
                strategy = GenerationType.SEQUENCE)
private int pid;
```

Creates the custom sequence with given name having initial value as the start with value and allocationsize value as the increment by value

output:: 1000 , 1011 ,1021,....

---

eg3::

```
@Id
@SequenceGenerator(name ="gen1",
                   sequenceName = "JPA_PID_SEQ1")
@GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
private Integer pid;
```

default intial value:: 1
default allocationsize: 50

output ::1,52,102,152,...

---

eg4:: (if given custom sequence is already available by in DB s/w)

```
@Id
@SequenceGenerator(name ="gen1",
                   sequenceName = "PID_SEQ",
                   initialValue = 500,
                   allocationSize = 5)
@GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
private Integer pid;
```

initialValue: (will be ignored, will considerd only when HB creates sequence)
allocationSize: (must match with orginal increment by value of the sequence otherwise exception will be raised)

org.hibernate.MappingException: The increment size of the [PID_SEQ] sequence is set to [5] in the entity mapping while the associated database sequence increment size is [1].

Use Generators in hibernate programming having the following priority
a) JPA Generators
b) Hibernate Generators
c) Custom Generators

JPA Table Generator
==========
=> It is the combination of sequence generator and HILO Generators...
=> Works in all Db s/ws...
=> We need specify helper db table name and col names,col values... like HILO Generator

#eg1::

note: does not create or use sequneces but creates helper db tables..

```
@Id
@TableGenerator(name = "gen1",
                pkColumnName = "ID_COL",
                valueColumnName = "ID_VAL",
                pkColumnValue = "PID",
                table = "ID_TAB",
                initialValue = 10,
                allocationSize = 5 )
@GeneratedValue(generator ="gen1", strategy = GenerationType.TABLE)
private Integer pid;
```

creates Helper table like this

ID_TAB
=======

Generated values are :: 11,17,22,27,32 and etc..

ID_COL	ID_VAL
PID	~~10~~ (holds next id values)

```
@Id
@GeneratedValue( strategy = GenerationType.TABLE)
private Integer pid;
```

default table name:: hiberate_sequences
default pkcolumn name :: sequences_name
default pk column value :: default
default value column name :: next_hi
default initial value :: 1 default allocationsize:1

creates the helper db table like this

hibernate_sequences

sequence_name	next_hi
default	~~1~~ ~~2~~ 3 (hold next id value)

generated values are 1,2,3,.....

AUTO
=====

=>Picks one JPA generator based on the capabilities of underlying DB s/w.. In case of oracle it uses "SEQUENCE" but in case of "mysql" it uses "TABLE".

#for oracle
==========

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Integer pid;
```

-->uses the default sequence hibernate_sequence.

```
//# for oracle
@Id
@SequenceGenerator(name = "gen1",
                   sequenceName = "PROD_PID_SEQ",
                   initialValue = 5,
                   allocationSize = 10 )
@GeneratedValue(generator = "gen1",strategy = GenerationType.AUTO)
```

creates and uses PROD_PID_SEQ sequence

```
#for mysql
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Integer pid;
```

internally uses JPA Table Generator having default values...

note:: Generators are given to generate the values to ID property, pk column dynamically .. we generally take pk column surrogate key column , which is column values with out any business meaning.. So no use cases for generators

---

Composite Id Configuration
==========
=>if db table is having singular pk column then we should use singular Id using <id> or @Id
=>if db table is having composite pk column then we should use composite Id using <composite-id> or @EmbeddedId

note:: pk constraint applied one colmn --> singular pk
pk constraint applied multiple columns togather --> compsoite pk

Programmers_Projects_Work (db table)
==========

Composite pk on pid,ProjId cols togather

pid	pname	deptno	projId	projName
1001	raja	10	501	proj1
1001	raja	10	502	proj2
1002	ramesh	20	501	proj1
1002	ramesh	20	502	proj2

Here cols are having duplicates, So we can not apply singular pk on any column..

HEre we can pk constraint on pid,projId colums togather as composite pk.

note:: while working with composite Id/indentifier we can not apply any generators

```
CREATE TABLE "SYSTEM"."PROGRAMERSPROJECTSWORK"
(   "PID" NUMBER(10,0) NOT NULL ENABLE,
    "PNAME" VARCHAR2(20 BYTE),
    "DEPTNO" NUMBER(5,0),
    "PROJID" NUMBER NOT NULL ENABLE,
    "PROJNAME" VARCHAR2(20 BYTE),
     PRIMARY KEY ("PID", "PROJID"));
```

Xml driven cfgs based Composite id filed
==========

PrgmrProjId.java (helper class/Id class)

```
public class PrgmrProjId implements Serializable{
  private Integer pid;
  private Integer projId;
  //setters && getter
   .....
  //toString
   .....
}
```

ProgrammerProjectInfo.java (Entity class)

```
public calss ProgrammerProjectInfo implements Serializable{
  private PrgmrProjId id; // HAS-A Relation (composition)
  private String pname;
  private Integer deptNo;
  private String projName;
  //setters && getters
     .....
  //toString
     ...
}
```

In ProgramerProjectInfo.hbm.xml
==========

```
<hibernate-mapping>
  <class name="pkg.ProgrammerProjectInfo" table="PROGRAMMERSPROJECTSWORK">
    <composite-id name="id" class="pkg.PrgmrProjId">
      <key-property name="pid" column="PID" type="int"/>
      <key-property name="projId" column="PROJID" type="int"/>
    </composite-id>
    <property name="pname" column="PNAME" type="string">
     ....
     ...

  </class>
</hibernate-mapping>
```

Composite Id /Identifier cfg...

How many Properties can be there in composite id filed?
a) more than 1 property , we can have any no.of properites..

How many Properties can be there in composite id filed?
a) more than 1 property , we can have any no.of properites..

Can we cfg composite id and singular ID in the same Entity class?
ans) not possible..

Is pk col mandatory while designing db table for hiberante?
Ans) no .. but recomanded to take for designing of db table

Q) Is id filed /property configuration mandatory for Entity class in mapping file/domain class?
Ans) yes ... Though Db table is not having PK column , we must configure id field..

Save object operation using composite PK

==========

```
public static void main(String[] args) {
        Session ses=null;
        Transaction tx=null;
        ProgrammerProjectInfo info=null;
        PrgmrProjId id=null;
        boolean flag=false;
            //get Session
            ses=HibernateUtil.getSession();

            try {
                //begin Tx
                tx=ses.beginTransaction();
              //prepare Entity object
               id=new PrgmrProjId();
               id.setPid(101); id.setProjId(5001);
               info=new ProgrammerProjectInfo();
               info.setId(id); info.setPname("rajesh"); info.setProjName("OpenFx");
               info.setDeptNo(567);
               //save object
                id=(PrgmrProjId) ses.save(info);
                System.out.println("Generated id value::"+id);
                    flag=true;
        }//try
            catch(HibernateException he) {
                flag=false;
                he.printStackTrace();
            }
            catch(Exception e) {
                flag=false;
                e.printStackTrace();
            }
            finally {
                //Perform TxMgmt
                if(flag) {
                    tx.commit();
                    System.out.println("object is saved");
                }
                else {
                    tx.rollback();
                    System.out.println("Object is not saved");
                }
                //close objs
                HibernateUtil.closeSession(ses);
                HibernateUtil.closeSessionFactory();
            }//finally

    }//class
```

Load object operation

```
public static void main(String[] args) {
        Session ses=null;
        ProgrammerProjectInfo info=null;
        PrgmrProjId id=null;
            //get Session
            ses=HibernateUtil.getSession();

            try {
              //prepare Entity object
               id=new PrgmrProjId();
               id.setPid(101); id.setProjId(5001);
               //load object
               info=ses.get(ProgrammerProjectInfo.class, id);
               if(info!=null)
                   System.out.println(info);
               else
                   System.out.println("record not found");
        }//try
            catch(HibernateException he) {
                he.printStackTrace();
            }
            catch(Exception e) {
                e.printStackTrace();
            }
            finally {
                //close objs
                HibernateUtil.closeSession(ses);
                HibernateUtil.closeSessionFactory();
            }//finally

    }//class
```

Annotation driven Compsite id Field

EmbeddedId ---> To cfg Id class type property in entity class (HAS-A relation property> as composite Id field
Embeddable --> To make Id class as hepler class.. to Entiy class..
(Both are JPA Annotations)

```
@Embeddable
public class PrgmrProjId implements Serializable {
    private Integer pid;
    private Integer projId;
  //setters && getters
     .....
  //toString()
     ....
}
```

```
@Table(name="PROGRAMMER_PROJECT_INFO")
@Entity
public class ProgrammerProjectInfo implements Serializable{
  @EmbeddedId
  private PrmgProjId id;
   private String pname;
   private String deptNo;
   private String projName;
   //setters & getters
      ....
       ...
}
```

=>if we use lombok to generate setter,getter , toString methods.. we can not apply mapping annotations on the top of getter methods .. but we can apply on the top of fileds/properties/member variables.

There must be only one EmbeddedId annotation and no Id annotation when the EmbeddedId annotation is used.

note:: The property on which we are applying @EmbeddedId to make the property as composite identity property must be of class type having @Embeddeable..

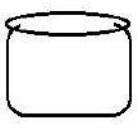
(Optional to have @Embeddable on the top of helper class..but recomanded to apply..)

Working with Date values
==========

=> while dealing with DOB,DOM,billDate, DOJ and etc.. we need to insert and retrive date values...
=>Hibernate provides abstration towards inserting date values... i.e we need to do multiple convertions..

In Plain Jdbc to insert date value
==========

Enduser --->String date value ---> SimpleDateFormat ---> java.util.Date class obj ----> java.sql.Date class obj-----> ps.setDate(-) --->
|--->parse(-)



In hibernate to insert date vlaue
==========

set java.util.Date obj to Entity Domain class obj --> ses.save(obj) ----->

Instead of this we can also use
java.sql.Date,
java.util.Calendar

example
=========

```
public class PersonInfo implements Serializable {
      private int pid;
      private String pname;
      private String paddrs;
      private  Date  dob;
      private Date  dom;
      private Date doj;
    //getters && setters
      ....
}
```

mapping file (PersonInfo.hbm.xml)

```
<hibernate-mapping>
      <class  name="com.nt.entity.PersonInfo"  table="PERSON_INFO">
        <id name="pid">
          <generator class="sequence"/>
        </id>
        <property name="pname"  length="20" type="string"/>
        <property name="paddrs"  length="20" type="string"/>
        <property name="dob"  type="timestamp"/>
        <property name="dom"  type="timestamp"/>
        <property name="doj"  type="timestamp"/>
      </class>
</hibernate-mapping>
```

type="date" --> ceeates the col of type "date" and inserts only "Date" values
type="timestamp" creates the col of type "timestamp" and insert both "date,time" values

What is the diff b/w java.sql.Date and java.util.Date?

java.util.Date obj holds both date and time values...
java.sql.Date obj holds only date value

java.util.Date (c)
↑ extends
java.sql.Date(c)

note:: when u take java.sql.Date class type properties in Entity class and u r setting values to Entity class obj through java.util.Date class obj indirectly then HB inserts both date and time values .. becoz it collects them from inernally avaiable java.util.Date class object...

Annotation driven Date values in hibernate
==========

```
@Entity
@Table(name="PERSON_INFO1")
public class PersonInfo implements Serializable {
      @Id
      @GeneratedValue(strategy = GenerationType.AUTO)
      private int pid;
      @Column(length=20)
      @Type(type="string")
      private String pname;
      @Column(length=20)
      @Type(type="string")
      private String paddrs;
      @Temporal(TemporalType.TIMESTAMP)  //default time stamp
      private  Date  dob;
      @Temporal(TemporalType.DATE)
      private Date  dom;
      @Temporal(TemporalType.TIME)
      private Date doj;
}
```

insertes both date and time

@Temporal specifies that info that should injected into db table col from given java.util.Date class obj data..

Oracle date types are date,timestamp.
mysql date types are date,time,timestamp, and etc..

---

we can use java8 java.time.LocalDateTime,LocalDate,LocalTime as the type of date properties in xml driven hibernate programming , but we should not place "type" attribute in the <property> tags of mapping file..

```
public class PersonInfo implements Serializable {
      private int pid;
      private String pname;
      private String paddrs;
      private  LocalDateTime  dob;
      private LocalDate  dom;
      private LocalTime doj;
      //setters  & getters
        .....
}
```

In Annotation driven hibernate programming we can java8 date,time api but there is no need to taking @Temporal annotation the top of properties..

```
@Entity
@Table(name="PERSON_INFO2")
public class PersonInfo implements Serializable {
      @Id
      @GeneratedValue(strategy = GenerationType.AUTO)
      private int pid;
      @Column(length=20)
      @Type(type="string")
      private String pname;
      @Column(length=20)
      @Type(type="string")
      private String paddrs;
      private  LocalDateTime  dob;
      private LocalDate  dom;
      private LocalTime doj;
      //getters  && setters
}
```

Versioning (or) Object Versioning

=> It keeps tracks of how many times object/record is loaded and modified by using hibernate..
=> It generated special column of type numeric based special number property of Entity class to keep track of the modifications..
=> This special property/col initial value is O and will be incremented by 1 for very modification..
=> To cfg this sepcialy property in mapping file use <version> annotation (after <id> tag) or use @Version as mapping annotation..

usecases:: keeping address modifications , caller tune modifications, profile details modification and etc..

Customer obj
id:101
name:raja
addrs: hyd
count: 0

→

Customer obj
id:101
name:raja
addrs: ~~hyd~~ delhi
count: ~~0~~ 1

Customer (db table)

id	name	addrs	count
101	raja	~~hyd~~ delhi	~~0~~ 1

note:: Versioning is also useful towards Locking process of the Object...

Exmaple (xml driven cfgs)

```
public class MobileCustomer  implements Serializable {
      private  Integer cno;
      private String cname;
      private long mobileNo;
      private  String callerTune;
      private  Integer versionCount;
      //getters  && setters
        ....
}
```

mapping file

```
<hibernate-mapping>
      <class  name="com.nt.entity.MobileCustomer"  table="MOBILECUSTOMER">
        <id name="cno">
          <generator class="sequence"/>
        </id>
        <version name="versionCount" type="int"  />
        <property name="cname"  length="20" type="string"/>
        <property name="mobileNo"  type="long"/>
        <property name="callerTune"  length="40"  type="string"  />
      </class>
</hibernate-mapping>
```

Annotation driven Object Versoninig

==========

```
@Entity
public class MobileCustomer  implements Serializable {
        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        private  Integer cno;
        @Column(length=20)
        @Type(type="string")
        private String cname;
        @Type(type="long")
        private long mobileNo;
        @Column(length=40)
        @Type(type="string")
        private  String callerTune;
        @Type(type="int")
        @Version
        private  Integer versionCount;
         //getters && setters
         ....
         ....
         }
```

note ::As of now Object versoning can be restricted one or more specific property values modification not

---

Object Time stamping

==========

=>Allows to keep track of object is saved (record inserted) and object is lastly updated...

usecase:: keeping track of when the Bank account is opened and lastly updated.

keeping track of when the vehicle,device is purchased and lastly serviced.

=> In xml driven cfgs .. we need to use <timestamp> to cfg special property that holds time stamp details... and that property type should be java.util.Date , java.util.Calendar, java.util.LocalDateTime and etc..

=>In Annotation driven cfgs .. we can use @CreationTimestamp and @UpdateTimestamp

note:: In xml driven cfgs we can no apply versioning and timestamp togather ... (DTD/Schema does not allow to use both tags a time) But possible in annotation driven env...

note:: In xml driven cfgs ... we can not take two seperate properties to hold creation time and lastupdation time.. (i.e single property holds both values overrding creation date time with last date time value every time) But annotation driven we can take two seperate properties for that...

Properties for timestamping can be of the follwing type

==========

- java.time.LocalDate (since Hibernate 5.2.3)
- java.time.LocalDateTime (since Hibernate 5.2.3) (Best)
- java.util.Date
- java.util.Calendar
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp

Xml driven env...

BankAccount

acno:1001

holder::raja

balance:: 9000

lastUpdated: ~~3-09-2020 9:41~~

~~3-09-2020 9:42~~

3-09-2020 10:10

9:42 10:10

~~10000~~ 60000

<timestamp>

BankAccount (db table)

==========

acno(pk)(n)	holder(vc2)	balance(float)	lastUpdated (timestamp)
1001	raja	~~9000~~	~~3-SEP-20 9:41~~
		~~10000~~	~~3-SEP-20 9:42~~
		60000	3-SEP-20 10:10

=>time stamp feature does not keep track of history of modifications done on the records.. like transaction history. for that we need keep db tables in one to many assocation.. using the concept of assocation mapping...

BankAccount(parent db table)

==========

acno(pk)	holder	balance
101	raja	90000

Bank_Transactions (child db table)

txId(pk)	date	type	acno
1234	10-JAN-20 9:55	withdraw	101
1134	10-AUG-20 9:55	deposite	101

Using timstamp feature of hibernate we can just keep track of when account created and/or lasly updated/operated.

```
public class BankAccount implements Serializable {
    private Long acno;
    private  String holderName;
    private  Double  balance;
    private String  type;
    private  LocalDateTime  openingDate;  // to just hold creation date and time
    //private  Timestamp lastUpdated;   //for timestamp
    private  Date lastUpdated;
   //setters && getters
     .....
     ....
}
```

mapping file

==========

```
<hibernate-mapping>
      <class  name="com.nt.entity.BankAccount" >
        <id name="acno" type="long">
          <generator class="sequence">
            <param name="sequence_name">ACNO_SEQ</param>
          </generator>
        </id>
        <timestamp name="lastUpdated"  />
        <property name="holderName"  length="20" type="string"/>
        <property name="type"  type="string" length="10"/>
        <property name="balance" type="double"/>
        <property name="openingDate"  />
      </class>
    </hibernate-mapping>
```

Annotation driven Object TimeStamping

=>we can use two hibernate annotations for this
@CreationTimestamp --> To hold object /record creation date and time
@UpdateTimestamp --> To hold when the object is lastly modified /updated.
=> In Annotation driven env.. we can apply both Versioning and Timestamp features togher on single Entity class (In xml driven cfgs not possble)

```
@Entity
@Table(name="HB_BANK_ACCOUNT")
public class BankAccount implements Serializable {
        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        @Type(type="long")
        private Long acno;

        @Column(length = 20)
        @Type(type="string")
        private String holderName;

        @Type(type="double")
        private Double balance;

        @Column(length = 10)
        @Type(type="string")
        private String type;

        @CreationTimestamp
        private LocalDateTime openingDate; // to just hold record creation date and time
        @UpdateTimestamp
        private LocalDateTime lastUpdated; // to hold the date and time indicating when the
record lastly updated
        @Version
        private Integer version;
    //setters & getters
      .....
      ......
}
```

note:: Xml driven cfgs do not allow java8 LocalDate, LocalDateTime,LocalTime type properties for Object time stamping but Annotation driven cfgs allow..

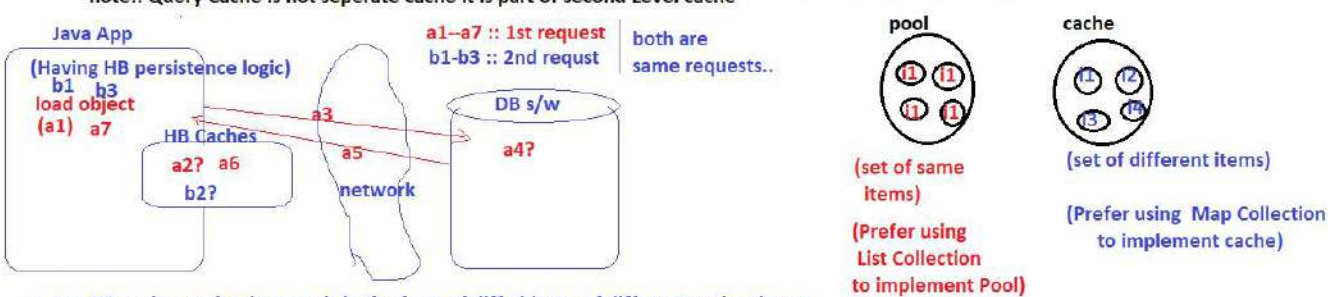
Caching in hibernate

Cache/Buffer

=>It is a temporarry memory that holds data for temporary period..
=> The cache at client side holds server supplied data and uses it across the multiple same requests.. to reduce the network round trips b/w client and server..
=> Hibernate Supports 2 levels of caching as Client side caching.. to hold the objects of Entity classes representing records and to use them across the multiple same requests...

1) First Level caching/ L1 cache / Level1 Cache /SessionCache (Built-in Cache - we can not disable)
2) Second Level caching/ L2 cache / Level2 Cache /SessionFactory Cache (It is configurable cache i.e we can eanble or disable)

note:: Query Cache is not seperate cache it is part of second Level cache



note:: HB caches maintain records in the form of diff objects of different entity classes

note:: To use cache effectively .. we need to empty the caches at regualar Internvals... So latest /new data will come and store in cache at regular internvals..

eg:: Stock trading , live game score ,weather report and etc...

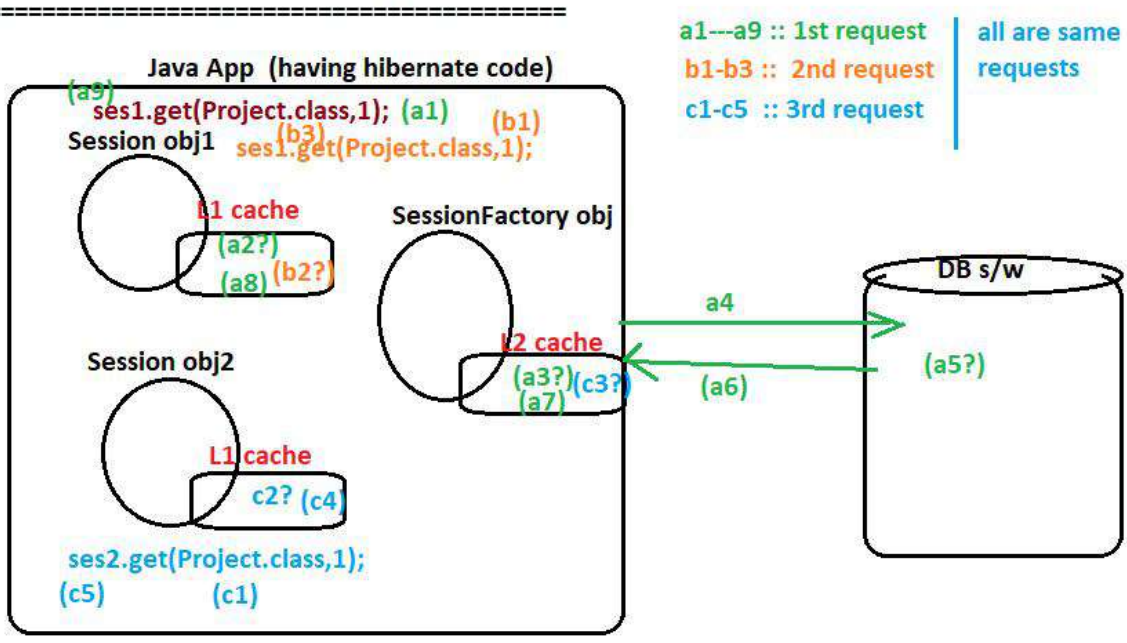
In hibernate First Level cache is in two ways::

(a) after calling ses.save(-) ,ses.persist(-) ,ses.update(-), ses.saveOrUpdate(-),ses.delete(-) methods keeps the given Entity objects in the first Level cache until we call tx.commit() method..

note:: The object is First Level cache keeps tracks of all modifications happening objs.. before calling tx.commit() and generates single update query representing all the modifications once the tx.commit() is called

(b) the Object loaded by ses.get(-,-)/ses.load(-,-) will be kept in First Level cache and will be used across the multiple ses.get(-,-)/load(-,-) method calls with same id/identity value.

Flow with respect two levels of caching in hibernate
==========



First Level Cache goals/ Advantages
==========

w.r.t ses.get(-,-)/ ses.load(-,-)

(a) if the loaded object is modified for multiple times the FirstLevel cache keeps track of all the changes and generates single update query instread of multiple update queries reflecting all changes when tx.commit() is called..

(b) if we try to load same object for multiple times only one time it will collect from DB .. keeps relavant entity obj is cache and uses it from multiple times to reduce
i)network roundtips with Db s/w
ii) to avoid extra objects creation for entity class (memory wastage is reduced)

(c) Session obj maintains only unique objects.. having one id value for 1 entity class only one object can be there in First Level cache

(d) when call ses.save(-), ses.persist(-), ses.delete(-),ses.saveOrUpdate(-) ,ses.update(-) methods first the object kept in First Level1 cache.. then its data will be used for Persistence opeations when tx.commit() is called..

L1 cache/First Level in hibernate is internally implemented as Map of Map

Map of Map obj Map<Class,Map<Object-id,Object-entity>>

Customer.class	map obj: 101 Customer obj; 102 Customer obj (key value)
Employee.class	map obj: 1001 Employee obj; 1002 Employee obj
Student.class	5001 Student obj
Project.class	10 Project obj (1g); 11 Project obj 3k

Customer c=ses.get(Customer.class,101);
Customer c1=ses.get(Customer.class,101);

```
public class FirstLevelCache{
 Map<Class,Map<Object,Object>> cacheMap=new HashMap();

                    (1e)  (3h)
  public  viod put(Object id , Object entity){
      Map<Object, Object> entityMap=null;

      Class clazz=entity.getClass();  (1f)
        if(cacheMap.containsKey(clazz)){  (3i)
            entityMap=cacheMap.get(clazz);  (3j)
                if(!entityMap.containKey(id)){
                        entityMap.put(id,entity);  3k
                    }
        }
        else{
            entityMap=new HashMap();
(1g)        entityMap.put(id,entity);
            cacheMap.put(clazz,entityMap);
        }
    }//method
                    (1b)  (2b)  (3b)
    public  Object  get(Class clazz, Object id){
            Map<Object ,Object> entityMap=null;
        if(!cacheMap.containsKey(clazz)){  (2c)  (3c)  (1c)
            // hit the Db s/w get id based Entity objec.
                ...
                put(id ,entity);  (1d)
            return entity;  (1h)
        }
        else{
                                (2d)  (3d)
            entityMap=cacheMap.get(clazz);
                if(!entityMap.containsKey(id)){  (2e)  (3e)
                    //hit the Db s/w and get id based Entity object  (3f)
                    put(id,entity)  (3g)
                    return entity;  (3l)
                }

                return  entityMap.get(id);  (2f)

        }
    }//get
```

```
(1i)        (1a)
Project p1=ses.get(Project.class,10);
Project p2=ses.get(Project.class,10);
(2g)        (2a)

Project p3=ses.get(Project.class,11)
3m          (3a)
```

Why Level1/First Level can not be disabled in hibernate?

Ans) All Persistence operations in hibernate takes place through Level1 cache.. i.e it is part of Hiberante Echo System. So we can not disable it.. For this reason only... they made First level/level1 cache as built-in cache in hibernate at Session object level.

To control Level1/First Level cache

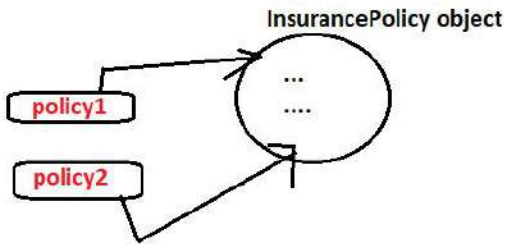
==========

To empty the entire cache

```
ses.clear();
```

To remove specific object cache

```
ses.evict(policy);
```



note:: when we call ses.close() method to the close the session all objects from first level/level1 cache will be emptied..

```
//Load object
policy=ses.get(InsurancePolicy.class,9001L); //gets from DB and puts in L1 cache
System.out.println("1"+policy);
System.out.println("..........................");
policy1=ses.get(InsurancePolicy.class,9001L);  //gets from L1 cache
System.out.println("2"+policy1);
System.out.println("...............................");
ses.evict(policy);  //removes from L1 cache
policy1=ses.get(InsurancePolicy.class,9001L);  //gets from DB  and puts in L1 cache
System.out.println("3"+policy1);
System.out.println("..........................");
policy1=ses.get(InsurancePolicy.class,9001L);  //gets from L1 cache
System.out.println("4"+ policy1);
ses.clear();  //removes  all object from L1 cache
policy1=ses.get(InsurancePolicy.class,9001L);  //gets from DB and puts in L1 cache
System.out.println("5"+policy1);
```

note:: Both ses.get(-,-) and ses.load(-,-) methods supports caching..

Why it is recomanded to implements Serializable(I) on Entity class?

Ans) if u r not working second level.. then there is no need of implementing it..
While working second level caches.. there is posibility enabling DiskCahing i.e keeping the objects of Entity class in file system through Serialization.. To support Serialization of disk caching .. it is recomanded to implement Serializable(I).

Entity Object life cycle diagram / 3 States of Entity object

==========

3 states

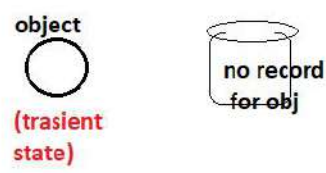
======

1. Transient state
2. Persistent state /Attached State
3. Detached state

Transient state

==========

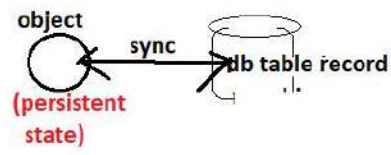
=> HEre the object of entity not bound/linked with Session object..
=> It will not have id value and does not hold /represent any Db table record data..



Persistent state

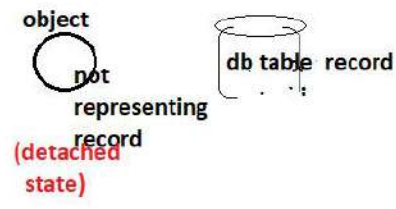
==========

=> HEre Object is linked with Session i.e it is placed in L1 cache of Session obj.
=>Contains id value and represents record in DB table having synchronization..
=> All persistence operations takes place by bringing object into this state.



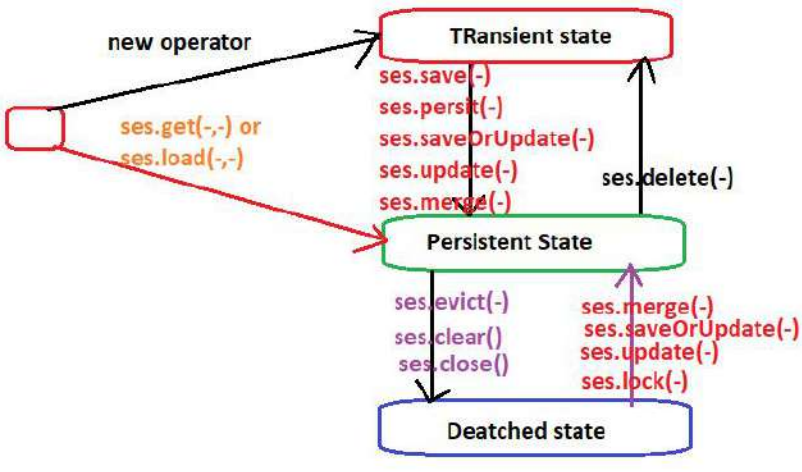
Datached State

->PReviously persistent ..but not now..
->contains id value.. but does not represent record in Db table and does not maintain synchronization also



if we delete the record in the middle of App execution when its object in Entity class is having pesistent state then what will be next tate of entity class?

ans) transient state becoz object becomes idle object with out record.



Connection Pooling and DataSources in hibernate

==========

=>SEssionFactory object holds jdbc con pool havign set of readymade jdbc con objects and uses them in the creation of HB session objects..
=> By default Hiberante App uses Hibernate's built-in JDBc Con pool which is not suitable for Production env.. becoz of performence issue..
=>To control Hiberante's built in jdbc con pool write the following entry in hibernate cfg file

```
<property name="connection.pool_size">25</property>
```

max pool size

What is the jdbc con pool that u have used in the Hibernate App?

Ans) *Do not use hibernate's bult-in jdbc con pool*

=>if ur hibernate App is standalone App then third patry standalone jdbc con pools like c3P0,proxool, hikari cp(best), vibur and etc..

if hiberante code based app is deployable in servers like hiberante is in web applications or webServices App then use underlying server managed jdbc con pool like weblogic , tomcat, widfly and etc.. servers managed jdbc con pool

note:: Hiberate f/w picks up the jdbc con pool based on the Connection Provider class that u configure..

note:: All connection Provider classes HB supplied pre-defined classes implementing org.hibernate.connection.ConnectionProvider

Hiberante is supporting the following standalone jdbc con pools

(a) c3PO (4)
(b) Proxool (3)
(c) HikariCP (1) (from 5.0)
(d) Vibur (2) (from 5.2)
(e) Agroal (from 5.4)
and etc...

con Pooltype	Provider class
c3P0	org.hibernate.c3p0.internal.C3P0ConnectionProvider
proxool	org.hibernate.proxool.internal.ProxoolConnectionProvider
hikari cp	org.hibernate.hikaricp.internal.HikariCPConnectionProvider
server managed jdbc con pool	org.hibernate.connection.DatasourceConnectionProvider
for built-in jdbc con pool	org.hibernate.connection.DriverManagerConnectionProvider

and etc...

Hibernate will internally determine which ConnectionProvider to use based on the following algorithm:
=>If hibernate.connection.provider_class is set, it takes precedence (best for standlaone)
=>else if hibernate.connection.datasource is set → Using DataSources (best for web application)
=>else if any setting prefixed by hibernate.c3p0. is set → Using c3p0
=>else if any setting prefixed by hibernate.proxool. is set → Using Proxool
=>else if any setting prefixed by hibernate.hikari. is set → Using HikariCP
=>else if any setting prefixed by hibernate.vibur. is set → Using Vibur DBCP
=>else if any setting prefixed by hibernate.agroal. is set → Using Agroal
=>else if hibernate.connection.url is set → Using Hibernate's built-in (and unsupported) pooling
=>else → User-provided Connections

Procedure to cfg Hikari cp in our Hibernate App?

step1) keep any standalone HB App ready

step2) Add hikari cp related hibernate jar file to the build.gradle.

step3) configure hikari cp related connection provider class in hibernate cfg file..

```
<property name="hibernate.connection.provider_class">
org.hibernate.hikaricp.internal.HikariCPConnectionProvider</property>
```

note:: https://docs.jboss.org/hibernate/orm/5.0/javadocs/ (use this url)

step4) Add hikiari cp properties in the hiberante cfg file..

```
<!-- Takes 20sec time before throwing error.. when pool can not give jdbc con object -->
<property name="hibernate.hikari.connectionTimeout">20000</property>
<!-- when pool is created it maintains 10 jdbc con objects -->
<property name="hibernate.hikari.minimumIdle">10</property>
<!-- when pool can grow max of 20 con objects -->
<property name="hibernate.hikari.maximumPoolSize">20</property>
<!-- if any con object in the pool is idle for30secs then it will be destroyed -->
<property name="hibernate.hikari.idleTimeout">30000</property>
```

step5) Run the Application.. observe the log files..

Procedure to configure Vibur jdbc con pool in our Hibernate App

step1) keep any App ready

step2) add hiberate-vibur jar file to build.gradle

```
// https://mvnrepository.com/artifact/org.hibernate/hibernate-vibur
implementation group: 'org.hibernate', name: 'hibernate-vibur', version: '5.4.0.Final'
```

step3) add Hibernate vibur cp connection provider class name in hibernate cfg file..

http://www.vibur.org/

```
<property name="hibernate.connection.provider_class">
org.hibernate.vibur.internal.ViburDBCPConnectionProvider</property>
```

step4) add hibernate.vibur.* properties in hiberante cfg file

from http://www.vibur.org/

```
<property name="hibernate.vibur.poolInitialSize">10</property>
<property name="hibernate.vibur.poolMaxSize">100</property>
<property name="hibernate.vibur.connectionIdleLimitInSeconds">30</property>
```

step5) Run the Client App..

ctrl +shift+ T :: to get source code of any searched class
f4 :: To know implementations .. of interfaces , interface method decl..

Why are not configuring DataSource class directly in hibernate.. why are configuring Connnection Provider class?

Ans) Hibernate f/w is designed to pickup DataSource class.. based on the Connection provider that we are configurating.. So they can allow only restricted DataSources and jdbc con pool to be associated with Hibernate for which hibernate.<datasource> properties are avaialble like hiberante.c3p0 properties , hibernate.hikaricp properties and etc...

As of now (5.4 version) Hibernate is giving 5 Connection provider classes for working 5 stanalone con pool s/ws.. they are c3p0, proxool , vibur, hikaricp , agroal and etc...

Procedure to cfg c3PO jdbc con pool in hibernate App

step1) keep hibernate app ready..

step2) add hiberante-c3p0 library/jar file to CLASSPATH or buildPath

```
// https://mvnrepository.com/artifact/org.hibernate/hibernate-c3p0
implementation group: 'org.hibernate', name: 'hibernate-c3p0', version: '5.4.18.Final'
```

step3) add hibernate.c3p0.___ properties to in hibernate cfg file..

```
<!-- c3Po properties -->
<property name="hibernate.c3p0.min_size">10</property>
<property name="hibernate.c3p0.max_size">30</property>
<property name="hibernate.c3p0.timeout">30</property>   In secs
<property name="hibernate.c3p0.max_statements">20</property>
```

At a time how many SQL queries

step4) Cfg Connection provider class in hibernate cfg file..

```
<property name="hibernate.connection.provider_class">og.hibernate.c3p0.internal.C3P0ConnectionProvider</property>
```

step5) run the Application..

note:: Though connection provider class is not configured... in hibernate cfg file.. it will use certain DataSource based on hibernate.<datasource>.____ properties added to hibernate cfg file or datasource related jar file that is added to hibernate App..

Procedure to cfg Agroal jdbc con pool/DataSource in our hibernate App

step1) keep hibernate App ready

step2) add hibernate-agroal jdbc con pool jar file /library to classpath/build path..

```
// https://mvnrepository.com/artifact/org.hibernate/hibernate-agroal
implementation group: 'org.hibernate', name: 'hibernate-agroal', version: '5.4.18.Final'
```

step3) add connection provider class.. in hibernate cfg file..

```
<property name="hibernate.connection.provider_class">org.hibernate.agroal.internal.AgroalConnectionProvider</property>
```

step4) add hibernate.agroal.___ properties in hibernate cfg file..

```
<!-- agroal properties -->
<property name="hibernate.agroal.minSize">10</property>
<property name="hibernate.agroal.maxSize">30</property>
```

collect from :: https://agroal.github.io/docs.html

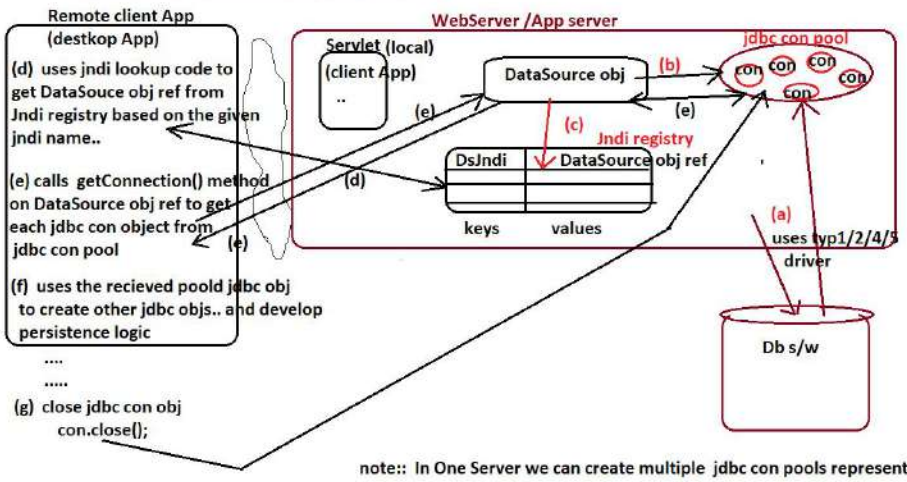
step5) run the Application...

=>WebServer/Application server is given to manage and execute the web applications

Application server= webServer ++

Working with server managed jdbc con pool

==========



=>To provide global visibility and accessibility to Java Objects .. their references will be kept in jndi registry having nicknames/aliasnames/ jndi names...

eg:: cos registry, rmi registry and etc..

note :: every Server gives one built-in jndi registry like weblogic registry, jboss registry , glassFish registry and etc...

=>Datasource obj represents jdbc con pool, each jdbc con object from jdbc con pool will be accessed through DataSoruce object..

note:: In One Server we can create multiple jdbc con pools represented by multiple datasource objs .For global visibility and accessbility we keep multiple datasource objs in Jndi registry having nicknames/jnid names.. as as shown above..

GlassFish

=========

type :: Appliation server java based

vendor :: Sun Ms /Oracle corp

version :: 5.x (compatible with jdk 1.8)

open source

default ports :: 8080 (http)

4848 (admin)

to download s/w :: as zip file from www.glassfish.org (https://download.oracle.com/glassfish/5.0/release/index.html)

To install server :: extract zip file

As of now hibernate is support the following 5 standalone jdbc con pools
(a)c3P0 (b) proxool (c) hikari (d) vibur (e) agroal

if want work with other than these standalone jdbc con pools s/w.. in hibernate App we need to develop Customer Connection Provider class and we need to cfg that class in hibernate cfg file...

Procedure to Custom Connection Provider to work with Apache DBCP2 jdbc con pool

step1) keep hibernate App ready

step2) add apache-dbcp2 jars/libraries to build.gradle.

```
// https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2
implementation group: 'org.apache.commons', name: 'commons-dbcp2', version: '2.7.0'
```

step3) Develope Custom Connection Provider class as shown below

```
public class ApacheDBCP2ConnectionProvider extends UserSuppliedConnectionProviderImpl {
    private BasicDataSource bds;

    public ApacheDBCP2ConnectionProvider() {
     System.out.println("ApacheDBCP2ConnectionProvider:: 0-param constructor");
            bds=new BasicDataSource();
            bds.setDriverClassName("oracle.jdbc.driver.OracleDriver");
            bds.setUrl("jdbc:oracle:thin:@localhost:1521:xe");
            bds.setUsername("system");
            bds.setPassword("manager");
            bds.setMinIdle(10);
            bds.setMaxIdle(100);
        }

    @Override
    public Connection getConnection() throws SQLException {
     System.out.println("ApacheDBCP2ConnectionProvider.getConnection()");
        return bds.getConnection();
    }

    @Override
    public void closeConnection(Connection conn) throws SQLException {
     System.out.println("ApacheDBCP2ConnectionProvider.closeConnection()");
      conn.close();
    }
}
```

step3) cfg the above Connection Provider class in hiberante cfg file...

```
<property name="hibernate.connection.provider_class">com.nt.provider.ApacheDBCP2ConnectionProvider</property>
```

note:: no need writing jdbc connection properies in in hiberante cfg file
(driver_class,url, username,password properites
are not required)

step4) run the Application...

Procedure to create domain in Glassfihs Server
==========

=>every Phyiscal App Server s/w installation will have multiple domains on 1 per project basis.

E:\glassfish-5.0\glassfish5\glassfish\bin>
asadmin create-domain --adminport=4444 --user=testuser GFNTHB914Domain
cmd option domain name
enter password :: testuser
enter password again:: testuser
. .
. .

changing http port number after domain creation in GlassFish
==========
Go to <GlassFihs_home>\domains\ GFNTHB914domain\config folder
--> open domain.xml file ---> modifity port attribute value of first <network-listener> tag
<network-listener port="7576" ...>

Procedure to create jdbc data soruce pointing to jdbc con pool for oracle in GlassFish SErver
==========

step1) make sure ojdbc6/7/8/9/10.jar file is added to GlassFish server..
copy ojdbc6/7/8/9/10.jar file to <Glassfish_hme>\domains\GFNTHB914Domain\lib\ext folder
domain folder

step2) start GlassFish domain server
E:\glassfish-5.0\glassfish5\glassfish\bin>asadmin start-domain GFNTHB914Domain

step3) Open Admin console screen
browser ---> http://localhost:4444 -->
submit username:: testuser
password::testuser

step4) create jdbc con pool for oracle ..
admin console screen ---> Resources ---> jdbc ---> jdbc con pool ---> new --->
name :: pool1 --> resorce type :: javax.sql.DataSource ---> vendor :oracle --->
next ---> set con pool properties ---> select and supply values to the following properties like

elect	Name	Value
	DataSourceName	OracleDataSource
	DriverType	thin
	ExplicitCachingEnabled	false
	ImplicitCachingEnabled	false
	LoginTimeout	0
	MaxStatements	0
	NetworkProtocol	tcp
	Password	manager
	PortNumber	1521
	ServerName	localhost
	ServiceName	xe
	URL	jdbc:oracle:thin:@localhost:1521:xe
	User	system

window key +shift +s --> for snipping..

--> next --->ping..

step5) create datasource pointing to jdbc con pool
admin screen ---> resoruces --->jdbc ---> jdbc resoruces (data source) --->
new ----> Jndi name:: DsJndi
jdbc con pool : pool1 ---> save..

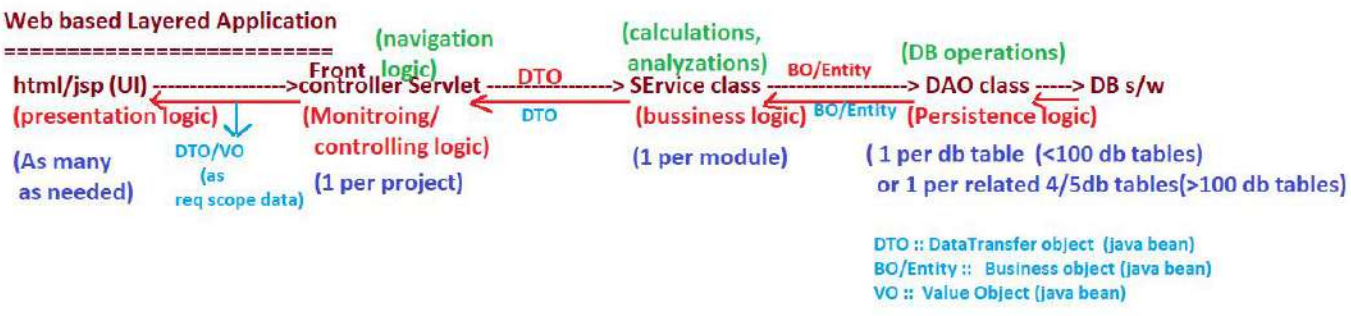
JNDI Name: DsJndi
Pool Name: pool1
Use the JDBC Connection Pools page to create new pools
Deployment Order: 100
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.
Description:
Status: Enabled

To use underlying server managed jdbc con pool in the hiberante code .. we need to write following entries in hibernat cfg file

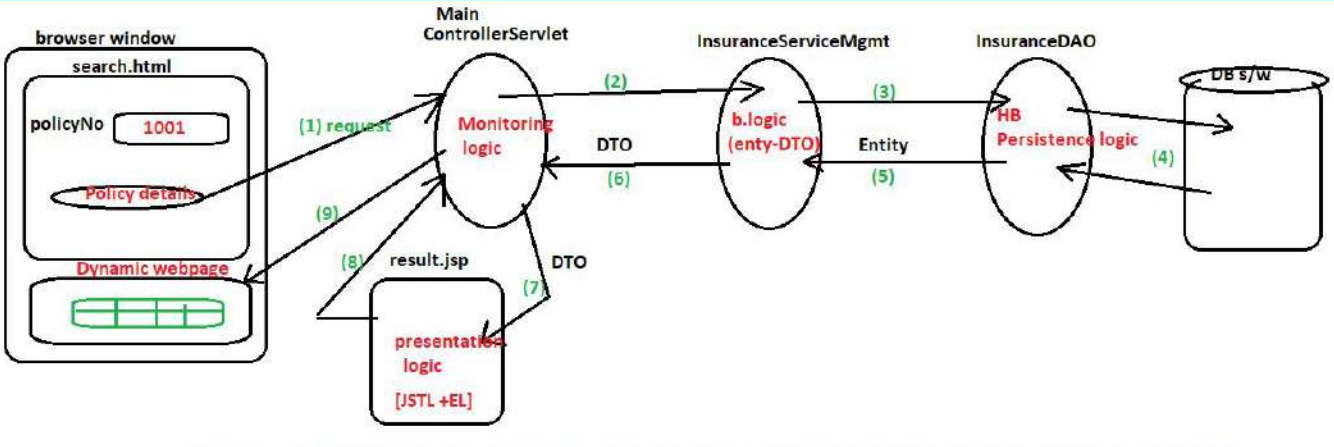
hiberante.cfg.xml
==========
<property name="hibernate.connection.provider_class"> org.hibernate.connection.DatasourceConnectionProvider </property>
<property name="hibernate.connection.datasource">DsJndi</property>

[Hibernate f/w uses the above info to perform jndi lookup(search0 operation on jndi registry of underying server to get DataSource object based on the given jndi name and that Datasource obj will be used hibernate f/w)

Web based Layered Application
==========



DTO :: DataTransfer object (java bean)
BO/Entity :: Business object (java bean)
VO :: Value Object (java bean)



Can not pass data to all layers by talikg BO /Entity /Model class (java bean) with out taking any Entity/DTO class?

Ans) we can do ... but not recomanded... the reasons
a) Placing dummy properties to hold business results effets Auto Schemal generation of ORM f/w
b) add new data on top of persistent data like adding serial number..
modifying retrieved data of DB table like round up , round down
activities for better presenttation becomes complex
c) Sending entity class with hb specific annotations to external comps like web service comps
(eg:: paytm,paypall) is going give problems.. If the extenal comps are not using hiberante libraries...

How many Session object we should take in an Application?

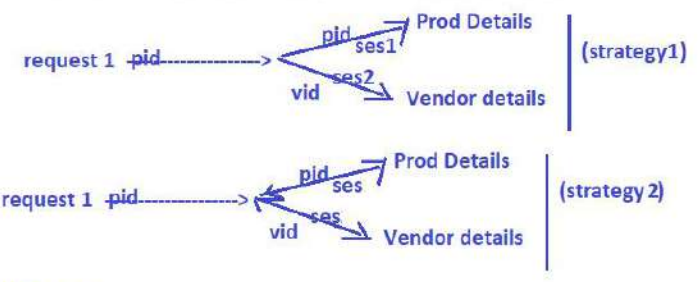
=>There are two strategies for this

(a) 1 session object per each persistence operation

(b) 1 session object per related persistence operations (Best)

1 session obj for 1 request is better ? (good)

or 1 session obj per each persistence opeartion is better ?



In DAO class

==========

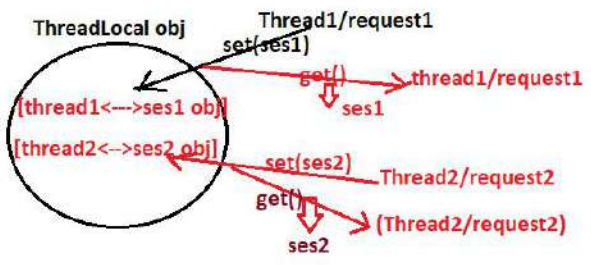
```
                    (b)
public  Product  getProduct(int pid);
  Session ses=HibernateUtil.getSession();  (c) ses1
  //get Product details
  Product p=ses.get(Product.class,pid);  (d) ses1
  //get Vendor details
   Vendor v=getVendor(p.getVid());  (e)
  ...  (j)
}
                    (f)
public  Vendor  getVendor(int vid){
  SEssion ses=HibernateUtil.getSession();  (g) (ses2)
   //get Vendor details
   Vendor v=ses.get(Vendor.class,vid);  (h) (ses2)
   return v;  (i)
  }
```

request1 (a) → Thread1

Here 1 request is performing two pesistence operations by using two different session objects..(Bad practice)

Solution is :: create HB session object on 1 per thread/request basis. and use that session object for multiple persistence operations performed by that thread or request by taking the support of ThreadLocal.

=>ThreadLocal allows each thread to keep one object at time..and allows only respective threads accissing /modifying/delting its object.. given from jdk1.4 ,but generics support is added from java6 ... This indirectly makes Object/data as thread safe..



ThreadLocal is internally hashMap collection having thread names as the keys and objects as the values..

HashMap object

thread1	ses1 object
thread2	ses2 obj2
keys	objects..

from request/thread1

==========

to put object in threadLocal

```
ThreadLocal<Session> threadLocal=new ThreadLocal();
threadLocal.set(ses1);
```

To get object form threadLocal

```
Session ses1=threadLocal.get();
```

To modify obj of threadLocal

```
threadLocal.set(ses10);
```

To remove object from threadLocal

```
threadLocal.remove();
```

Different scopes in Stand alone java app

==========

Local

instance

class

thread

different scope in web application

==========

page

request/ thread

session

application

Improvied HiberanteUtil.java to implenent 1 HB session object for related persistence operations strategy

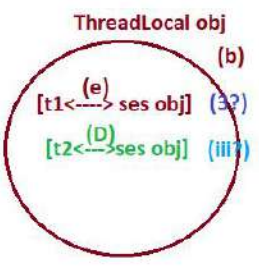
i.e in web applicaiton/standalone App 1 request/task related all persistence operations will take place by using same HB session obj

==========

HibernateUtil.java

==========

```
public class HibernateUtil{
  private  static  ThreadLocal<Session> threadLocal=new ThreadLocal();
   private static  SessionFactory factory;

   static{
      try{
        Configuration cfg=new Configuration();
        cfg.configure("/com/nt/cfgs/hibernate.cfg.xml");
(c)     factory=cfg.builSessionFactory();
       }
       catch(Exception e){
          e.printStackTrace();
        }
    } //static block
                                (d) (2) (B) (ii)
   public   static Session getSession(){
    Session ses=null;
    // get Session object from threadLocal
      ses=threadLocal.get();  (3?)  (C?)  (iii?)
       if(ses==null){
          if(factory!=null){
            ses=factory.openSession();  (e)
            threadLocal.set(ses);  (D)
          } //if
       } //if
       return ses;  (f)  (4)  (E)  (iv)
    } //m ethod

    public  static  void closeSession(){
    Session ses=null;
    //get Sessoion of the current
      ses=threadLocal.get();
    if(ses!=null){
      ses.close();
      threadLocal.remove();
    }
  }
  public static void closeSessionFactory(){
    if(factory!=null)
        factory.close();
  }
}
```



task1 /request1 (thread -->t1)

==========

```
(a)
(g) Session ses=HibernateUtil.getSession();
(5) Session ses=HibernateUtil.getSession();
                                  (1)
```

task2/request2 (thread--t2)

==========

```
(E) Session ses=HiberanteUtil.getSEsssion();  (A)
    Session ses=HiberanteUtil.getSession();  (i)
(v)
```

Using this HiberanateUtil class

In DAO class

==========

```
                    (b)
public  Product  getProduct(int pid);
  Session ses=HibernateUtil.getSession();  (c) ses1
  //get Product details
  Product p=ses.get(Product.class,pid);  (d) ses1
  //get Vendor details
   Vendor v=getVendor(p.getVid());  (e)
  ...  (j)
}
                    (f)
public  Vendor  getVendor(int vid){
  SEssion ses=HibernateUtil.getSession();  (g) (ses1)
   //get Vendor details
   Vendor v=ses.get(Vendor.class,vid);  (h) (ses1)
   return v;  (i)
  }
```

request1 (a) → Thread1

(Here we are using same session object for multiple persisntence operations of a same request /thread)

---

1 HB Session obj all persistences of a request strategy given the following advantages

==========

(a) Good performence

(b) Less mememory utlization

(c) better control on Transaction management

(d) if ThreadLocal is used... "ses" object becomes thread safe..

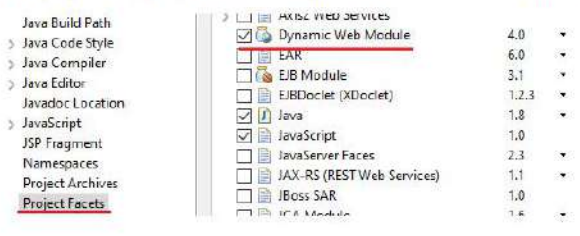
PRocedure to develop web application Using Gradle

==========

step1) create Gradle Project.

step2) convert Gradle Project into web application...

right click on project --->properties --->Project facets --> select dynamic web module 4.0 --> apply..



step3) chage plugin to war... and also following jar files in build.gradle..

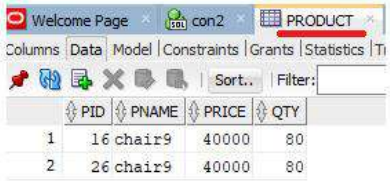
In build.gradle

==========

```
plugins {
  // Apply the java-library plugin to add support for Java Library
  id 'war'
}
repositories {
  jcenter()
}
dependencies {
   // https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api
  implementation group: 'javax.servlet', name: 'javax.servlet-api', version: '4.0.1'
  // https://mvnrepository.com/artifact/org.hibernate/hibernate-core
implementation  group: 'org.hibernate', name: 'hibernate-core', version: '5.4.21.Final'

 // https://mvnrepository.com/artifact/com.oracle.ojdbc/ojdbc8
implementation  group: 'com.oracle.ojdbc', name: 'ojdbc8', version: '19.3.0.0'

}
```

step4) create packages and develop the Application...

step4) Make sure Prodduct DB table is avaiable in Oracle DB s/w



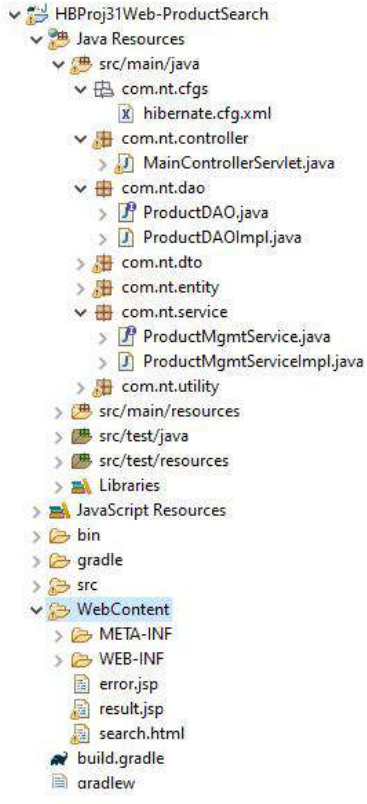
	PID	PNAME	PRICE	QTY
1	16	chair9	40000	80
2	26	chair9	40000	80

In hibernate.cfg.xml

```
<!-- Working server Managed jdbc con pool -->
 <property name="hibernate.connection.provider_class">
        org.hibernate.engine.jdbc.connections.internal.DatasourceConnectionProviderImpl
</property>
 <property name="hibernate.connection.datasource">DsJndi</property>
```

Jndi name of DataSource in underlying WebServer or Application server

step5) develop the soruce code the application ..

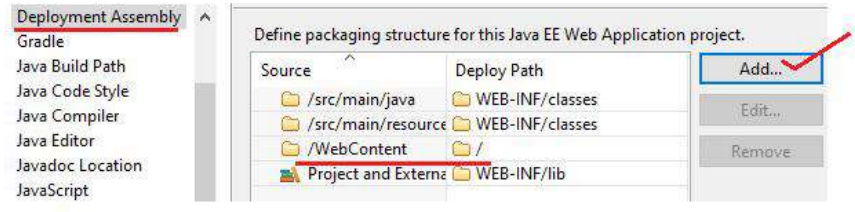


=>All Exceptions in hibernate unchecked exceptions... so they will be propagated automatically from one layer to another layer .. so no need using "throws","throw" clauses explicitly...

=> While developing Layered Apps ... it recomanded to take interface, impl class model in every layer.. to achieve loose coupling or runtime binding or dynamic polymorphism

note:: By default the webcontent folder does not participate in deployment.. whe should add it explicitly... by using deployment assembly option..

right on project ---> properties ----> deployment assembly ---> add ----> folder webcontent ---> apply --ok..



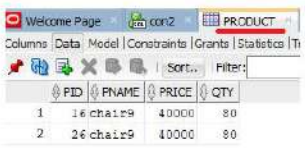
step6) make sure that Glassfish domain Server is cfg with eclipse IDE

a) install oracle weblogigc tools as server adpaters

window -->preferences ---> server --->Runtime env.. ---> add -----> oracle --->oracle weblogic tools ---> next ---> accept terms and conditions --> ..... restart IDE --> ....

b) cfg GlassFish server with Eclipse IDE.

step4) Make sure Prodduct DB table is avaiable in Oracle DB s/w

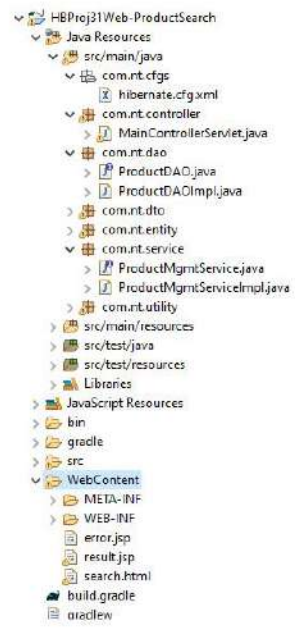


In hibernate.cfg.xml

```
<!-- Working server Managed jdbc con pool -->
 <property name="hibernate.connection.provider_class">
        org.hibernate.engine.jdbc.connections.internal.DatasourceConnectionProviderImpl
</property>
 <property name="hibernate.connection.datasource">DsJndi</property>
```

Jndi name of DataSource in underlying WebServer or Application server

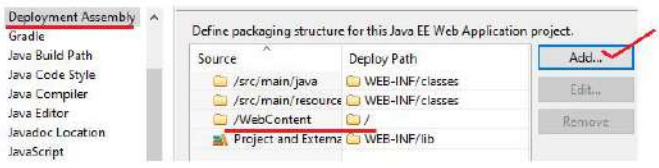
step5) develop the soruce code the application ..



=>All Exceptions in hibernate unchecked exceptions... so they will be propagated automatically from one layer to another layer .. so no need using "throws","throw" clauses explicitly...

=> While developing Layered Apps ... It recomanded to take Interface, impl class model in every layer.. to achieve loose coupling or runtime binding or dynamic polymorphism

note:: By default the webcontent folder does not participate in deployment..
whe should add it explicitly... by using deployment assembly option..
right on project ---> properties ----> deployment assembly ---> add ----> folder webcontent ---> apply --ok..



step6) make sure that Glassfish domain Server is cfg with eclipse IDE

a) install oracle weblogigc tools as server adpaters
window -->preferences ---> server --->Runtime env.. ---> add -----> oracle --->oracle
weblogic tools ---> next ---> accept terms and conditions --> ..... restart IDE --> ....

b) cfg GlassFish server with Eclipse IDE.
window -->preferences ---> server --->Runtime env.. ---> add --> Glassfish --->GlassFish ---> create LocalServer --->
-->GlassFish Home :: E:\glassfish-5.0\glassfish5\glassfish
-->next ---> select GFNTHB914Domain ---> username:: testuser , password:: testuser ---> .....

step7) run the application...
Rigth click on the project ----> run as ---> Run on server -->select GlassFish server ---> .....

Procedure to create Jdbc Con pool for oracle in the tomcat server cfg with Eclipse IDE
==========

step1) make sure that tomcat server is cfg with Eclipse IDE

window -->preferences ---> server --->Runtime env.. ---> add --> apache ---> Tomcat 9.0 ---> Tomcat home :: E:\Tomcat 9.x

step2) Add the following <Resource> tag in <Context> tag of context.xml file of ProjectExplore --->servers --->Tomcat section

```
<Resource name="DsJndi" auth="Container"
          type="javax.sql.DataSource" driverClassName="oracle.jdbc.OracleDriver"
          url="jdbc:oracle:thin:@localhost:1521:xe"
          username="system" password="manager" maxTotal="20" maxIdle="10"
          maxWaitMillis="20000"/>
```

collect from <Tomcat_home>/webapps/docs/jndi-datasource-examples-howto.html

---

=>While specifying DataSource Jndi name of Tomcat server in hibernate cfg file we need to add some extra prefixes as shown below

```
<property name="hibernate.connection.datasource">java:/comp/env/DsJndi</property>
<!-- only for Tomcat place extra prefix(java:/comp/env/) -->
```

=>We should avoid or minimize java code in jsp page to make our jsp page more readable and more friendly for UI Designers.. For this we can use JSTL +EL concepts
=> JSTL is having 5 jsp taglibaries .. and every jsp tag library contains bunch of tags and identified with taglib uri
=> To use jsp library tags in jsp page we need to import taglib url and also specifying user-defined prefix.

To use JSTL tags in our jsp pages
==========

a) add JSTL Libraries as dependents to projects..

```
// https://mvnrepository.com/artifact/javax.servlet/jstl
implementation group: 'javax.servlet', name: 'jstl', version: '1.2'
// https://mvnrepository.com/artifact/taglibs/standard
implementation group: 'taglibs', name: 'standard', version: '1.1.2'
```

b) import jstl jsp taglib uri and use its tags..

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@page isELIgnored="false" %>

<h1 style="color:red;text-align:center">Result page</h1>
 <c:choose>
   <c:when test="${!empty pDTO || pDTO ne null } ">
    <table border="1" align="center">
        <tr>
         <td>${pDTO.pid} </td>
         <td>${pDTO.pname} </td>
         <td>${pDTO.price} </td>
         <td>${pDTO.qty} </td>
         <td>${pDTO.category} </td>
        </tr>
      </table>
   </c:when>
    <c:otherwise>
       <h1 style="color:red;text-align:center"> No Product found </h1>
    </c:otherwise>
 </c:choose>

 <br><br>
  <a href="search.html">home</a>
```

DAO class
=========

```
public  Product getProduct(int pid){

  Session ses=HibernateUtil.getSession();   ses1
   ... use ses
      //Load product..
    getVendor(vid);
    getLocation(lid)
 HiberanteUtil.closeSession();
}
```

```
public   Vendor getVendor(int vid){

  Session ses=HiberanteUtil.getSession();  ses1
  //load   Vendor
     ....
    HibernateUtil.closeSession();
}
```

```
public   Vendor getLocation(int lid){

  Session ses=HiberanteUtil.getSession();    ses 1
  //load   Location
     ....
    HibernateUtil.closeSession();
}
```

=>In ThreadLocal managed SEssion object based web application.. we do not close Session obj at the end of each persistence method.. becoz 1 request may call multiple persistence methods.. and we want to use same session object in all persistence methods calls of a request.. So it is better to close session object .. in requestDestroy(-) method ServletRequestListener which executes .. for request-response related request object destruction..

```
@WebListener
public class SessionClosingServletRequestListener implements ServletRequestListener {
      @Override
      public void requestDestroyed(ServletRequestEvent sre) {
            System.out.println("SessionClosingServletRequestListener.requestDestroyed(-)");
        HibernateUtil.closeSession();
      }

}
```

In web applications, we need close SessionFactory object at the end web application execution. like when server is stopped or web application is stopped or reloaded .For all these activities Servletcontext will be destroyed.. So by using contextDestroyed(-) method of ServletContextListener(I) , we can close SessionFactory obj.

```
@WebListener
public class SessionFactoryClosingContextListener implements ServletContextListener {
      @Override
      public void contextDestroyed(ServletContextEvent sce) {
            System.out.println("SessionFactoryClosingContextListener.contextDestroyed()");
        HibernateUtil.closeSessionFactory();
      }

}
```

Contextual Session (factory.getCurrentSession() method)
==========

This makes SessionFactory object to create HB session object based on context or condtion we specify in hibernate cfg file for "hibernate.current_session_context_class" property...

in hiberante.cfg.xml
==========

```
<property name="hibernate.current_session_context_class">thread </property>
                                                          jta
                                                          managed
```

org.hibernate.context.JTASessionContext (jta):
current sessions are tracked and scoped by a JTA transaction. The processing here is exactly the same as in the older JTA-only approach. See the Javadocs for details. (useful while working spring Tx or server mnaged Tx)

org.hibernate.context.ThreadLocalSessionContext (thread): (suitable for web applications)
current sessions are tracked by thread of execution. here Session object will be close automatically at the end of thread

org.hibernate.context.ManagedSessionContext (managed):
current sessions are tracked by thread of execution. However, you are responsible to bind and unbind a Session instance with static methods on this class: it does not open, flush, or close a Session.

=>All these instructions will be considered only when we call factory.getCurrentSession() method to get the Sesssion object

Example
=========

step1) keep any web application ready that is using hiberante...

step2) Enable Contextual Session in hiberatne cfg file having "thread" as the value..

```
<!-- Contextual Session -->
        <property name="hibernate.current_session_context_class">thread</property>
```

step3) improve HibernateUtil.java by calling factory.getCurrentSession() method

```
public static  Session getSession() {
        Session ses=null;
        //get Session object from ThreadLocal
    if(ses==null) {
        if(factory!=null) {
            ses=factory.getCurrentSession();
        }//if
    }//if
        return ses;
}
```

note:: here no need of ServletListener to close Session obj... and no need of placing closeSession() method in HibernateUtil.java

step4) make sure that Tranaction tx= ses.beginTransaction() method is called in persistence methods of DAO class... though the persistenece methods are performing select operations..

In DAO class

```
@Override
    public Product getProduct(int pid) {
        Session ses=null,ses1=null;
        Product prod=null;
        Transaction tx=null;
      //get SEssion object
        ses=HibernateUtil.getSession();
        ses1=HibernateUtil.getSession();
        // dummy Begin Tx
         tx=ses.beginTransaction();
        System.out.println(ses.hashCode()+"  "+ses1.hashCode());
        //get/load object
        prod=ses.get(Product.class, pid);
        return prod;
    }
```

---

if we factory.getCurrentSession() method without cfg call "contextual SEssion" property in hibernate cfg file (*'hibernate.current_session_context_class*) then we get org.hibernate.HibernateException: No CurrentSessionContext configured!

What is the different b/w factory.openSession() and factory.getCurrentSession() methods?

factory.openSession()	factory.getCurrentSession()
(a) creates and returns new HB session obj always	(a) creates/locates and returns Hiberante Session obj based on context/contrract that is specified in hibernate cfg file.. using *'hibernate.current_session_context_class* property
(b) useful to create Session object on 1 per each persistence operation basis	(b) useful to create Session object on 1 per thread /request basis i.e all related Persistence opeations of a request will same session object
(c) no need of cfg any properties in hb cfg file..	(c) *'hibernate.current_session_context_class* property must be cfg having values thread/jta/managed value..
(d) No need of taking Tx for Select operations	(d) One dummy Tx required in even for Select operations..
(e) Not so industry standard	(e) Vert much industry standard...

Interacting with more than 1 DB s/w using Hibernate

==========

Usecase :: => copying one Bank customer details to another Bank .. when merger is happend
=> transfer money operation b/w two accounts of two diff banks..

Union bank = union bank + andhraBank + Corporation Bank
SBI = SBI+ SBH+SBP+SBT+SBB+SBM
VI company = voda fone + idea.
and etc..

Example App

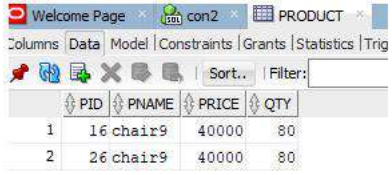
==========

Copy Product details based on given pid from Oracle Product Db table to mysql Product Db table

Oracle Product db table
==========
|---->pid (n) (pk)
|--->pname (vc2)
|--->price (float)
|--->qty (float)
(with records)

MySQL Product db table
==========
|---->pid (int) (pk)
|--->pname (vc)
|--->price (float)
|--->qty (float)
(no records)

=>Both db table names, col names can be different ..but cols count and type must match.. (compatiable)



2 hb confiuration files
==========
oracle-hibernate.cfg.xml
mysql-hibernate.cfg.xml

1 Entity class
==========
Product.java

1 Mapping file (if both DB table names and col names are matching)
==========
Product.hbm.xml

2 Mapping files (if both DB table names and col names are not matching)
==========
oracle-product.hbm.xml
mysql-product.hbm.xml

optional if we use mapping annoations...

=>natrual Ids
=>BootStrapping of Hibernate (1.legacy 2. ServiceRegistry)
=>working with lobs (files)
=>Hibernate tool in Eclipse for dyamic generation ..

DataTransferDAO.java
DataTransferDAOImpl.java

DAO pattern

2 HibernateUtil classes
==========
OracleHiberanteUtil.java
MysqlHibernateUtil.java

InteractingWithMultipleDBsTest.java (ClientApp)

Can we develope HB App with out HB cfg file (xml) and HB mapping (xml) file ?

Ans) yes possible... Avoiding HB cfg file is not recomanded...

note:: we can replace hb cfg file (xml) with properties file or programatic approach (java code) (not recomanded) | (In Boot strapping of hibernate)

note:: we can replace hb mapping file(xml) with mapping annotations added to Entity class

note: In Spring with HB (either using Spring ORM or spring data) there is no need of both xml files)

By using single SessionFactory obj can we able to talk with Db s/w?
Ans) not possible

BootStrapping in Hibernate
==========
=>It is all about keeping 3 objects (Configuration,SessionFactory,Session) of Hibernate ready.. to use them for PErsistence operations..

Two approaches of Bootstrapping
==========
--> To create the above 3 objects .. we need instructions in the form hibernate cfg properties and they can be given in different ways..

a) Legacy Approach /Old Apporach
|---> Using hibernate cfg file (xml) --->Declarative Approach (best)
|---> Using hibernate.properties file
|---> Using Programatic Approach

b) Modren Approach (from Hibernate 3.6)
|--->Using SErvice Registries

Working with hibernate.properties file

=> Every Hiberante App first attempts to load .. hibernate.properties file .. before loading hibernate.cfg.xml file.. to collect hibernate cfg properties.. (from classpath (by default "src/main/java" folder))

=> In this properties file.. keys are fixed ... values will be changed .. according the DB s/w ,jdbc driver s/w and HB settings we use..

Example App
==========
refer :: HBProj34-BootStrapping-Legacy Project

step1) keep any old hibernate app ready..

step2) delete hiberante cfg file..

step3) develop hibernate.properties file as shown below.. (add to src/main/java folder)

hibernate.properties
==========
```
# conenction properties
hibernate.connection.driver_class=oracle.jdbc.driver.OracleDriver
hibernate.connection.url=jdbc:oracle:thin:@localhost:1521:xe
hibernate.connection.username=system
hibernate.connection.password=manager

#dialect
hibernate.dialect=org.hibernate.dialect.Oracle10gDialect
hibernate.show_sql=true
hibernate.format_sql=true
hibernate.hbm2ddl.auto=update
```

the hiberate.propeties file of src/main/java folder (classpath folder) will be detected automatically during activation HB f/w.. if file name or its location is changed we need to specify name,location explicitly in Client App..

step4) add following code in HiberanteUtil.java static folder to create SessionFactory object..

```
static {
    Configuration cfg=null;
    try {
        //boot strap hibernate
        cfg=new Configuration();
        //speficyt the name of the mapping file
        cfg.addFile("src/main/java/com/nt/entity/InsurancePolicy.hbm.xml");
        //build SessionFactory
        factory=cfg.buildSessionFactory();
    }//tru
    catch(HibernateException he) {
        he.printStackTrace();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}//static
```

(or)
cfg.addAnnotatedClass(InsurancePolicy.class)
For Annotation driven Entity class.

=>There are no annotations for Hibernate cfg properites.. So we should always supply them as xml file or properties file or programatic approach.. we have only mapping annotations alternate mapping file (xml)

step5) Run the Client App ...

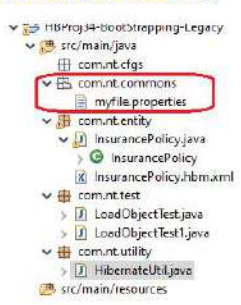
Limitations with hiberrnate.properties
==========
(a) properties file name and location is fixed
(b) we can not specify mappipg file names.. in hibernate.properties...
(sperate java statements should be in HibernateUtil.java)
(c) Except for "connection properties" .. for remaining all properties we must add "hibernate" word in the property name
(d) It is not industry standard..
(e) No XSD/DTD rules to verify content..

Advantages
==========
=>Gives loose coupling towards changing DB s/w and hibernate cfg properties..

How to work with Properties file by taking our choice name and location for that file?

Ans) We need to write the following code in the static block of HiberanteUtil.java as shown below..

```
static {
    Configuration cfg=null;
    InputStream is=null;
    Properties props=null;
    try {
        //load and read Properties file
        is=new FileInputStream("src/main/java/com/nt/commons/myfile.properties");
        // put properties file content to java.util.Properties object
        props=new Properties();
        props.load(is);
        //boot strap hibernate
        cfg=new Configuration();
        cfg.setProperties(props);
        //speficyt the name of the mapping file
        cfg.addFile("src/main/java/com/nt/entity/InsurancePolicy.hbm.xml");
        //cfg.addAnnotatedClass(InsurancePolicy.class)
        //build SessionFactory
        factory=cfg.buildSessionFactory();
    }//tru
    catch(HibernateException he) {
        he.printStackTrace();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}//static
```



Programatic Approach (Java code)
==========
a) keep old App ready by removing propertes file and cfg file (xml)

b) add the following code in the staic block of HibernateUtil.java

```
static {
    Configuration cfg=null;

    try {
        //boot strap hibernate
        cfg=new Configuration();
        //set hibernate cfg properties explicitly
        cfg.setProperty("hibernate.connection.driver_class", "oracle.jdbc.driver.OracleDriver");
        cfg.setProperty("hibernate.connection.url", "jdbc:oracle:thin:@localhost:1521:xe");
        cfg.setProperty("hibernate.connection.username", "system");
        cfg.setProperty("hibernate.connection.password", "manager");

        cfg.setProperty("hibernate.connection.dialect", "org.hibernate.dialect.Oracle10gDialect");
        cfg.setProperty("hibernate.show_sql", "true");

        //speficyt the name of the mapping file
        cfg.addFile("src/main/java/com/nt/entity/InsurancePolicy.hbm.xml");
        //cfg.addAnnotatedClass(InsurancePolicy.class)
        //build SessionFactory
        factory=cfg.buildSessionFactory();
    }//tru
    catch(HibernateException he) {
        he.printStackTrace();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}//static
```

Limitations
==========
(a) we can not specify mappipg file names.. in hibernate.properties...
(sperate java statements should be in HibernateUtil.java)
(b) In all properties the Word "hibernate" is mandatory..
(c) Since we are hard coding in .java file.. we will loose the flexibility of modification..
(d) Not industry standard..

Can we develop hibernate app by eleminating both Cfg file(xml) and mapping file(xml)?
ans) Yes , we can
==>Replace HB cfg file(xml) with properties file or programatic code (Not at all recomanded)
===> Replace HB mapping ile (xml) with mapping annotations.. placed in Entity class.. (Recomanded)

if we place all the 3 approaches (hibernate.properites file, programatic , declarative) in our HB App then what will be applied?

=> if cfg.configure(-) is called after programatic approach code.. then xml file cfgs takes place otherwisse progrmatic code (java code cfg.setProperty(-,-)) settings will take place..

Boot strapping in Hiberante
==========

a) Legacy Approach
   i) Using xml file (declarative)
   ii) using hibernate.properties file
   iii) Using java code (programatic)
b) Modren Approach (From Hibernate 3.6) (Using service Registries)

Modern Boot Strapping
==========

Service
=======

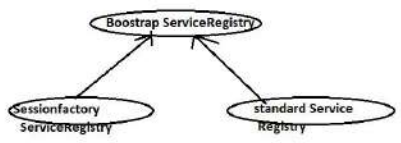
=>It is pluggable or addtional logic to provide extra functionalaties on the top of existing s/w

eg:: StrategyService , ConnectionProvder Service, DialectResolverService and etc..

note:: All Hibernate services directly or indirectly implements org.hibernate.service.Service(I) (marker interface)

ServiceRegistry
==========

=> It is a registry that holds services, manages the services and provides access to the services..
=> All Service registries are the different classes implementing org.hibernate.service.ServiceRegistry(I)
=> ServiceRegistries are hierarchal i.e child service registry can use the services of parent service registry..



BootStrapServiceRegistry contains the following services:-
(a)ClassLoaderService
(b)IntegraterService
(c)StrategyService and etc..
→StandardServiceRegistry contains the following standard services:-
(a)JdbcService
(b)ConnectionProviderService
(c)DialectResolver
(d)BatchService and etc..

ServiceRegistryBuilder
==========

=> It is a builder that is required to create ServiceRegistries..
=> Using this builder we can also add services to SErviceRegstry during the creation process.
=> All ServiceRegistryBuilders are the classes implementing org.hibernate.testing.ServiceRegistryBuilder(I)

eg:: BootServiceRegistryBuilder , StandardSErviceRegistryBuilder , SessionFactoryServiceRegistryBuilder..

ServiceRegistryBuilder ----------creates--------->SErviceRegsitry ----------> holds and manages services

BootStrapping code (Legcy code) (In All versions)
==========
(deprecrated in 4.x)

```
Configuration cfg=new Configuration();
cfg.configure("com/nt/cfgs/hibernate.cfg.xml");
SessionFactory factory=cfg.buildSessionFactory();
Session ses=factory.openSession();
        (or)
SEssion ses=factory.getCurrentSession();
```

(There is no provision to add new services/ modified services)

Bootstapping code of Modren Approach (In Hiberante 5.x)
==========

```
Configuration cfg=new Configuration();
cfg.configure("com/nt/cfgs/hibernate.cfg.xml");
  cfg.addResource("com/nt/entity/InsurancePolicy.hbm.xml");
//cfg.addAnnotatedClass(InsurancePolicy.class) //for annoated classes
//create SErviceRegistry Builder
ServiceRegistryBuilder builder=new StandardServiceRegistryBuilder();
//create ServiceRegstiry having basic services of hibernate.cfg.xml
ServiceRegistry registry=builder.applySettings(cfg.getProperties()).build();
//create SEssionFactory obj
SessionFactory factory=cfg.buildSessionFactory(registry);
//create Session object
Session ses=factory.openSession()/getCurrentSession();
```

(Here we can added custom services, modified services to the service registries)

=>cfg.buildSessionFactory() is deprecated in HB 4.x and brought back 5.x (legacy approach)
=>cfg.buildSessionFactory(registry) is instroced in HB 3.6 and contined in HB 4.x and HB 5.x
⇧
(Modern Approach)

Boot stapping code in Hbierante 4.x
==========

```
Configuration cfg=new Configuration();
cfg.configure("com/nt/cfg/hibernate.cfg.xml");
//build ServiceRegistry
ServiceRegistryBuilder builder=new ServiceRegistryBuilder();
//create ServiceRegistry
ServiceRegstry registry=builder.applySettings(cfg.getProperties())
                                                .buildServiceRegistry();
    //build SessionFactory
SEssionFactory factory=cfg.buildSessionFactory(registry);
Session ses=factory.openSession()/getCurrentSession();
```

Procedure to Custom Service to SErvice regsitry
==========

step1) keep Hiberatne 5.x boostrapping app ready

step2) Develop Custom service class...as ConnectionProvider class (having Programmer created jdbc con obj)

```
public class UserConnectionService implements ConnectionProvider {

    @Override
    public boolean isUnwrappableAs(Class unwrapType) {return false;}
    @Override
    public <T> T unwrap(Class<T> unwrapType) {      return null;    }

    Connection con;
    @Override
    public Connection getConnection() throws SQLException {
        System.out.println("UserConnectionService.getConnection()");
      con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "manager");
        return con;
    }
    @Override
    public void closeConnection(Connection conn) throws SQLException {
     con.close();
    }

    @Override
    public boolean supportsAggressiveRelease() {
        // TODO Auto-generated method stub
        return false;
    }

}
```

step3) add service to ServiceRegsitryBuilder to place in service registry..

```
//add service                 In HiberatneUtil.java
builder.addService(ConnectionProvider.class, new UserConnectionService());
                   service type              Custom service object..
```

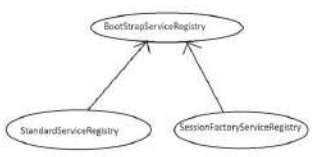
step4) run the the application...

Using Legacy Approach... we can devleop few custom services.. but not all types... and there is no provision to create new SErvice Type.. where as in Modern Approach.. these problems can be solved..

Reference

What is Service?
Service is a pluggable unit to existing software providing additional functionality to existing s/w.
In Hibernate Services these are given to use external resources and built-in resources in Hibernate.
e.g. ClassLoaderService, Strategy Service, Connection Provider Service and etc..
→All service classes of Hibernate implements org.hibernate.service.Service(I) (Marker Interface)..

Service Registries:-
The central service API, aside from the service themselves is the org.hibernate.service. Service Registry Interface. The main purpose of a service registry is to hold, manage and provide access to services.
→Service Registries are hierarchical. Service is one registry that can depend on and utilize services in the same registry as well as any parent registries.
e.g.BootStrapingServiceRegistry, StandardService Registry, SessionFactoryServiceRegistry



Note:-
All ServiceRegistries implement org.hibernate.service.Service Registry Interface directly or indirectly.
→The process of keeping Services with ServiceRegistry is called ServiceBinding and removing Services from ServiceRegistry is called Service Unbinding.
→To create ServiceRegistries (Container to manage Services) we use ServiceRegistryBuilders..
e.g. BootStrapServiceRegistryBuilder, StandardServiceRegistryBuilder and etc..
BootStrapServiceRegistry contains the following services:-
(a)ClassLoaderService
(b)IntegraterService
(c)StrategyService and etc..
→StandardServiceRegistry contains the following standard services:-
(a)JdbcService
(b)ConnectionProviderService
(c)DialectResolver
(d)BatchService and etc..

Modern BootStrapind Code:-

```
Configuration cfg= new Configuration();
cfg=cfg.configure("com/nt/cfgs/hibernate.cfg.xml");
//create ServiceRegistryBuilder
```

Modern BootStrapind Code:-

```
Configuration cfg= new Configuration();
cfg=cfg.configure("com/nt/cfgs/hibernate.cfg.xml");
//create ServiceRegistryBuilder
```


```
StandardServiceRegistryBuilder builder=new StandardServiceRegistryBuilder();
//create ServiceRegistry
StandardServiceRegistry= builder.applySettings(cfg.getProperties()).build();
//create SessionFactory
SessionFactory factory=new SessionFactory(registry);
//open Session session
Session ses=factory.openSession();
```

Bulk Operations in hiberante
==========

=> To select or manipulate 1 or more records /object having our choice criteria values.. we need to go for thise "Bulk Opeations" concept.

a) HQL/JPQL (2nd)
(HQL-Hibernate Query Language)
(JPQL -->Java Persistence Query language)

b) Native SQL (3rd)

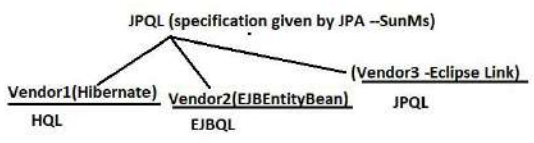
c) Criteria API or QBC (1st)
(QBC--Query By Criteria)

HQL/JPQL
========

=> It is objects based Query language i.e these queries will be written by usjng entity class names and their property names.. (not table db table names and col names)
=> HB dialect internally converts HQL queries into DB specific SQL Queries
=> HQL queries are DB s/w independent queries ,they help us to develop portable persistence logic
=> HQL supports both select and non-select operations
=> HQL can used for single row operations and also for bulk operations having our choice conditions/criterias
=> HQL supports both Positional prams (? (jdbc style) or ?<n> (jpa style) and named Params (:<name>)
note:: Hiberante 5.x is not supporting jdbc style positonal params(?) .. but supporing jpa style positional params (?<n> like ?1,?2,?3 and etc..)
=>HQL supports all condition clauses, operators, joins, aggrate functions, Projections and etc...
=> HQL keywords are not case-sensitive, but entity class names, propperties names used in HQL are case-sensitive
=> Query object represnets each HQL Query .. It is the object java class that implemetns
org.hibernate.query.Query<R> (I) (from 5.2 onwords)
(or)
org.hiberante.Query<R> (I)

=> To execute select HQL Queries call list() or iterate() method on Query Object.. and call executeUpdate() method on Query object to execute non-select HQL queries
=> The Learning curve of HQL is very small based on SQL knowledge..
note:: HQL insert Query can not be used to insert single record.. but can used insert records in to 1 db table by selecting them from another Db table
note:: HQL based PL/SQL programming is not possible..

=>JPQL is given by sun Ms as part of JPA Specification having rules to develop Objects based query language
HQL is the implementation of JPQL Specification given by Hibernate ORM f/w



SQL> SELECT * FROM EMP WHERE EMPNO>=? AND EMPNO<=?
db table name colname
HQL> FROM com.nt.entity.Employee WHERE eno>=?1 AND eno<=?2
Enity class Entity class property name
(or)
HQL> FROM com.nt.entity.Employee as e WHERE e.eno>=?1 AND e.eno<=?2
FROM alias name
HQL> SELECT e com.nt.entity.Employee e WHERE e.eno>=?1 AND e.eno<=?2

SQL> DELELE FROM EMP WHERE JOB=?

HQL> DELETE FROM com.nt.entity.Employee WHERE job=?1 (or)

HQL> DELETE FROM com.nt.entity.Employee WHERE job=:desg

if u r slecting all cols/properties in the HQL select query then place SELECT keyword in the HQL query is optional.

Why Jdbc style Positional params support is removed in hiberatne 5.x?
Ans) if query having more jdbc style plain postional params (?) then counting them and providing index for them while setting values is very complex.. (mostly it is errorProne process)
To overcome this problem , they have given jpa style postional params (?<n> like ?1,?2,...) and named params (:<name>)
note:: named params are more recomanded.. becoz they are self descriptive

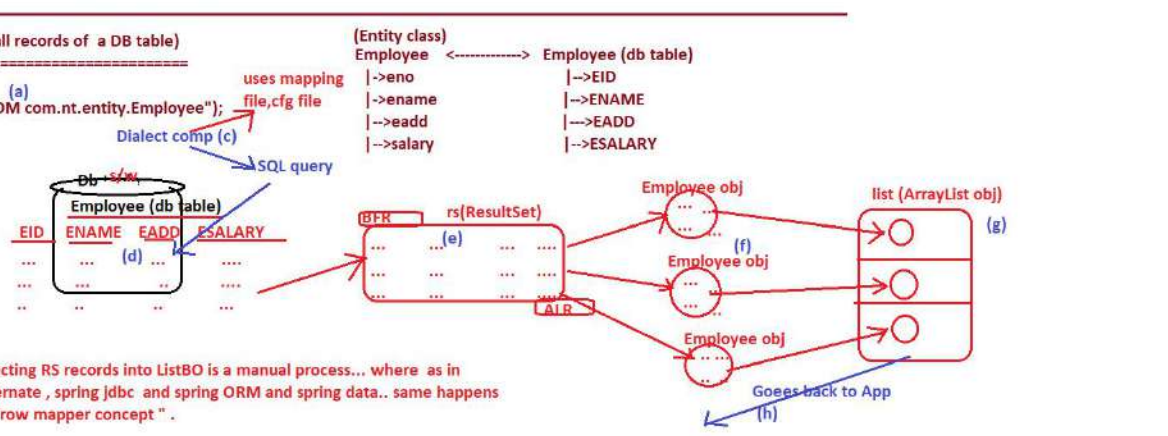
HQL Select Queries
==========

(a) Entity Queries (Getting All cols/all properites values of the records)
eg:: FROM com.nt.entity.Employee (with or with out condition)

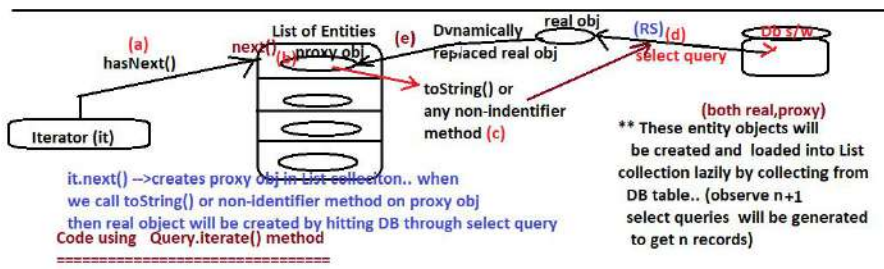
(b) Scalar Queries /Projections (Getting specific 1 column or specific multiple col values)
eg:: SELECT eno,ename,eadd FROM com.nt.entity.Employee (with or with out condition)
eg:: select eno FROM com.nt.entity.Employee (with or with out condition)
eg:: select count(*) FROM com.nt.entity.Employee

#Example1 (To get All col values of all records of a DB table)
==========

```
//prepare Query object
Query query=ses.createQuery(" FROM com.nt.entity.Employee");
//execute HQL query
List<Employee> list=query.list();
//process the results
list.forEach(emp->{
   S.o.println(emp);
} );
```



note:: in plain JDBC Connecting RS records into ListBO is a manual process... where as in ORM f/ws like Hibernate , spring jdbc and spring ORM and spring data.. same happens internally by using "row mapper concept " .



it.next() -->creates proxy obj in List collection.. when we call toString() or non-identifier method on proxy obj then real object will be created by hitting DB through select query

Code using Query.iterate() method
==========

```
//prepare Query object
Query<InsurancePolicy> query=ses.createQuery("FROM com.nt.entity.InsurancePolicy ");
//execute HQL query
Iterator<InsurancePolicy> it=query.iterate(); //retuns ListBO refered by Iterator object
//process Result
while(it.hasNext()) {
    InsurancePolicy policy=it.next();
    System.out.println(policy);
}
```

=>This iterate() method performs lazy loading of records with support Proxy objects created for the InMemory sub class of Entity class.. later these proxy object will be replaced real object created for the Entity class representing db table records..
mehod
=>This first generates 1 select SQL query getting all ids value.. then generates n SQL select queries having each recieved id value as the criteria value.

note:
iterate() is deprecated from the version 5.2 becoz its lazy loading for multiple records fetching hitting for multiple times on 1 hit per records.. which degrading the performence..

query1 :: SELECT POLICYID FROM INSURANCEPOLICY;
101 ,102 ,103

query2 :: SELECT * FROM INSURANCEPOLICY WHERE POLICYID=101 (1 proxy obj , 1 real obj)
... ... //101 record

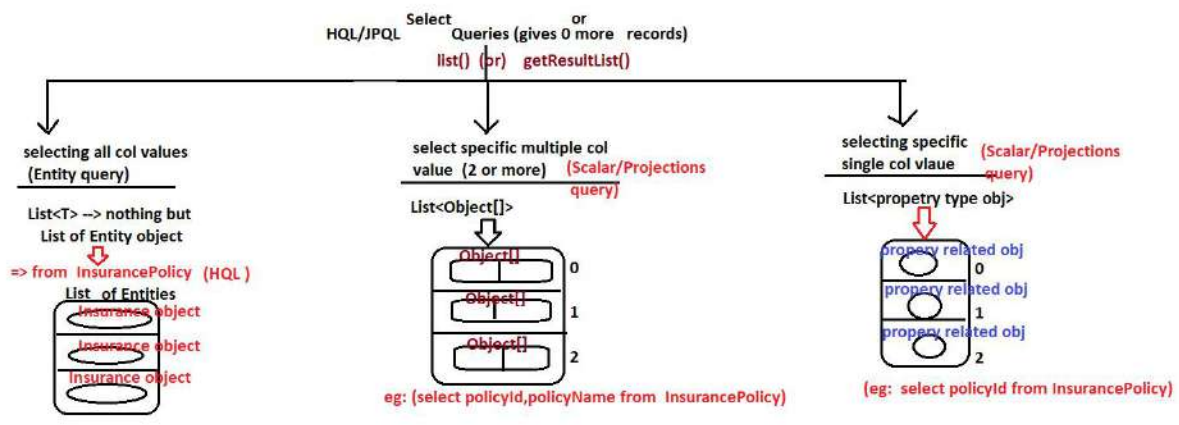
query3 :: SELECT * FROM INSURANCEPOLICY WHERE POLICYID=102 (1 proxy obj , 1 real obj)
... ... //102 record

query4 :: SELECT * FROM INSURANCEPOLICY WHERE POLICYID=103 (1 proxy obj , 1 real obj)
... ... //103 record

what the difference b/w Query.list() or Query.getResultList() and Query.iterate() method?

list() or getResultList()	iterate()
(a) perform eager loading of bulk records	(a) performs lazy loading of bulk records
(b) returns List Collection directly	(b) returns Iterator object pointing List collection
(c) generates single select SQL query to get n records from DB table	(c) generates 1+ n select SQL queries to get n records db table
(d) not a deprecated method	(d) deprecated method from HB 5.2
(e) Suitable for good performence	(e) Not suitable , becoz it degrades performence by hitting for multiple times
(f) for getting "n" records it hits the DB only for 1 time	(f) Hits DB for multiple times ..
(g) Does not create any Proxy objects	(g) Creates both Proxy objects and also real objects
(h) Entity class can be taken as final class becoz no Proxy class will be generated	(h) Entity class can not be taken final class becoz proxy class geneartion is required..
(i) It is industry standard	(i) not industry standard..

note: the lazy="true" attribute of <class> tag in mapping file ,@Proxy(lazy="true") are not related bulk operations in HB..(HQL,NativeSQL,QBC)
That option will be used only ses.load(-,-) method.. (no one else will use it)..



```
//=======================HQL SElect Entity query with JPA style positional Params ==================
//prepare Query object
Query<InsurancePolicy> query=
    ses.createQuery("FROM com.nt.entity.InsurancePolicy WHERE policyId>=?2 AND policyId<=?1 ");
//set query param values
/*  query.setInteger(1,9001);
    query.setInteger(2, 9003); deprecated from 5.2 */
    query.setParameter(2,6000L);
    query.setParameter(1, 10000L);
    //execute the query
    List<InsurancePolicy> list=query.getResultList();
    //process the collection
    list.forEach(System.out::println);
```

=> from HB 5.x there is no support for jdbc style plain positional params(?).. Exception will be thrown
=> we can use jpa style ordinal/positional params.. (?<n>) .. But
(a) we should take them either starting with ?0 or ?1 (overall availability)
(b) There should not any gap b/w parameter numbers but can be in any order
(c) negetive numbers are not allowed

eg1: FROM com.nt.entity.InsurancePolicy WHERE policyId>=?0 AND policyId<=?1 (valid)
eg2: FROM com.nt.entity.InsurancePolicy WHERE policyId>=?1 AND policyId<=?3 (invalid)
eg3: FROM com.nt.entity.InsurancePolicy WHERE policyId>=?1 AND policyId<=?2 (valid)
eg4: FROM com.nt.entity.InsurancePolicy WHERE policyId>=?-1 AND policyId<=?-2 (invalid)
eg5: FROM com.nt.entity.InsurancePolicy WHERE policyId>=?2 AND policyId<=?3 (invalid)
eg6: FROM com.nt.entity.InsurancePolicy WHERE policyId>=?2 AND policyId<=?1 (valid)

Limitations positional params /ordinal params
==========
=> we should explicity give/rember indexes.. for parameters
=> Parameters (? or ?<n>) are not self descriptive
=> Parameters (? or ?<n>) do not define their type..
=> Becomes complex to handler when more than 3/4 params are given..
=> Hibernate 5.x is not supporting jdbc style plain parameters(?)

=> To overcome the above problems take the support of named parameters

syntax:
:<name>

=> FROM com.nt.entity.InsurancePolicy WHERE policyId>=:<min> AND policyId<=:<max>

note:: all modern versions (hiernate 3.x onwords) of hibernate are supporting named params...

Avantages of named params
==========
=>named params are self descriptive
=> no need of giving any indexing or counting
=>all versions of hiberante support named params..
and etc..

note:: The JPA style postional/ordinal params(?<n>) and named parameter(:<name> will be converted into jdbc style plain params (?) along with SQL query generation becoz the jdbc code and DBs/w supports only plain params (?)

```
try {
    Transaction tx=ses.beginTransaction(); //dummy
    //prepare Query object
    Query<InsurancePolicy> query=ses.createQuery("FROM com.nt.entity.InsurancePolicy WHERE company IN(:org1,:org2)");
    //set values to named params
        query.setParameter("org1","LIC");
    query.setParameter("org2","TATA");
        //execute query
        List<InsurancePolicy> list=query.getResultList();
        //proess reuslt
        list.forEach(System.out::println);
        }//try
        catch(HibernateException he) {
            he.printStackTrace();
        }
        finally {
            HibernateUtil.closeSessionFactory();
    }
```

Can we place both named and postional params in one JPQL/HQL query?

Ans) yes We can but postional params must be jpa style postional/ordinal params..

FROM com.nt.entity.InsurancePolicy WHERE company IN(:org1,?2)

In Hiberate 4.x we can place plain postional params and named params in a HQL query but we must place all postional params before any named parameters are defined otherwise exception will come

=> FROM com.nt.entity.InsurancePolicy WHERE company IN(:org1,?) // invalid
=> FROM com.nt.entity.InsurancePolicy WHERE company IN(?,:org1) //valid

(these are not recomaded pratices.. So prefer all params as named params)

Q) can we take HQL keywords ,entity class names and property names as the params?
Ans) no , not possible ... we can take only input values through params

FROM :p1 WHERE company IN(:org1,:org2) // invalid
FROM com.nt.entity.InsurancePolicy WHERE company IN(:org1,:org2) // valid
:p1 :p2 WHERE :p3 IN(:org1,:org2) // invalid

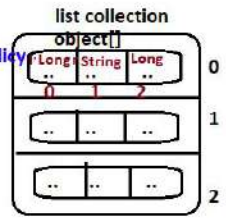
Q) Can we hard code input values in the Query
A) yes

FROM com.nt.entity.InsurancePolicy WHERE company IN(:org1,:org2,'BAJAJ')

=>Getting speciific multiple cols or specific single col vlaue from DB table is called Scalar query execution...

==== HQL scalar Query getting multiple specific col values =======

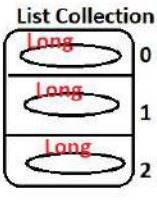
```
Query query=
ses.createQuery("SELECT policyId,policyName,tenure FROM com.nt.entity.InsurancePolicy
WHERE policyName like :initLetters ");
        //set values to query params
        query.setParameter("initLetters","J%");
        //execute HQL query
        List<Object[]> list=query.getResultList();
        //process result
        list.forEach(row->{
            for(Object obj:row) {
                System.out.print(obj+"  ");
            }//for
            System.out.println();
        });
```



=>Arrays are object in java .. so they can be stored in collections as elements..

// ========== HQL SELECT SCALAR Query that gives single specific col value========

```
//prepare Query object
Query query=
ses.createQuery("SELECT policyId FROM com.nt.entity.InsurancePolicy WHERE tenure>=:max");
    //set quer param value
    query.setParameter("max",25);
    //execute Query
    List<Long> list=query.getResultList();
        //process result
        list.forEach(System.out::println);
```



// ========== HQL SELECT SCALAR Query that gives single specific col value using iterate()========

```
//prepare Query object
Query query=ses.createQuery("SELECT policyId FROM com.nt.entity.InsurancePolicy WHERE tenure>=:max");
    //set quer param value
    query.setParameter("max",15);
    //execute Query
    Iterator<Long> it=query.iterate();
    //process result
while(it.hasNext()) {
    System.out.println(it.next());
}
```

Here single select SQL Query will be generated to get all records from Db table...

note:: query.iterate() method performs lazy loading only when it is dealing with Entity Queries (selecting all col values of Db table) .. not when it is dealing with scalar queries.. becoz Proxy class generatation for Entity class is possible .. but proxy class generation for non-Entity objects and their arrays not possible..

HQL Select Qurery (that gives 1 record)

list() | getResultList()

- Entity query (get all col values) — List<T> having 1 element
- Scalar query (specific multiple col values) — List<Object[]> having 1 element
- Scalar query (Specific single col value) — List<PropertyType obj> having 1 element.

```
//=================== HQL SELECT Entity Query that gives single record=====
            //prepare Query object
              Query query=ses.createQuery("FROM com.nt.entity.InsurancePolicy WHERE policyId=:id");
              //set param value
              query.setParameter("id",9007L);
              //execute Query
              List<InsurancePolicy> list=query.getResultList();
              System.out.println(list.size());
              if(!list.isEmpty()) {
                InsurancePolicy policy=list.get(0);
              System.out.println(policy);
              }
              else {
                    System.out.println("record not found");
              }
```

==========HQL Select Scalar Query(Aggrate function) ==========

```
//prepare Query object
            Query query=ses.createQuery("SELECT COUNT(*) FROM com.nt.entity.InsurancePolicy");
            //execute Query
            List<?> list=query.getResultList();
            if(!list.isEmpty()) {
              Long count=(Long) list.get(0);
          System.out.println("records count::"+count);
```

note:: if HQL SElect Entity query gives 0 or 1 record then instead of using list() or iterate() or getResultList() methods that give List Collection .. It is recomanded to use uniqueResult() or uniqueResultOptional() method to avoid memory wastage..

=>HQL Select Entity query giving single record ---> list()/getResusltList() ---> List<T> with 1 element
=>HQL Select Entity query giving single record ---> uniqueResult() --> Object (Entity object)
=>HQL Select Entity query giving single record ---> uniqueResultOptional() --> Optional object having Entity (Best)

```
//prepare Query object
            Query query=ses.createQuery(" FROM com.nt.entity.InsurancePolicy WHERE policyId=:id");
            //set query param value
            query.setParameter("id",9010L);
            //execute Query
            /*InsurancePolicy policy=(InsurancePolicy)query.uniqueResult();  (not so recomanded)
            if(policy!=null)
                  System.out.println(policy);
            else
                  System.out.println("record not found");*/
            Optional<InsurancePolicy>opt=query.uniqueResultOptional();   (best)
            if(opt.isPresent()) {
                  InsurancePolicy policy=opt.get();
                    System.out.println(policy);
            }
            else {
                  System.out.println("record not found");
            }
```

Optional api is introduced from java 8 .. very useful to hold object.. to check the availability object without expcitly performing "null" verification. multiple programmer friendly utility methods are given to simplicy the object programmer.. like

```
equals(Object obj) : boolean - Optional
filter(Predicate<? super InsurancePolicy> predicate) : Optional<
flatMap(Function<? super InsurancePolicy,? extends Optional<i
get() : InsurancePolicy - Optional
getClass() : Class<?> - Object
hashCode() : int - Optional
ifPresent(Consumer<? super InsurancePolicy> action) : void - O
ifPresentOrElse(Consumer<? super InsurancePolicy> action, Run
isEmpty() : boolean - Optional
isPresent() : boolean - Optional
map(Function<? super InsurancePolicy,? extends U> mapper) :
notify() : void - Object
notifyAll() : void - Object
or(Supplier<? extends Optional<? extends InsurancePolicy>> su
orElse(InsurancePolicy other) : InsurancePolicy - Optional
orElseGet(Supplier<? extends InsurancePolicy> supplier) : Insura
orElseThrow() : InsurancePolicy - Optional
orElseThrow(Supplier<? extends X> exceptionSupplier) : Insuran
stream() : Stream<InsurancePolicy> - Optional
```

note:: when should we use ses.get(-,-) /load(-,-) and when should we use Query.uniqueResult() / uniqueResultOptional() method?

Ans) To get single record based on id/pk value as criteria value use ses.get(-,-)/load(-,-) method. To get same single record by taking non id/pk value as the criteria value through HQL Query use Query,uniqueResult()/uniqueResultOptional() method.

Assignment:: Get Insurancepolicy details who is having nth(any number) highest tenure.. (Write HQL Query)

Assigmnet :: Get InsurancePolicy details whose having lowest policyId (Inner HQL)

Executing HQL Non-Select Queries

=> must be executed in Transaction env..
=> use Query.executeUpdate() method for this.. which gives numberic value representing no.of records that are effected.
=> HQL Insert query for direct record insertion not there... but to insert bulk records in to one db table by selecting them from another Db table is possible

```
//=================== HQL SELECT UPDATe Query ================
            boolean flag=false;
            Transaction tx=null;
            int count=0;
            try {
                  tx=ses.beginTransaction(); //mandatory
                  //prepare Query object
                  Query query=ses.createQuery(" UPDATE com.nt.entity.InsurancePolicy SET tenure=tenure
+:bonusYears WHERE policyName like :initialLetters");
                  //set query param value
                  query.setParameter("bonusYears",10);
                  query.setParameter("initialLetters","J%");
                  //execute Query
                  count=query.executeUpdate();
             flag=true;
            }//try
                  catch(HibernateException he) {
                        he.printStackTrace();
                        flag=false;
                  }
                  finally {
                        if(flag) {
                              tx.commit();
                              System.out.println("no.of records that are effected::"+count);
                        }
                        else {
                              tx.rollback();
                        }
                  HibernateUtil.closeSessionFactory();
                  }//finally
```

HQL Insert Query

=> HQL Insert is not given to insert one record to Db table directly.. becoz linking generators with HQL insert query is not possible.. So use ses.save(-) to insert a record.
=> But we can use HQL Insert Query to insert bulk records in to one db table by selecting them from another DB table
-> INSERT INTO ..... VALUES .. query is not given but
INSERT INTO ... SELECT FROM .. query is given .. (same DB)

copy InsurancePolicy db table records to Premium_InsurancePolicy db table only when tenure >=25

2 Db tables
==========

InsurancePolicy (existing db table)

	COLUMN_NAME	DATA_TYPE
1	POLICYID	NUMBER(10,0)
2	POLICYNAME	VARCHAR2(20 BYTE)
3	POLICYTYPE	VARCHAR2(20 BYTE)
4	COMPANY	VARCHAR2(20 BYTE)
5	TENURE	NUMBER

Premium_InsurancePolicy (can be generated dynamically)

	COLUMN_NAME	DATA_TYPE
1	POLICYID	NUMBER(10,0)
2	POLICYNAME	VARCHAR2(20 BYTE)
3	POLICYTYPE	VARCHAR2(20 BYTE)
4	COMPANY	VARCHAR2(20 BYTE)
5	TENURE	NUMBER

2 Entity classes
==========

InsurancePolicy (Entity class)
- company
- policyId
- policyName
- policyType
- tenure

PremiumInsurancePolicy (Entity class)
- company
- policyId
- policyName
- policyType
- tenure

mapping files (1 or 2)
==========
=> 1 mapping file with 2 <class> tags (or)
=> 2 mapping files seperately (recomanded)
|--->InsurancePolocy.hb.xml
|--->PremiumInsurancePolocy.hb.xml

1 DAO interfce, 1 DAO Impl class
==========

1 ClientApp
==========

HB cfg file
==========
hibernate.cfg.xml (1 file)
|--->specifying two mapping files..

DAO class
==========

```
public class InsurancePolicyDAOImpl implements IInsurancePolicyDAO {
  private static final String HQL_TRANSFER_INSURANCE_POLICIES="INSERT INTO
com.nt.entity.PremiumInsurancePolicy(pid,policyName,company,policyType,tenure) SELECT
i.policyId,i.policyName,i.company,i.policyType,i.tenure FROM com.nt.entity.InsurancePolicy as i WHERE
i.tenure>=:min ";
      @Override
      public String transferPremiumPolicies(int minTenure) {
            Session ses=null;
            Transaction tx=null;
            int count=0;
            Query query=null;
            boolean flag=false;
            String msg=null;
            //get Session
            ses=HibernateUtil.getSession();
            try {
                  //begin Tx
                  tx=ses.beginTransaction();
                  //prepare Query object
                   query=ses.createQuery(HQL_TRANSFER_INSURANCE_POLICIES);
                   //set values to query params
                   query.setParameter("min",minTenure);
                   //execute the Query
                   count=query.executeUpdate();
                   flag=true;
            }
            catch(HibernateException he) {
                  he.printStackTrace();
                  flag=false;
            }
            finally {
                  if(flag) {
                        tx.commit();
                  msg="no.of records that are copied::"+count;
                  }
                  else {
                        tx.rollback();
                        msg="record not copied/inserted";
                  }
            }
            return msg;
      }//method
}//class
```

```
HBProj39-HQLInsert
src/main/java
com.nt.cfg
hibernate.cfg.xml
com.nt.dao
IInsurancePolicyDAO.java
InsurancePolicyDAOImpl.java
com.nt.entity
InsurancePolicy.java
PremiumInsurancePolicy.java
InsurancePolicy.hbm.xml
PremiumInsurancePolicy.hbm.xml
com.nt.test
HQLInsertTest.java
com.nt.utility
HibernateUtil.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-13]
Project and External Dependencies
bin
gradle
src
build.gradle
gradlew
gradlew.bat
settings.gradle
```

Named HQL Queries

=> So far our HQL query is specific one SEssion object becoz Query object created having hard coded HQL query on HB Session object..
=>To make HQL Query accessible and executable through mulitple SEssion objs of multiple DAO classes or Client Apps we need go for Named HQL Query.. i.e we define HQL query in mapping file using <query> tag or in Entity class using @NamedQuery having logical name and we access and execute that HQL query in muliple DAOs and client Apps..

Example App
==========

step1) place named HQL query in the Mapping file after <class> tag(s)

```
<query name="HQL_INSERT_TRANSFER_POLICIES">
  <![CDATA[
      INSERT INTO com.nt.entity.PremiumInsurancePolicy(pid,policyName,company,policyType,tenure) SELECT
i.policyId,i.policyName,i.company,i.policyType,i.tenure FROM com.nt.entity.InsurancePolicy as i WHERE
i.tenure>=:min
  ]]>
</query>
```

note:: One mapping file can have any no.of named HQL queries using multiple <query> tags..

<![CDATA[.....]]> is taken to suppress xml meaning on the body of the tag for symbols like < , <= , & and etc..

step2) Get Access to Named Query in the client Apps or DAO classes and execute Query

```
//prepare Query object having access to NamedQuery
      Query query=ses.getNamedQuery("HQL_INSERT_TRANSFER_POLICIES");
      //set values to query params
       query.setParameter("min",minTenure);
       //execute the Query
      int count=query.executeUpdate();
```

The Name given the mapping file

=>For<query> tag @NamedQuery is given as alternate to place named HQL query Annotation driven entity classes.. upto Hiberante 4.x given based <=java7 @NamedQuery annotation is not repeatable annotation i.e we can not repeate annotation at same level To overcome this problem we can keep this annotation inside another annotation @NamedQueries for multiple times as param values..

```
@NamedQueries(value={@NamedQuery(name="HQL_QUERY1", query="FROM com.nt.entity.InsurancePolicy"),
                     @NamedQuery(name="HQL_QUERY2",query="DELETE FROM com.nt.entity.InsurancePolicy WHERE policyId=:id")
                    }
              )
public class InsurancePolicy implements Serializable{
  //properties
    ...
    ....
}
```

note:: Java8 annotations we have support for repeatable annotaions.. If annotation source code is having @Repeatable then that annotation is repeatable annotation in java.. In Hiberante 5.x is giving @NamedQuery Annotation as repeatable annotation.. So no need of @NamedQueries to use @NamedQuery annotation for multiple times on same Entity class.

```
@NamedQuery(name="HQL_QUERY1", query="FROM com.nt.entity.InsurancePolicy")
@NamedQuery(name="HQL_QUERY2",query="DELETE FROM com.nt.entity.InsurancePolicy WHERE policyId=:id")
public class InsurancePolicy implements Serializable{
  //properties
    ...
    ....
}
```

on

Soruce code::

```
@Repeatable(NamedQueries.class)
@Target({TYPE})
@Retention(RUNTIME)
public @interface NamedQuery {
  ....
}
```

Example @NamedQuery
==========

```
@NamedQuery(name="HQL_INSERT_TRANSFER_POLICIES",
            query="INSERT INTO
com.nt.entity.PremiumInsurancePolicy(pid,policyName,company,policyType,tenure) SELECT
i.policyId,i.policyName,i.company,i.policyType,i.tenure FROM com.nt.entity.InsurancePolicy as i WHERE
i.tenure>=:min")
public class PremiumInsurancePolicy implements Serializable {
    @Id
    private Long pid;
    private String policyName;
    private String policyType;
    private String company;
    private Integer tenure;
   //setters && getters
     ...
   //toString
     ....
}
```

In DAO class/ Client App
==========

```
//prepare Query object having access to NamedQuery
Query query=ses.getNamedQuery("HQL_INSERT_TRANSFER_POLICIES");
      //set values to query params
       query.setParameter("min",minTenure);
       //execute the Query
      int count=query.executeUpdate();
```

Pagination
==========

=> The process of displaying huge no.of records page by page is called pagination .. This avoid loading all records .. i.e loads only those records/objects that are required to display in the current page to improve performence

eg:: Gmail Inbox , report generation, google search results and etc..

To perform pagination directly by using SQL query we need to complex query using the psuedo column rownum

```
SELECT outer.*
  FROM (SELECT ROWNUM rn, inner.*
     FROM (  SELECT e.*
           FROM employee e  ) inner) outer
  WHERE outer.rn >=1 AND outer.rn <=3
                 4                 6
                 7                 9
```

=>we are assuming the page size is ::3

"rownum" is invisble psuedo column of Db table having record numbers..

=>Hibernate simplifies the pagination process by giving two simple methods.. i.e the complex SQL query will be generanted dynamically by hibernate s

a) setFirstResult(-) b) setMaxResult(-)

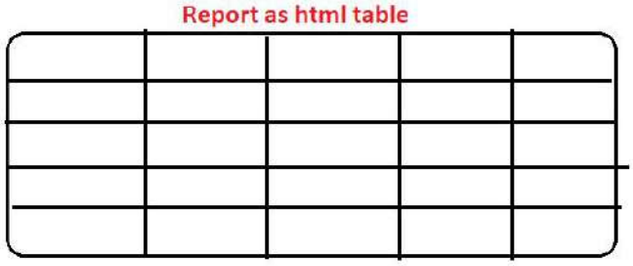
startPos page size

```
Query query=ses.createQuery("FROM com.nt.entity.Employee");
query.setFirstResult(0/3/6/...);
query.setMaxResults(3);          //pagination settings

List<Employee> list=query.list();
```

Example code
==========

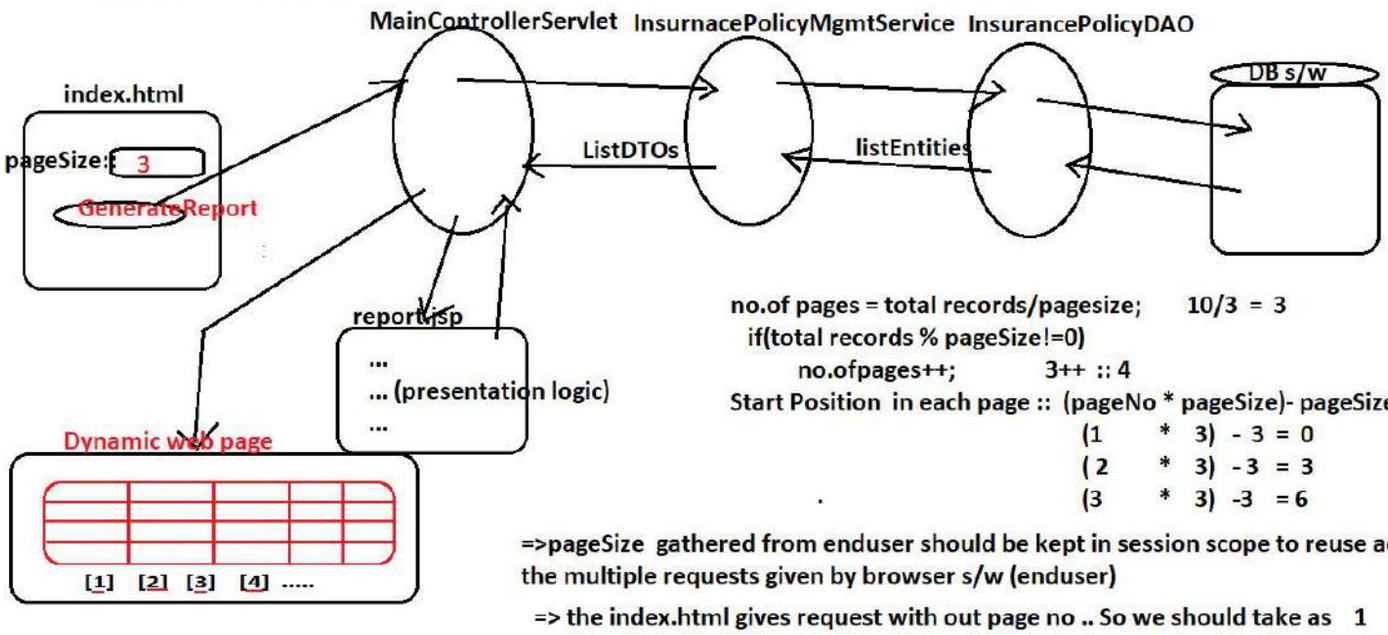
```
Query query=ses.createQuery("FROM com.nt.entity.InsurancePolicy");
      //pagination settings
      query.setFirstResult(0/2/4/6/...); //page start Position
      query.setMaxResults(2); // page size
//execute Query
      List<InsurancePolicy> list=query.getResultList();
      //process the reuslts
list.forEach(System.out::println);
```

Report as html table



[1] [2] [3] [4] .....

html/jsp -----> controller Serlvet -----> service class ---------->DAO ----------> DB s/w



no.of pages = total records/pagesize; 10/3 = 3

if(total records % pageSize!=0)

no.ofpages++; 3++ :: 4

Start Position in each page :: (pageNo * pageSize)- pageSize;

(1 * 3) - 3 = 0

(2 * 3) - 3 = 3

(3 * 3) -3 = 6

=>pageSize gathered from enduser should be kept in session scope to reuse across the multiple requests given by browser s/w (enduser)

=> the index.html gives request with out page no .. So we should take as 1

=> while working factory.getCurrentSession() based Contextual Session object , we need begin Tx only when there is no Active Tx in all persistence methods of DAO class ..becoz 1 request /1 task may call multiple persistence methods.. of DAO classes to complete the job.

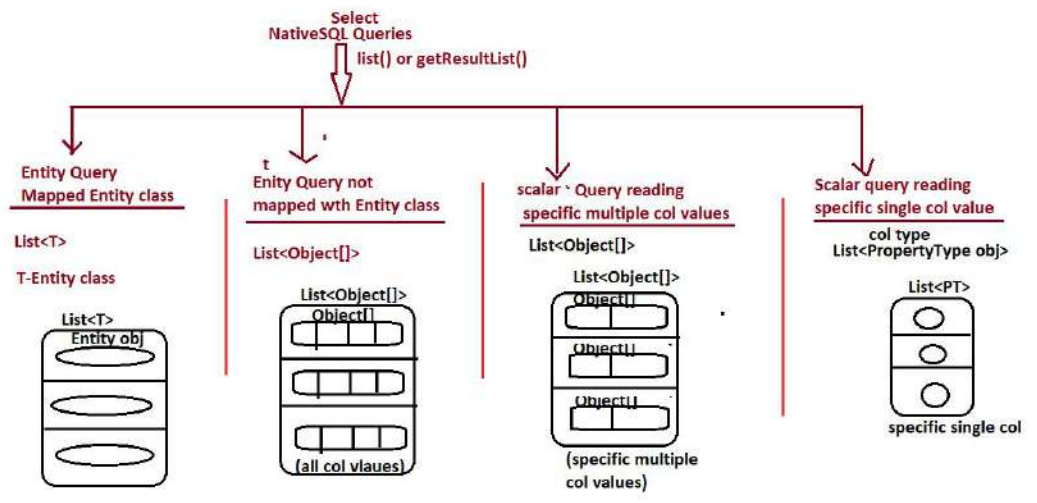
```
if(!ses.getTransaction().isActive())
        tx=ses.beginTransaction();//dummy
```

NativeSQL
==========

=>given to execute plain SQL queries that are supported by underlying DB s/w
=> Use this option only for the operations that are not possible using HQL like HQL Insert for single record insertion
=> Supports both select and non-select Operations
=> Supports both named , positional params (jdbc style ? and jpa style ?<n>)
=> These queries performence is bit good .. becoz they go and execute in Db s/w directly with out any translation
=> Useful while working with DB s/w specific SQL queries and features..
=>Mainly given to call PL/SQL procedures /functions
=> SQLQuery object represents each Native SQL query .. (Now it is NativeQuery object)
=> there is no lazy loadng with Native SQL queries .. i.e we can not call iterate() on SQLQuery object
=>SQLQuery object means it is the object of java class that implemetns org.hb.query.SqlQuery (I).

=>supports NamedNativeSQL queries with the support of @NamedNativeQuery, @NamedNativeQueries annotations
=> These queries will be written by specifying DB table name and col names .
=>*To excute non-select native SQL Query use executeUpdate() on SQLQuery object.*

note: org.hiberante.SQLQuery(I) deprecated and the alternate is org.hibernate.query.NativeQuery (I)



=====Native SQL Entity Query with out mapping with Entity class ==========

```
//create SQL Queyr object
        NativeQuery<Object[]> squery=ses.createSQLQuery("SELECT * FROM INSURANCEPOLICY WHERE TENURE>=? AND TENURE<=?");
         //set query param values
        squery.setParameter(1,25);
        squery.setParameter(2,35);
        //exsecute Query
        List<Object[]> list=squery.getResultList();
        //process results
        list.forEach(row->{
            for(Object o:row) {
                System.out.print(o+"  ");
            }
            System.out.println();
        });
```

//==========Executing Native SQL Entity Query by mapping it to entity class ==========

```
//create SQL Queyr object
        NativeQuery<InsurancePolicy> squery=ses.createSQLQuery("SELECT * FROM INSURANCEPOLICY WHERE TENURE>=?1 AND TENURE<=?2");
         //set query param values
        squery.setParameter(1,25);
        squery.setParameter(2,35);
        //map results with Entity class
        squery.addEntity(InsurancePolicy.class);
        //exsecute Query
         List<InsurancePolicy> list=squery.getResultList();
         //process results
         list.forEach(System.out::println);
```

//==========Executing Native SQL Scalar Query selecting specific multiple col values =======

```
NativeQuery<Object[]> squery=
                ses.createSQLQuery("SELECT POLICYID,POLICYNAME FROM INSURANCEPOLICY WHERE COMPANY IN(:comp1,:comp2)");
         //set query param values
        squery.setParameter("comp1","LIC");
        squery.setParameter("comp2","TATA");
        //map results with Hibernate datypes
    squery.addScalar("POLICYID",StandardBasicTypes.INTEGER);
    squery.addScalar("POLICYNAME",StandardBasicTypes.STRING);
        //exsecute Query                    HB Data type
            List<Object[]> list=squery.getResultList();
            //process results
            list.forEach(row->{
                for(Object o:row) {
                    System.out.print(o+"  "+o.getClass());
                }
                System.out.println();
            });
```

=>It is recomaned to map Entity query results with Entity classes and scalar query results with Hibernate Data types...

note:: There is no support to call iterate() method on SQLQuery/NativeQuer obj.. i.e native SQL queries do not support Lazy Loading..

org.hibernate.query.Query<T> (I)
↑ extends
org.hiberante.query.NativeQuery<T> (I)

//========== Inserting record with direct values ==========

```
Transaction tx=null;
    int count=0;
    boolean flag=false;
    try {
        //Locate/Begin Tx
        if(!ses.getTransaction().isActive())
            tx=ses.beginTransaction();
        //create NativeQuery object having Native SQL query
        NativeQuery nquery=ses.createSQLQuery("INSERT INTO INSURANCEPOLICY VALUES(POLICYID_SEQ.NEXTVAL,?,?,?,?)");
        //set values to query params
        nquery.setParameter(1,"JA++");
        nquery.setParameter(2, "Life");
        nquery.setParameter(3,"TATA");
        nquery.setParameter(4,15);
        //execute the query
        count=nquery.executeUpdate();
        flag=true;
    }//try
    catch(HibernateException he) {
        he.printStackTrace();
        flag=false;
    }
    catch(Exception e) {
        e.printStackTrace();
        flag=false;
    }
    finally {
        if(flag) {
            tx.commit();
            System.out.println("Record insert :: count"+count);
        }
        else {
            tx.rollback();
            System.out.println("record not inserted");
        }
    }//finally
```

=>Native SQL insert query does not deal with generators cfg in mapping file or entity class.. only ses.save(-) method deals generators that are configured..

NamedNativeSQL queries
==========

=>To provide global visibility to SQL Query across the multiple Session objs of same or different dAO classes or Client Apps.. we need to place native SQL queries having logical name either in mapping file or in Entity class
=> In mapping file tag name is <sql-query> tag
=> In Entity class annotations are @NamedNativeQueries , @NamedNativeQuery

Example using Xml driven cfgs (NamedNativeSQL queries)
==========

step1) place named native SQL query in mapping file

after </class> tag
==========

```
<sql-query name="GET_ALL_POLICIES_TYPE">
    <![CDATA[  SELECT POLICYID,POLICYNAME,POLICYTYPE FROM INSURANCEPOLICY WHERE POLICYTYPE=?  ]]>
 </sql-query>
```

step2) In DAO /Client app access and execute the SQL query

```
//get Access to Named NativeSQL query
    NativeQuery<Object[]> nquery=
        ses.getNamedNativeQuery("GET_ALL_POLICIES_TYPE");
    //set value to query params
    nquery.setParameter(1,"LIFE");
    //execute query
    List<Object[]> list=nquery.getResultList();
    list.forEach(row->{
    for(Object o:row)
        System.out.print(o+"  ");
        System.out.println();
    });
```

Example on NamedNativeSQL query using @NamedNativeQuery annoation
==========

```
@NamedNativeQuery(name = "GET_ALL_POLICIES_TYPE",
            query="SELECT POLICYID,POLICYNAME,POLICYTYPE FROM INSURANCEPOLICY WHERE POLICYTYPE=?")
@NamedNativeQuery(name = "GET_ALL_POLICIES",
             query="SELECT * FROM INSURANCEPOLICY",
             resultClass = InsurancePolicy.class)
@NamedNativeQuery(name = "DELETE_POLICY_BY_ID",
                query="DELETE FROM INSURANCEPOLICY  WHERE POLICYID=:id")
public class InsurancePolicy implements Serializable {
  ....
  ....
}
```

Calling PL/SQL Procedure /function (Stored Procedure/Functiion) of DB s/w Uisng Hiberante Persistence logic

=> To make b.logic or persitence logic reusable across the multiple modules of a Project by keeping logics in DB s/w . then we go for PL/SQL Procedures/functions

eg1:: Authentication logic is to required in every module of the Project..So place in DB s/w only for 1 time as PL/SQL procedure or function and use it for multiple times..

eg2:: Attendence calculation logic is required in every module of the University Project..So place in DB s/w only for 1 time as PL/SQL procedure or function and use it for multiple times..

=> PL/SQL procedure does not return a value, To send n results/outputs from PL/SQL Procedure we need to use "n" out params

=> PL/SQL function return a value, To send n results/outputs from PL/SQL function we need to use "n-1" out params and 1 return value.

IN PL/SQL proecure or function we will have type,mode for params

the modes are

a) IN mode (default)
b) OUT mode — y=x*x; — x -->IN mode , y->out mode
c) INOUT mode — x=x*x; — x- INOUT mode

Calling PL/SQL procedure/function in hibernate

|--->1.Legacy/Old Approach (upto hibearnte 5.2)

|--> PL/SQL Procedure/function must designed according to hibernate rules and these rules will vary DB s/w to DB s/w..

|-->We need to use Named Native SQL queries to call PL/SQL procedures/ functions

This approch removed from hiberante 5.3 version

--->2. Modern Approach (upto hibearnte from 5.3 onwords)

|--> No rules are required to follow while creating PL/SQL procedure or function

|--> We need to use "org.hibernate.procedure.ProcedureCall" obj to call PL/SQL procedure or function

Introduced in 4.x and continued in 5.x .. From HB 5.3 only this approach is allowed..

Example1 (Calling PL/SQL procedure that Performs Autthentication)

==========

step1) create db table and PL/SQL procedure ready..

USERINFO

	UNAME	PWD
1	raja	rani
2	ramesh rao	suresh
3	jani	begum
4	dhoni	cricket

PL/SQL procedure in Oracle

```
CREATE OR REPLACE PROCEDURE P_AUTHENTICATION
(
 USERNAME IN VARCHAR2
, PASSWORD IN VARCHAR2
, RESULT OUT VARCHAR2
) AS
 CNT NUMBER(5);
BEGIN
 SELECT COUNT(*) INTO CNT FROM USERINFO WHERE UNAME=USERNAME AND PWD=PASSWORD;

 IF(CNT<>0) THEN
  RESULT:='VALID CREDENTIALS';
 ELSE
  RESULT:='INVALID CREDENTIALS';
 END IF;
END P_AUTHENTICATION;
```

step2) create Maven Project in Eclipse

file menu---> new ---> maven Project ---> select "maven-archetype-quickstart" (Project Template Toolkit for standalone App) --> provide

groupId: nit
artifactId : HBProj44-CallingPL-SQLProcedure
Version:: default
package :com.nt.test

step3) do following things on project

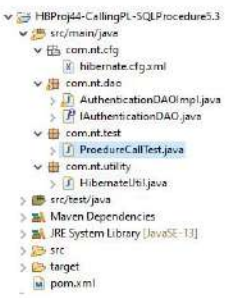
a) change JRE version to 13/...
b) change compiler version 13/...
c) change java version in Project facets to 13

step4) add the following dependencies in pom.xml (like build.gradle)

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.4.22.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-hikaricp -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-hikaricp</artifactId>
  <version>5.4.22.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc8 -->
<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc8</artifactId>
  <version>19.7.0.0</version>
</dependency>
</dependencies>
```

step4) add the following packages to src/main/java folder

|--->com.nt.cfgs
|--->hibernate.cfg.xml
|--->com.nt.dao
|--->IAuthenticationDAO.java
|--->AuthencationDAOImpl.java
|--->com.nt.test
|---->ProcedureCallTest.java
|--->com.nt.utility
|--->HibernateUtil.java



Code calling PL/SQL procedure

==========

```
//create Procedure Call object
 ProcedureCall call=ses.createStoredProcedureCall("P_AUTHENTICATION");
  //register IN ,OUT params with JDBC types and also IN params with values
   call.registerParameter(1,String.class, ParameterMode.IN).bindValue(username);
   call.registerParameter(2,String.class, ParameterMode.IN).bindValue(password);
   call.registerParameter(3,String.class,ParameterMode.OUT);
   //call PL/SQL procedure
 String result=(String) call.getOutputParameterValue(3);
```

Calling PL/SQL Procedure of oracle DB s/w from HibernateApp that deals with Entity query

=> Cursor is a in memory variable of oracle's PL/SQL programming that can hold bunch of records given select SQL query.. It is like JDBC ResultSet object..

=>sys_refcursor is built-in cursor datatype

eg:: In PL/SQL procedure

```
details   sys_refcursor;

open details
   SELECT * FROM  INSURANCEPOLICY WHERE  TENURE>= ..   and TENURE<= ..
```

the records given the select SQL query will be stored into cursor variable automatically

PL/SQL procedure in Oracle

```
CREATE OR REPLACE PROCEDURE P_GET_POLICIES_BY_TENURE
(
 MINTENURE IN NUMBER
, MAXTENURE IN NUMBER
, DETAILS OUT SYS_REFCURSOR
) AS
BEGIN

 OPEN DETAILS FOR
  SELECT POLICYID,POLICYNAME,POLICYTYPE,COMPANY,TENURE FROM
        INSURANCEPOLICY WHERE TENURE>=MINTENURE AND TENURE<=MAXTENURE;

END ;
```

Code in DAO class

```
@Override
    public List<InsurancePolicy> getPolociesByTenure(int start, int end) {
        Session ses=null;
        Transaction tx=null;
        ProcedureCall call=null;
        Output output=null;
        ResultSetOutput rsOutput=null;
        List<InsurancePolicy> list=null;
        //get Session object
        ses=HibernateUtil.getSession();
        // begin Tx
        if(!ses.getTransaction().isActive())
            tx=ses.beginTransaction();
        //create PorcedureCall object
        call=ses.createStoredProcedureCall("P_GET_POLICIES_BY_TENURE" ,InsurancePolicy.class);
        //register both IN,OUT params and set values to IN params
        call.registerParameter(1, Integer.class,ParameterMode.IN).bindValue(start);
        call.registerParameter(2, Integer.class,ParameterMode.IN).bindValue(end);
        call.registerParameter(3,Class.class,ParameterMode.REF_CURSOR );
        //call PL/SQP proceure  and get ouput
        output=call.getOutputs().getCurrent();
        rsOutput=(ResultSetOutput)output;
        //get ResultList object from rsOutput
        list=rsOutput.getResultList();
        return list;
    }
```

refer HBProj45

note:: we can not perform pagination while calling PL/SQL procedure from Hibernate

---

=Hibernate Modern approach is not giving any provision to call PL/SQL function , So we work with Callback Interfaces like ReturningWork(I) which expsoes jdbc con object to create other jdbc objs and to call PL/SQL function using CallableStatement object (plain jdbc code)

Work(I)
ReturningWork(I)

These are hibernate supplied callback interfaces to write plain jdbc code by using the recived jdbc con object.

Work(I)
|--->void excute(Connection con) [Callback method]
ReturningWork<T>(I)
|--><T> execute(Connection con) [Callback method]

Functional interfaces..

The method that will be called automatically underlying HB session object having jdbc con object is called Callback method.. The interface that contains such callback methods is called callback interface..

Example
=========

PL/SQL function in oracle

```
CREATE OR REPLACE FUNCTION FX_GET_POLICY_DETAILS_BY_ID
(
 ID IN NUMBER
, NAME OUT VARCHAR2
, DURATION OUT NUMBER
) RETURN VARCHAR2 AS
 FIRM VARCHAR2(10);     //local variable
BEGIN

 SELECT POLICYNAME,COMPANY,TENURE INTO NAME,FIRM,DURATION FROM INSURANCEPOLICY WHERE POLICYID=ID;

 RETURN FIRM;
END;
```

```
//code dAO class
=================
@Override
    public String[] getPolocyById(int id) {
        Session ses=null;
        Transaction tx=null;
      //get Session object
        ses=HibernateUtil.getSession();
        //create/Locate Tx
        if(!ses.getTransaction().isActive())
            tx=ses.beginTransaction();
         String result[]=ses.doReturningWork(new ReturningWork<String[]>() {

            @Override
            public String[] execute(Connection con) throws SQLException {
                CallableStatement cs=null;
                String outputs[]=null;
                //create CallableStatement object
                cs=con.prepareCall("{?=call  FX_GET_POLICY_DETAILS_BY_ID(?,?,?)}");
                //register OUT,RETURN params with JDBC types
                cs.registerOutParameter(1, Types.VARCHAR);
                cs.registerOutParameter(3,Types.VARCHAR);
                cs.registerOutParameter(4, Types.INTEGER);
                //set values to IN param
                cs.setInt(2,id);
                //execute PL/SQL function
                cs.execute();
                //gather results  from return ,out params
                outputs=new String[3];
                outputs[0]=cs.getString(1); //company
                outputs[1]=cs.getString(3); //name
                outputs[2]=String.valueOf(cs.getInt(4));
            return outputs;
            }//method
        }//anonymous inner class
         ); //method call

        return result;
    }//method
```

Anonymous inner class obj is passed here whose class is implementing ReturningWork(I)

refer HBProj45

Criteria API /QBC (QueryByCriteria)

==========

=>Allows to write entire peristtence (both singlerow and bulk operations) as java statements with out using any queries..

=>Criteria Api Persistnece logic is based on Entity class and properties ..So it is DB s/w independent Persistence logic

=>Criteria API Persistence logic will generate SQL queries based on the given entity class names and properties names

=>Criteria API does not support non-select Persinstence operations , But both Entity and scalar select operations are supported

=>Supports Pagination.

=>No support for calling PL/SQL procedures and functions

=>Does not support DDL Operations

=> Since everything is java code.. So there is no need of working with named Queries

Two modes of writing QBC/Criteria API Logic

==========

a) HB QBC/Criteria API (Specific to hibernate env..) (from hibernate initial days)

b) JPA QBC/Criteria API (common for all ORM f/ws) (from hibernate 4.x)

a) HB QBC /Criteria API Logic

==========

=> It is specific Hibernate ORM.. and this code does not execute in other ORM f/ws..

=>Criteria object represents the base Entity class .. which internally generate Select query with out condition.. By adding more Criterion objects to that Criteria object we can add where clause conditions to that select query.

=> criteria.list() is given to execute QBC logics.. This generates SQL queries internally.

=>Using Projections Concept we can deal with scalar select opeations

=> we can add total 3 objects on Criteria object

a) Criterion objects (for where clause conditions)

b) Projection objs (for scalar select operations)

c) Order objects (for orderBy operations)

Sample code (To get all records of Project DB table)

==========

Project(entity class)--->Project (db table)

internally builds SELECT * FROM PROJECT

```
//create Criteria object
Criteria criteria=ses.createCriteria(Project.class);
//execute QBC logic
List<Project> list=criteria.list();
//process the reults
list.forEach(System.out::println);
```

Criteria object means it is the object of java class that implments org.hibernate.Criteria(I)

con2 | PROJECT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Detai

Sort.. | Filter:

	PROJID	COMPANY	COST	LOCATION	PROJNAME	TEAMSIZE
1	121	POLARIS	1000000	hyd	openFx	13
2	4533	Infosys	1000000	delhi	citiBank	12
3	234	CTS	200000	hyd	amex	12
4	456	cts	300000	chennai	amerprose	10
5	6	POLARIS	100000000	hyd	openFx1	13
6	8	POLARIS1	100000000	hyd1	openFx2	13

note:: QBC/Criteria API logics does not need any cfgs in mapping file(xml) or using mapping annotations

note:: we can change the default jdk version of maven (that is 1.7) to other version in two ways

a) Using Java version of Project Facects

b) Using

```
<maven.compiler.source>13</maven.compiler.source>
<maven.compiler.target>13</maven.compiler.target>
```

of pom.xml + maven update (right click on project --> maven --> update Poject -->select force snapshot

---

Criteria with Criterion , Order objects

==========

```
//prepare Criteria object
Criteria criteria=ses.createCriteria(Project.class);
//prepare Criterion objects
Criterion cond1=Restrictions.ge("teamSize",10);
Criterion cond2=Restrictions.le("teamSize",20);
//add Criterion objs to Criteria object
criteria.add(cond1); criteria.add(cond2);
//Order object
Order order=Order.desc("projName");
criteria.addOrder(order);
//execute QBC logic
List<Project> list=criteria.list();
//process results
list.forEach(System.out::println);
```

=>The Restrictions.xxx() methods return Criterion objects to add to Criteria object..

What is the diff b/w Criteria Object and Criterion object?

Ans) Criteria object means it is the object of java class that implemetns org.hiberante.Criteria(i) and it represents the base Entity class for which select query will be generated with out condtions

=> Criterion object is the object of java class that implements Criterion(I) and it represents each where condition on base select Query given Criteria object. We add multiple Criterion objects on Criteria object.

note:: the multiple criterion objects added to Criteria object will execute by having "and" clause.

//========== Adding multiple condtions (Criterion objs) with or and and clauses to Criteria object ==========

```
//prepare Criteria object
    Criteria criteria=ses.~~createCriteria~~(Project.class);
    //prepared Criteria objects
Criterion cond1=Restrictions.between("teamSize", 10, 20);
//Criterion cond2=Restrictions.in("company",List.of("TATA","POLARIS"));
Criterion cond2=Restrictions.in("company","TATA","POLARIS");
Criterion cond3=Restrictions.ilike("projName","o%");
//create Criterion object having and clause b/w cond1,cond2 , or caluse with cond3
    Criterion finalCond=Restrictions.or(Restrictions.and(cond1,cond2),cond3);
//add Criterion obj to Criteria obj
    criteria.add(finalCond);
//execute QBC logic
List<Project> list=criteria.list();
list.forEach(System.out::println);
```

//========== Adding Sqlcondtion based Criterion object ==========

```
//prepare Criteria object
Criteria criteria=ses.~~createCriteria~~(Project.class);
//create Criterion object
Criterion  cond=Restrictions.sqlRestriction("rowNum>=? and rownum<=?",
                                            new Object[] {1,3},
                                            new Type[]
{StandardBasicTypes.INTEGER,StandardBasicTypes.INTEGER});
                                    //add Criterion object to Criteria object
                                    criteria.add(cond);
                                    //execute QBC logic
                                    List<Project> list=criteria.list();
                                    list.forEach(System.out::println);
```

=>if certain conditions can not be writtern by using property names or Restrictions.xxx() methods then we can write same condition by using SQL statement.. using the support of Restrictions.sqlRestriction(-)

=>To perform Scalar select operations using HB -QBC (selecting specific muliple col values or speicific single col value or aggragate results) then go for Projection concept..

========== getting specific single property value using Projections ==========

```
// Create Criteria object
Criteria criteria=ses.~~createCriteria~~(Project.class);
//prepare Projection for single property
Projection p1=Projections.property("company");
//set Projection object to Criteria object
criteria.setProjection(p1);
//prepare Criterion object
Criterion cond=Restrictions.eq("location","hyd");
//add Criterion object to Criteria obj
criteria.add(cond);
//execute QBC logic
List<String> list=criteria.list();
//process resukts
list.forEach(System.out::println);
```

```
Criterion cond=Restrictions.like("company","_____");
```

company name whose name is having 5 chars

//=== Getting specific multiple col values using HB-QBC Projections=====

```
// Create Criteria object
Criteria criteria=ses.~~createCriteria~~(Project.class);
//prepare Projection objects for multiple properties
Projection p1=Projections.property("company");
Projection p2=Projections.property("projId");
//add Projections to ProjectionList object
ProjectionList pList=Projections.projectionList();
pList.add(p1); pList.add(p2);
//create ProjectionList to Criteria
criteria.setProjection(pList);
//execute QBC logic
List<Object[]> list=criteria.list();
//process results
list.forEach(row->{
    System.out.println(row[0]+"......"+row[1]);
});
```

//=== Getting single aggregate result using HB-QBC Projections

```
// Create Criteria object
Criteria criteria=ses.~~createCriteria~~(Project.class);
//prepare Projection object for aggragate result
Projection p1=Projections.max("cost");
//add Projection to Criteria
criteria.setProjection(p1);
    //execute QBC logic
    List<Long> list=criteria.list();
//process results
System.out.println("records count::"+list.get(0));
```

Assigment1:: write HB-QBC logic to get specific 3 col values by 2 condtions in desc order
Assigment1:: write HB-QBC logic to get multiple aggragate results..

QBC supports pagination
==========

```
Criteria criteria=ses.~~createCriteria~~(Project.class);
//pagination settings
criteria.setFirstResult(1);
criteria.setMaxResults(3);
//execute QBC logic
List<Project> list=criteria.list();
//process resukts
list.forEach(System.out::println);
```

HQL/JPQL	nativeSQL	QBC
a)Persistence logic Db s/w independent	a)dependent	a) independent
b) query will be written using entity class and property names	b) query will be written using db table name and col names	b) QBC logic is java code that will be written using Entity class name and property names
c) supports only jpa style positional params and named params	c) supports jdbc style positional params, named params, jpa style params	c) No params are required..
d) supports boht select and non-select operation	d) same	d) Only select operation
e) not suitable for calling PL/SQL procedures and functions	e) suitable for calling PL/SQL procedures and functions	e) not suitable

JPA-Criteria or JPA-QBC
==========

=> this qbc logic is common in all orm frameworks

=>Introduced in hibernate 4.x , but highlighted from hb 5.3

=> Both select and non-select operations operations are possible..

```
//========== Executing entity Query ==================

//create CriteriaBuiler object
    CriteriaBuilder     builder=ses.getCriteriaBuilder();
        //create CriteriaQuery object
    CriteriaQuery<Project>  ctQuery=builder.createQuery(Project.class);
    //Create Root obj specifying Entity class   /table name from which records should be selected
    Root<Project> root=ctQuery.from(Project.class);
    //add root object to CriteriaQuery object
    ctQuery.select(root);
    //prepare Query object having ctQuery obj
    Query<Project> query=ses.createQuery(ctQuery);
    //execute JPA QBC logic
    List<Project> list=query.getResultList();
    //process the Result
    list.forEach(System.out::println);
```

=>CriteriaBuilder obj is given to construct criteria queries, compound selections, expressions, predicates, orderings.

=>CriteriaQuery object defines functionality that is specific to top-level queries.

=>Root object defines a root type in the from clause. Query roots always reference entities.

=>Query object represents HQL/JPQL queries, compiled CriteriaQuery, CriteriaUpdate, CriteriaDelete objects.

#Above code using Method Chaining concept

```
//create CriteriaBuiler object
CriteriaBuilder     builder=ses.getCriteriaBuilder();
    //create CriteriaQuery object
CriteriaQuery<Project>  ctQuery=builder.createQuery(Project.class);
//Create Root obj specifying Entity class   /table name from which records should be selected
Root<Project> root=ctQuery.from(Project.class);
ctQuery.select(root).
            where(builder.and(builder.ge(root.get("projId"),100L),
                    builder.le(root.get("projId"),2000L))).orderBy(builder.desc(root.get("projName")));
//prepare Query
Query query=ses.createQuery(ctQuery);
//execute JPA -QBC
List<Project> list=query.getResultList();
//process result
list.forEach(System.out::println);
```

=====JPA QBC with condtions ==========

```
//create CriteriaBuiler object
    CriteriaBuilder     builder=ses.getCriteriaBuilder();
        //create CriteriaQuery object
    CriteriaQuery<Project>  ctQuery=builder.createQuery(Project.class);
    //create Root object
    Root<Project> root=ctQuery.from(Project.class);
    //prepare condtions
    //SQL :: select * from Project where  teamSize between(10,20)  and  projName like 'o%'
    ctQuery.select(root).
            where(builder.
                    and(builder.between(root.get("teamSize"), 10,20),
                        builder.like(root.get("projName"),"o%")
                        )
                  );


    //prepare Query
    Query query=ses.createQuery(ctQuery);
    //execute JPA -QBC
    List<Project> list=query.getResultList();
    //process result
    list.forEach(System.out::println);
```

Assignment1:: get Project details whose names are having 3 characters

Assignment1:: get Project who are having highest team size

Executing Scalar JPA QBC Logic
==========
=> It is all about getting specific muliple col values or single col vlaues or aggragate results...

Executing scalar select Query (Getting specific multiple col values)

```
//SQL -- select projId,ProjName from Project where location='hyd'
        //create CriteriaBuiler object
CriteriaBuilder builder=ses.getCriteriaBuilder();
    //create CriteriaQuery object
CriteriaQuery<Object[]> ctQuery=builder.createQuery(Object[].class);
//Create Root obj specifying Entity class /table name from which records should be selected
Root<Project> root=ctQuery.from(Project.class); //from Project
//add muliple cols
ctQuery.multiselect(root.get("projId"),root.get("projName")).  // select projId,ProjName
        where(builder.equal(root.get("location"),"hyd")); // where location='hyd'
//prepare Query object
Query<Object[]> query=ses.createQuery(ctQuery);
//execute logic
List<Object[]> list=query.getResultList();
//process the result
list.forEach(row->{
    for(Object val:row) {
        System.out.print(val+"  ");
    }
    System.out.println();
});
```

note:: we can not apply ".class" property on generics to get the object of java.lang.Class on generics.. So where ever it is required go for array.

```
Class c= List<Stirng>.class; //invalid
Class c=Stirng[].class; //valid
Class c1=List<Object[]>.class ; //invalid
Class c1=Object[].class // valid
```

//Executing Scalar JPA QBC logic (getting spefic single col values)
==========

```
//SQL -- select ProjName from Project where cost between(?,?) order by projName asc
        //create CriteriaBuiler object
CriteriaBuilder builder=ses.getCriteriaBuilder();
    //create CriteriaQuery object
CriteriaQuery<String> ctQuery=builder.createQuery(String.class);
//Create Root obj
Root<Project> root=ctQuery.from(Project.class); //from Project
//add select and where clauses
ctQuery.multiselect(root.get("projName")).  //select projName
        where(builder.between(root.get("cost"), 1000000, 2000000)). //where cost between(1000000,2000000)
        orderBy(builder.asc(root.get("projName"))); //order by projName asc

//prepare Query object
    Query<String> query=ses.createQuery(ctQuery);
    //execute logic
    List<String> list=query.getResultList();
    //process the result
    list.forEach(System.out::println);
}//try
catch(HibernateException he) {
    he.printStackTrace();
}
finally {
    HibernateUtil.closeSessionFactory();
}
```

Executing scalar JPA-QBC logic that deals with aggragate results

```
//SQL -- select count(*) from Project
        //create CriteriaBuiler object
CriteriaBuilder builder=ses.getCriteriaBuilder();
    //create CriteriaQuery object
CriteriaQuery<Long> ctQuery=builder.createQuery(Long.class);
//Create Root obj
Root<Project> root=ctQuery.from(Project.class); //from Project
//add select and where clauses
    ctQuery.multiselect(builder.count(root.get("projId")));
//prepare Query object
    Query<Long> query=ses.createQuery(ctQuery);
    //execute logic
    long count= query.uniqueResult();
    //process the result
    System.out.println("records count "+count);
```

note ::
Here also we can use setFirstResult(-) and setMaxResults(-) on Query object to work with pagination concept

---

=> HB-QBC does not support non-select Operations where JPA-QBC supports non-select Operaitons (update,delete) using CriteriaUpdate , CriteriaDelete objects.

Why CriteriaInsert object is not given?
Ans) QBC logics can not deal with generators . So use ses.save(-) to save object/ to insert record by involving generators..

```
//========== Executing Non-Select Update QBC Logic ==================
Transaction tx=null;
 int count=0;
 boolean flag=false;
 try {
if(!ses.getTransaction().isActive())
tx=ses.beginTransaction();
 //create CriteriaBuilder obj
 CriteriaBuilder ctBuilder=ses.getCriteriaBuilder();
 // update Project set teamSize=?,location=? where cost>=?
 //create CriteriaUpdate object
 CriteriaUpdate<Project> ctUpdate=ctBuilder.createCriteriaUpdate(Project.class);
 //create Root object
 Root<Project> root=ctUpdate.from(Project.class); // update Project
 //perform set operation and where clause
 ctUpdate.set(root.get("teamSize"),20). //set teamSize=30
          set(root.get("location"),"colombo"). // ,location='hyd'
              where(ctBuilder.ge(root.get("cost"),1000000)); //where cost>=1000000
  //prepare Query object
  Query<Integer> query=ses.createQuery(ctUpdate);
     count=query.executeUpdate();
     flag=true;
  }
  catch(HibernateException he) {
     he.printStackTrace();
  }
  finally {
     if(flag) {
          tx.commit();
          System.out.println("no.of records that are effected::"+count);
     }
     else {
          tx.rollback();
          System.out.println("No records are found for update");
     }
      //close SessionFactory
     HibernateUtil.closeSessionFactory();
  }//finally
```

```
//========== Executing Non-Select DELETE QBC Logic ==================
Transaction tx=null;
 int count=0;
 boolean flag=false;
 try {
if(!ses.getTransaction().isActive())
tx=ses.beginTransaction();

 //create CriteriaBuilder obj
 CriteriaBuilder ctBuilder=ses.getCriteriaBuilder();
 //SQL ::delete from Project where cost between(100000,1000000)
 //create CriteriaUpdate object
 CriteriaDelete<Project> ctDelete=ctBuilder.createCriteriaDelete(Project.class);
 //create Root object
 Root<Project> root=ctDelete.from(Project.class); // delete from Project
 ctDelete.where(ctBuilder.between(root.get("cost"),100000,1000000)); // where cost between(100000,10000000)
  //prepare Query object
  Query<Integer> query=ses.createQuery(ctDelete);
count=query.executeUpdate();
     flag=true;
 }
 catch(HibernateException he) {
     he.printStackTrace();
 }
 finally {
     if(flag) {
          tx.commit();
          System.out.println("no.of records that are deleted::"+count);
     }
     else {
          tx.rollback();
          System.out.println("No records are found for deletion");
     }
      //close SessionFactory
     HibernateUtil.closeSessionFactory();
 }//finally
```

Hibernate Reverse Engineering tool

=> This tool is capable generating HB entity classes , mapping files, cfg file and etc.. dynamically based on the given DB table

=> For this we need to install hibernate tool given by jboss as plugin to the eclipse

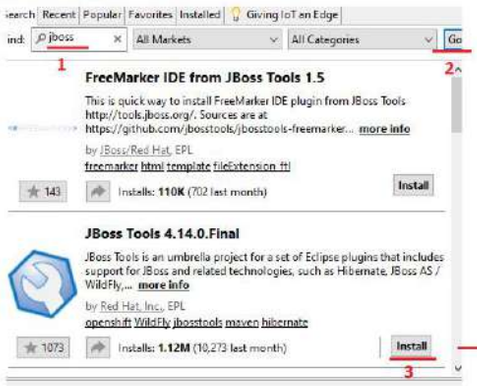
=> Hibernate generating DB tables is called HB Engineering

=> Generating HB code,files from DB tables is called HB reverse engineering

Procedure

step1) Install "HibernateTool" plugin in eclipse IDE

help menu --->eclipse market place --->



Eclipse Market place ==> To install third party plugin to eclipse (eg:: STS, Jboss Tool and etc..)

Install new Software ==> To install eclipse supplied plugins to eclipse (eg:: windows builder)

select Hibernate tools (check box)

Accept terms and conditions

install anyway

restart IDE

step2) Create Maven/Gradle Standalone Project and create basic packages in src/main/java folder ..

|---> com.nt.cfgs

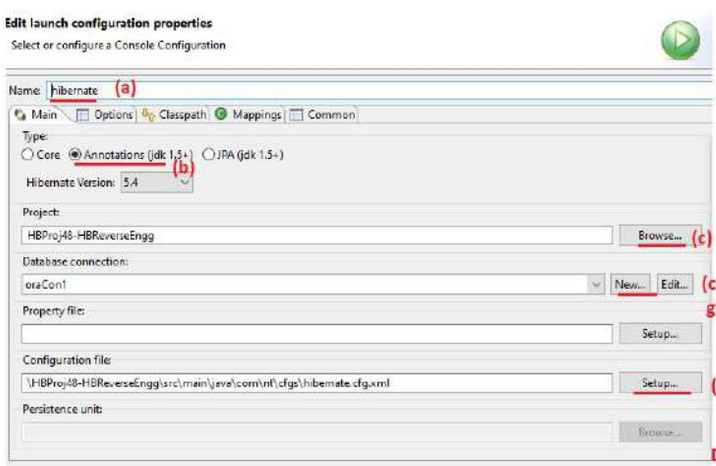
|---> com.nt.entity

|--> com.nt.test

|--> com.nt.utility

step3) create Hibernate configuration by using HB cfg wizard..

window menu--->open perspective --->others ---> Hibernate ---> Hibernate cfg window -->



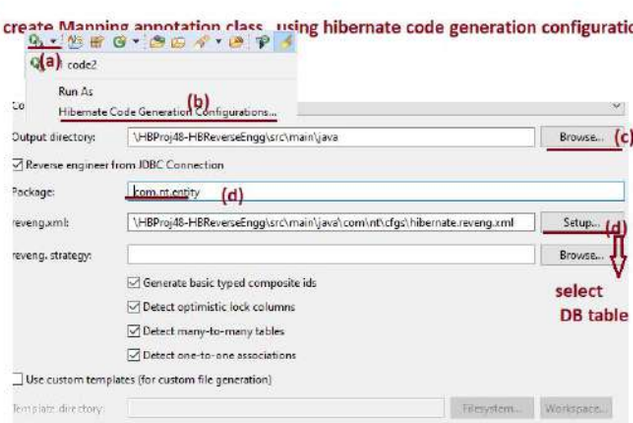
(c) step2 project

(create new con by giving jdbc properties, adding ojdbc6/7/8.jar file)

(create new file in com/nt/cfgs folder)

open sessionfactory tab of hibernate.cfg.xml file and more properties.. to cfg file..

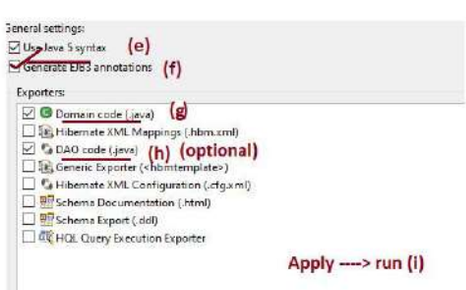
step4) create Mapping annotation class using hibernate code generation configuration wizard



select DB table

for videos

https://www.youtube.com/c/NareshIT/search?query=hibernate%20reverse%20engg



Apply ----> run (i)

It gives one Entity class Project.java (becoz Project DB table is selected) in com.nt.entity pakage (addtional ProjectHome.java ---Delete this file)

step5) copy HibernateUtil.java from old project

note:: add this line to hibernate.cfg.xml file

```
<mapping class="com.nt.entity.Project"/>
```

step6) develop client App and run the Application..

HB filters
=========

=>HB filter is a named ,parameterized global condition that can be enabled or disabled on HB Session obj.
This concept is very useful to execute SQL Queries with implicit conditions..
usecases:: The blocked, closed accounts should not participate in any queries execution..

note:: In real pratice the blocked, closed account will be not deleted from DB table .. but their status will be marked as closed or blocked..(This is called soft deletion)

Procedure work with Hibernate Filters
==========

step1) Define filter having name and parameters..

In mapping file (after </class> tag)
==========

```
<filter-def name="FILTER_BANKACCOUNT_STATUS">
   <filter-param name="accountType1" type="string"/>
   <filter-param name="accountType2" type="string"/>
  </filter-def>
```

step2) write filter conditon for Entity class to make it as implicit condition

In Mapping file (before </class> tag)

```
<!-- prepare filter conditon -->
   <filter name="FILTER_BANKACCOUNT_STATUS" condition="STATUS NOT IN(:accountType1,:accountType2)"/>
```

SQL Conidtion

step3) enable filter on Session object and set values filter parameters.. before executing any queries/ persistence logic either in DAO class or in Client App.

```
//enable filter
Filter filter1=ses.enableFilter("FILTER_BANKACCOUNT_STATUS");
//set filter param values
filter1.setParameter("accountType1","blocked");
filter1.setParameter("accountType2","closed");
//parepare and execute HQL
Query query=ses.createQuery("FROM BankAccount WHERE balance>=:amt");
query.setParameter("amt", 20000f);
List<BankAccount> list=query.getResultList();
list.forEach(System.out::println);

//disable filter
ses.disableFilter("FILTER_BANKACCOUNT_STATUS");
//parepare and execute HQL
Query query1=ses.createQuery("FROM BankAccount WHERE balance>=:amt");
query1.setParameter("amt", 20000f);
List<BankAccount> list1=query1.getResultList();
list1.forEach(System.out::println);
```

executes Query with filter condition

Executes the Query with out filter condition

=>HQL,HB-QBC, JPA-QBC supports Hibernate filters where as NativeSQL does not support HB filters becoz NativeSQL queries go to DB s/w as it is with out applying much convertions

---

Annotation driven HB filters
==========

@FilterDefs, @FlterDef -->To define HB filter
@Filters, @Flter -->To prepare HB filter condition
@FilterDefs,@Filters are container annotaion to hold @FilterDef,@Filter annotation for multiple times.. but not required from Hibernate 4.x becoz @Filter,@FilterDef are given as Repeatable annotation (java8 feature)
@ParamDef -->To define filter parameter

example
========

```
@Entity
@Table(name="BANK_ACCOUNT")
@FilterDef(name = "FILTER_BANKACCOUNT_STATUS",
           parameters = { @ParamDef(name="accountType1",type = "string"),
                          @ParamDef(name="accountType2",type = "string")
                        }
          )
@Filter(name ="FILTER_BANKACCOUNT_STATUS",
        condition = "STATUS NOT IN(:accountType1,:accountType2)")
public class BankAccount implements Serializable {
  ...
  ...
  ...
}
```

To define a filter

To prepare filter condition

Soft deletion in Hibernate

=> It is not deleting record from DB table physically.. It is all about marking the record from DB table as deleted.

=> When we close the account in BAnk.. they do not actually delete the record the record.. in fact they mark the record as deleted/ closed.

=>For soft deletion , we should not use "delete SQL query" though ses.delete(-) method is called.. we should use "update SQL " query i.e we should cfg custom SQL query for ses.delete(-) method call.. To cfg custom queries for hibernate persistence method calls like ses.save(-), ses.update(-) and ses.delete(-) and etc.. we use can <sql-xxx> tags or @SQLXxx annotations.

tags--> <sql-insert> ,<sql-update>,<sql-delete>
annotations --> @SQLInsert ,@SQLUpdate, @SQLDelete

(Allows to cfg our choice SQL queries by ignoring the SQL Dialect generated SQL queries)

note:: There is no <sql-select> tag and @SQLSelect Annotations.. i.e there is no need customizing Load object operation related select SQL queries...

Example Application on softDeletion (Using xml cfgs)

step1) Design DB table having one flag column like "status" to maintain active or inactive state of the record..

```
SQL> select * from bank_account;

    ACNO HOLDERNAME          BALANCE  STATUS
---------- --------------------  ----------  --------------------
    1001   rajesh                40000      active
    1002   suresh                50000      blocked
    3456   naresh                30000      active
     789   phani                 40000      closed
```

step2) Develop Hiberante App in reqular mannder having BankAccount as the Entity class..

step3) place <sql-delete> inside <class> tag of mapping file having "update SQL Query" to perform softdeletion..

```
<sql-delete>UPDATE BANK_ACCOUNT SET STATUS='closed' WHERE ACNO=?</sql-delete>
```

step4) Write Client App calling ses.delete(-) to delete the Record.. But it will execute the above cfg update query ..

```
Transaction tx=null;
        boolean flag=false;
        try {
            if(!ses.getTransaction().isActive())
                tx=ses.beginTransaction();
            //delete object
            BankAccount account=new BankAccount();
            account.setAcno(1001);
            ses.delete(account);
            flag=true;
        }
        catch(HibernateException he) {
            flag=false;
        }
        finally {
            if(flag) {
                tx.commit();
                System.out.println("Account closed (soft deletion)");
            }
            else {
                tx.rollback();
                System.out.println("Account not closed");
            }

            HibernateUtil.closeSessionFactory();
        }
```

=>In order apply implicit condition on every SELECT SQL query specific one entity class then place "where" attribute in the <class> of hibernate mapping file..

```
<class name="com.nt.entity.BankAccount" table="BANK_ACCOUNT"
                where="STATUS NOT IN('closed','blocked')">
  ....
  ....
</class>
```

The above condition is implicit condtion for select SQL Queries specific to Current entity class. where as the Hiberante Filter conditions global, parameterized conditions having ability to enable or disable..

note:: <sql-XXX> or @SQLXxx annotations related custom queries execution takes place only for single row method calls like ses.save(-) or ses.update(-) or ses.delete(-) and etc.. i.e they will not execute for HQL, nativeSQL and JPA QBC logics.. So while working HQL,NativeSQL, JPA QBC env.. we need to write "update SQL query directly for soft deletion".

HQL based soft Deletion

```
Transaction tx=null;
        boolean flag=true;
        int count=0;
        try {
            if(!ses.getTransaction().isActive())
                tx=ses.beginTransaction();
            Query query=ses.createQuery("update BankAccount set status='closed' where acno=:no");
            query.setParameter("no", 4567);
            //execute query
            count=query.executeUpdate();
        }
        catch(HibernateException he) {
    he.printStackTrace();
    flag=false;
        }
        finally {
            if(flag) {
                tx.commit();
                System.out.println("record deleted (soft deletion)");
            }
            else {
                tx.rollback();
                System.out.println("record not deleted");
            }
            HibernateUtil.closeSessionFactory();
        }
```

Soft Deleteion in annotation driven HB programming

@SQLDelete :: To specify and execute Custom SQL query ses.delete() method
(i.e here we can specify update query that is required for the soft deletion)
alternate to <sql-delete> tag

@Where :: To specify implicit condition for select SQL Query as specific to single entity class
alternate to "where" attribute of <class> tag.

```
@Entity
@Table(name="BANK_ACCOUNT")
@SQLDelete(sql = "UPDATE BANK_ACCOUNT SET STATUS='closed' WHERE ACNO=?")
@Where(clause = "STATUS NOT IN('closed','blocked')")
public class BankAccount implements Serializable {
    @Id
    @GeneratedValue
    private int acno;
    private String holderName;
    private float balance;
    private String status;

     //getters && setters
     ....
     ....
     ...
}
```

(for soft deletion)

Advanced o-r mapping

==========

=>There are mulitple mismatches b/w Java and RDBMS s/w ... like we can keep entity classes in the composion or inheritence and we can not keep db tables in the inheritence or composition but we want to map the Entity classes of composition or inheritence with 1 or more DB table.. by taking the support of advanced o-r mapping..

The advanced o-r mappings are
- => component mapping (when entity classes are in the composition)
- => Inheritence mapping (when entity classes are in the inheritence)
- => Collection mapping (When entity classes are having collections with primitives)
- => Association Mapping (When entity classes are in relationship like
  [1 .. 1 , 1 ... M , M ... 1 , M ... M] (or)
  [1 ... 1 , 1 ... * , * ..... 1 , *... *]

Component Mapping

==========

=>The property that is pointing to single column in db table is called simple property.

=>The property that is pointing to multiple columns in db table having sub properties is called component property.

=> Sometimes we take classes in composition (HAS-A relation) to improve the modularity.

```
public class Student{
  private int sno;
  private String sname;
  private float avg;
  private String doorNo;
  private String streetName;
  private String areaName;
  private String state;
  private String country;
  private int pincode
  //setters && getters
  ....
}
```

```
public class Employee{
  private int eno;
  private String ename;
  private float salary;
  private String doorNo;
  private String streetName;
  private String areaName;
  private String state;
  private String country;
  private int pincode
  //setters && getters
  ....
}
```

```
public class Customer{
  private int cno;
  private String cname;
  private float billAmt;
  private String doorNo;
  private String streetName;
  private String areaName;
  private String state;
  private String country;
  private int pincode
  //setters && getters
  ....
}
```

(Bad designing)

---

using composition (HAS-A Realationship)

==========

```
public class Address{
  private String doorNo;
  private String streetName;
  private String areaName;
  private String state;
  private String country;
  private int pincode
  //setters && getters
  ... .....
}
```

```
public class Student{
  private int sno;
  private String sname;
  private float avg;
  private Address addrs; (HAS-A realtion)
  //setters ....
}
```

```
public class Employee{
  private int eno;
  private String ename;
  private float salary;
  private Address addrs; (HAS-A relation)
  //setters && getters
  ....
}
```

(Good Designing)

Student db table acccording to the above Student Entity class

---

Option1)

```
Student (db table)
  |--->sno (n) (pk)
  |--->sname (vc2)
  |--> avg (float)
  |--> Addrs location (UDT-Struct)

SQL> CREATE TYPE location AS OBJECT (
  doorNo VARCHAR2(10),
  streetName VARCHAR2(20),
  areaname varchar2(10),
  state varchar2(10),
  country varchar2(10),
  pincode number(10) );
```

[This model will not work in few DB s/ws.. i.e no portability]

option2

```
Student (db table)
  |-->sno (n) (pk)
  |-->sname (vc2)
  |-->avg (float)
  |-->doorNo (vc2)
  |--> streetName (vc2)
  |-->areaName (vc2)
  |-->state (vc2)
  |-->country (vc2)
  |-->pincode n(10)
```

[This model will work in all DB s/ws.. i.e there is DB portability]

What is Problem of granuality?

Ans) if we design classes having composition and Db table with out User defined Data types i.e designing db table with more columns.. make the programmers to map more java classes (more granuals) with single DB table.. but basic o-r mapping does not support that process..So it is called problem of granuality. To address this use component mapping.. which gives addtional instructions to Hibernate to map HAS-A relation property(componet property) with multiple cols of DB table

Entity classes with composition (HAS-A Relationship)

```
public class Address{
  private String doorNo;
  private String streetName;
  private String areaName;
  private String state;
  private String country;
  private int pincode
  //setters && getters
  ... .....
}
```

```
public class Student{
  private int sno;
  private String sname;
  private float avg;
  private Address addrs; (HAS-A realtion)
  //setters ....
}
```

(Good Designing)

Student (db table)

sno (pk)(n)	sname	avg	doorNo	streetName	areaName	state	country	pinCode
---	---	---	---	---	---	---	---	---

=>In xml cfgs use <component> tag to map component property with mutiple cols

=>In Annotations cfgs use @Embedded,@Embeddable for the same.

In Student.hbm.xml

==========

```
<hibernate-mapping>
  <class name="pkg.Student" table="STUDENT">
    <id name="sno"/>
    <p name="sname"/>
    <p name="avg">
    <component name="address" class="pkg.Address">
      <property name="doorNo"/>
      <p name="streetName"/>
      <p name="areaName"/>
      <p name="state"/>
      <p name="country"/>
      <p name="pincode"/>
    </component>
    </class>
  </hibernate-mapping>
```

Component Mapping

Code in Client App

==========

```
Session ses=null;
        Address addrs1=null, addrs2=null;
        StudentInfo info1=null,info2=null;
        //get Session
        ses=HibernateUtil.getSession();
        //prepare objects
        addrs1=new Address();
        addrs1.setDoorNo("10-20-3/A");
        addrs1.setStreetName("RK Street");
        addrs1.setAreaName("Ameerpet");
        addrs1.setState("Telangana");
        addrs1.setCountry("India");
        addrs1.setPinCode(566777L);
        info1=new StudentInfo();
        info1.setSname("rajesh"); info1.setAvg(89.55f);
        info1.setAddrs(addrs1); //HAS-A relationship

        addrs2=new Address();
        addrs2.setDoorNo("11-20-3/A");
        addrs2.setStreetName("DK Street");
        addrs2.setAreaName("Ameerpet");
        addrs2.setState("Telangana");
        addrs2.setCountry("India");
        addrs2.setPinCode(562777L);
        info2=new StudentInfo();
        info2.setSname("raj"); info2.setAvg(89.55f);
        info2.setAddrs(addrs2); //HAS-A relationship

        Transaction tx=null;
        boolean flag=false;
        try {
            //begin Tx
            if(!ses.getTransaction().isActive())
                tx=ses.beginTransaction();
            ///save objs
            ses.save(info1);
            ses.save(info2);
        flag=true;
        }//try
        catch(HibernateException he) {
            flag=false;
            he.printStackTrace();
        }
        finally {
            if(flag) {
                tx.commit();
                System.out.println("object are saved");
            }
            else {
                tx.rollback();
                System.out.println("Object is not saved");
            }
            HibernateUtil.closeSessionFactory();
        }//finally
```

Select Operation in ComponentMapping
==========

```
/// Load objects
Query<StudentInfo> query=ses.createQuery("from com.nt.entity.StudentInfo where addrs.streetName=:street ");
query.setParameter("street","RK Street");
List<StudentInfo> list=query.getResultList();
list.forEach(System.out::println);
```

Annotation driven Component Mapping
==========

=>Use @Embeddable on the top of helper class (HAS-A Relationship class like Address class) to indicate that it is not a Entity class.. It is a helper class / attachable class to Entity class.
=>Use @Embedded the mark the property as Component property..

In StudentInfo class

```
@Embedded
private Address addrs;
```

Example
=========

```
@Data
@Entity
@Table(name="STUDENT_INFO")
public class StudentInfo implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Type(type="int")
    private int sno;
    @Type(type="string")
    @Column(length=20)
    private String sname;
    @Type(type="float")
    @Column(scale = 10,precision = 2)
    private float avg;
    @Embedded  // To make the property as component property
    private Address addrs;

}
```

Address.java

```
@Data
@Embeddable
public class Address {
    @Type(type="string")
    @Column(length=20)
    private String doorNo;
    @Type(type="string")
    @Column(length=20)
    private String streetName;
    @Type(type="string")
    @Column(length=20)
    private String state;
    @Type(type="string")
    @Column(length=20)
    private String areaName;
    @Type(type="string")
    @Column(length=20)
    private String country;
    @Type(type="long")
    @Column(length=10)
    private long pinCode;

}
```

Inheritence Mapping

=>we can keep the entity classes of hibernate in the Inheritence becoz they are java classes but we can not keep db tables in the inheritence..i.e we want map the classes of inheritence with one or more Db tables which is not possible while working basic o-r mapping.. This is called sub types problem. To overcome this problem we need to give special instructions to Hibernate to map the classes of inheritence with one or more Db tables..using advanced o-r mapping called "Inheritence Mapping".

=>if One class wants to use entire properties and methods of another class then go for Inheritence. (IS-A Relation)

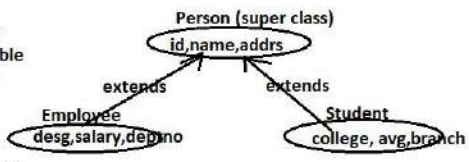
=>if One class wants to use few properties and few methods of another class then go for composition. (HAS-A relation)

4 types of Inheritence Mapping /4 Inheritence Mapping strategies
==========

(a) Table Per Class Hierarchy [All entity classes of inheritence hierarchy will be mapped with single db table]
(best) (b) Table per sub class [each sub class will be mapped with seperate db table having 1 .. 1 relation with super class db table]
(c) Table per concrete class [each sub class will be mapped with seperate db table having both common and specific cols]
(d) Table per concrete class with Implicit Polymorphism [each sub class will be mapped with seperate db table having both common and specific cols and classes will be cfg as normal classes with out highlighting their inheritence]

(a) Table Per Class Hierarchy

=> Here all the entity classes of inheritence hierarchy will be mapped with single DB table and this db table should contain huge no.of columns becoz it has to maintain cols for super and all sub classes related properties..



=> The Db table should have one extra column along with the above huge no.of columns indicating which record is inserted using which Entity class of inheritence hierarchy..

Db table
======

INH_PERSON
==========

Discriminator column

id (pk)	name	addrs	desg	salary	deptno	college	avg	branch	person_type
101	raja	hyd							per
102	ramesh	hyd	clerk	9000	1001				emp
103	mahesh	hyd	actor	8000	1001				emp
104	suresh	vizag				CBIET	90.55	EC	stud

=>In this strategy descriminator column cfg and discriminator value cfg for each domain/entity class is mandatory..

=> In this strategy use <class> tag to cfg super class and <subclass>(as sub tags of <class> tag) tags to cfg sub classes and <discriminator> to specify and cfg discriminator column

Person.hbm.xml
==========

```
<hinbernate-mapping  package="com.nt.entity">
  <class name="Person"  table="INH_PERSON"  discriminator-value="per">
    <id name="id">
        <generator class="increment"/>
    </id>
    <discriminator  column="PERSON_TYPE"  type="string" length="10"/>
    <property name="name"/>
    <property name="addrs"/>
    <subclass  name="Employee"   discriminator-value="emp">
      <property name="desg"/>
      <property name="salary"/>
      <property name="deptNo"/>
    </subclass>
    <subclass  name="Student"   discriminator-value="stud">
      <property name="college"/>
      <property name="branch"/>
      <property name="avg"/>
    </subclass>
  </class>
</hibernate-mapping>
```

becomes discriminator value when record is inserted using Person entity class

becomes discriminator value when record is inserted using Employee entity class

becomes discriminator value when record is inserted using Student entity class

=>descriminator column is direct column and it will not mapped with any Entity class property..

Limitations
==========to

a) We need design Db table with huge no.of cols which is a bad practice.
b) The Columns mapped with the properties of sub classes can not be taken as "non-null" cols.
c) To work with discriminator column.. we must use NativeSQL as this column is not mapped with any property of entity class..
d) Becoz lengthy Db table , it is not industry standard..
e) adding new sub classes makes to add more colums to existing db table.. which quite issue in production env...

note:: In this Strategy Discriminator column cfg is mandatory.. If not cfg then we get
org.hibernate.MappingException: No discriminator found for com.nt.entity.Student. Discriminator is needed when 'single-table-per-hierarchy' is used and a class has subclasses

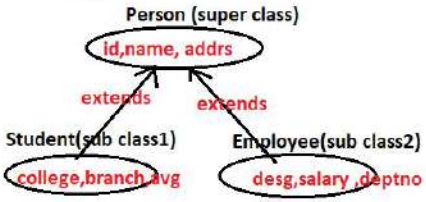
note:: In this strategy , if not discriminator-value is not configured in <class> , <subclass> tags then the fully qualified Entity class name will be take as the default discriminator value.

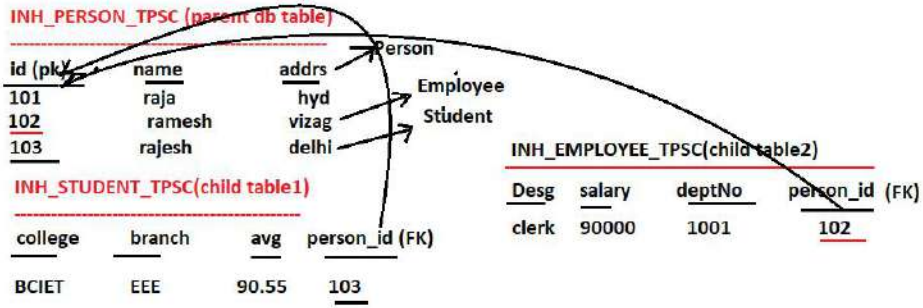
note: while working any strategy of inheritence mapping , if we try to fetch the records by using super Entity class of the Inheritence hierarchy then not we get records the inserted by using Super class object.. we also get records even inserted by using sub class objects.. becoz super class ref variable can refer its own class objects and its sub class objects..

Table per sub class
==========

=>Here every super class having common properties will be mapped with parent DB table having common cols and sub classes will be mapped with child db table having specific cols.. i.e the db tables of sub classes should have 1 to 1 relationship with parent db table of super class..

=>Since every class of injeritence hierarchy is having seperate db table.. so there is no need of taking disciminator column here..





=>In Mapping file use <class> to cfg super class and use <joined-subclass> tags to cfg sub classes.

In Person.hbm.xml
==========

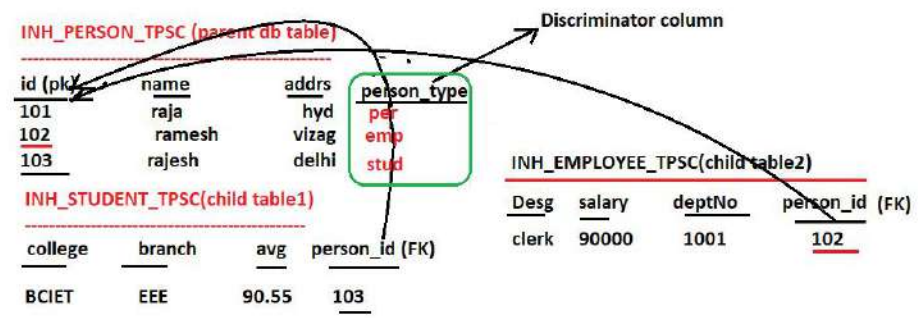
```
<hibernate-mapping package="com.nt.entity">
   <class name="Person"  table="INH_PERSON_TPSC">
       <id name="id"  type="int">
           <generator class="Inrement"/>
       </id>
       <property name="sname" length="10" type="string"/>
       <property name="sadd" length="10" type="string"/>

    <joined-subclass  name="Employee"   table="INH_EMPLOYEE_TPSC">
        <key  column="person_id"/>  <!--FK column name -->
        <property name="desg"  type="string" length="10"/>
        <property name="salary"  type="float"  scale="2" precision="10"/>
        <property name="deptNo"  type="long"/>
    </joined-subclass>
    <joined-subclass name="Student"  table="INH_STUDNET_TPSC">
        <key column="person_id"/>
        <proerty name="college" type="string" length="10"/>
        <proerty name="branch" type="string" length="10"/>
        <proerty name="avg" type="float" precision="5"  scale="2"/>
    </joined-subclass>
  </class>
</hibernate-mapping>
```

Advantates of this mapping
==========

(a) No need of taking db table having huge no.of colums .. becoz we are taking multiple db tables in relationship
(b) Taking Discriminator column is optional
(c) The columns of db tables mapped by sub classes can have not-null constraint
(d) It is industry standard..

Table per sub class with discrminator column support
==========

=>In this strategy the parent contains common col values which comes becoz of any Entity class object based save operation .. we can add the optional discriminator column in the parent db table indicating which record is inserted using which entity class of inheritence hierarchy



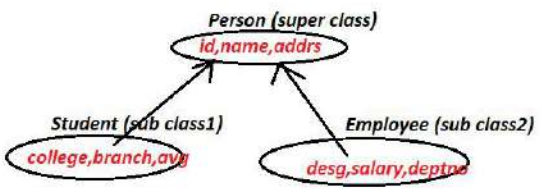
if we mixup table per class hierarchy and table per sub class inheritence mapping strategies then we can add discriminator column to the parent db table..

mapping file
==========

```
<hibernate-mapping package="com.nt.entity">
  <class name="Person" table="INH_PERSON_TPSC_DISC" discriminator-value="per" >
    <id name="id" type="int">
      <generator class="increment"/>
    </id>
    <discriminator column="PERSON_TYPE" length="10" type="string"/>
    <property name="name" type="string" length="15"/>
    <property name="addrs" type="string" length="15"/>

    <subclass name="Student" discriminator-value="stud">
     <join table="INH_STUDENT_TPSC_DISC">
      <key column="PERSON_ID"/>
       <property name="college" type="string" length="10"/>
       <property name="branch" type="string" length="10"/>
       <property name="avg" type="float" scale="2" precision="5"/>
      </join>
    </subclass>

    <subclass name="Employee" discriminator-value="emp" >
     <join table="INH_EMPLOYEE_TPSC_DISC">
      <key column="PERSON_ID"/>
       <property name="desg" type="string" length="10"/>
       <property name="salary" type="float" scale="2" precision="10"/>
       <property name="deptNo" type="int"/>
      </join>
    </subclass>
  </class>
</hibernate-mapping>
```

Table per concrete class
==========

=> In this strategy Every class inheritence hierarchy will have its own Db table.. but the db tables of sub classes will be having cols representing both common columns and specific columns i.e commons cols will be repeated across the multiple db tables like in super class class mapped db table and in the db tables mapped by sub classes..



INH_PERSON_TPCC (table1)
========== for super class

id(pk)	name	addrs
101	raja	hyd

INH_STUDENT_TPCC (table2 for sub class1)
==========

id(pk)	name	addrs	college	branch	avg
102	ramesh	hyd	CBIET	EEE	90.56

INH_EMPLOYEE_TPCC(table2 for subclass2)
==========

id(pk)	name	addrs	desg	salary	depno
103	suresh	hyd	CLERK	9000	1001

repeating same cols across the multiple db tables is aganist DB normalization principles.. So it is not recomanded to use.

note:: Since every class of inheritence hierarchy is having its own db table that to having both common and specific cols.. So there is no of discrminator column in the DB table designing.

=> Here Db tables of the entity classes are independent db tables having no FK Relationship among them

=>cfg super class using <class> tag and cfg sub classes using <union-subclass> tags..

mapping file
==========

```
<hibernate-mapping package="com.nt.entity">
  <class name="Person" table="INH_PERSON_TPCC" >
    <id name="id" type="int">
      <generator class="increment"/>
    </id>
    <property name="name" type="string" length="15"/>
    <property name="addrs" type="string" length="15"/>

    <union-subclass name="Student" table="INH_STUDENT_TPCC" >
      <property name="college" type="string" length="10"/>
      <property name="branch" type="string" length="10"/>
      <property name="avg" type="float" scale="2" precision="5"/>
    </union-subclass>

    <union-subclass name="Employee" table="INH_EMPLOYEE_TPCC">
      <property name="desg" type="string" length="10"/>
      <property name="salary" type="float" scale="2" precision="10"/>
      <property name="deptNo" type="int"/>
    </union-subclass>
  </class>
</hibernate-mapping>
```

When "from PErson" query is executed.. the generated SQL query is

```
Hibernate:
  select
    person0_.id as id1_1_,
    person0_.name as name2_1_,
    person0_.addrs as addrs3_1_,
    person0_.college as college1_2_,
    person0_.branch as branch2_2_,
    person0_.avg as avg3_2_,
    person0_.desg as desg1_0_,
    person0_.salary as salary2_0_,
    person0_.deptNo as deptno3_0_,
    person0_.clazz_ as clazz_
from
  ( select
    id,
    name,
    addrs,
    null as college,
    null as branch,
    null as avg,
    null as desg,
    null as salary,
    null as deptNo,
    0 as clazz_
  from
    INH_PERSON_TPCC
  union
  all select
    id,
    name,
    addrs,
    college,
    branch,
    avg,
    null as desg,
    null as salary,
    null as deptNo,
    1 as clazz_
  from
    INH_STUDENT_TPCC
  union
  all select
    id,
    name,
    addrs,
    null as college,
    null as branch,
    null as avg,
    desg,
    salary,
    deptNo,
    2 as clazz_
  from
    INH_EMPLOYEE_TPCC
) person0_
```

Annotation driven Inheritence Mapping

=>Supports only 3 strategies of inheritence mapping they are

a) Table per class hierarchy (Single table)
b) Table per sub class (Joined Table)
c) Table per concrete class (Table per class)

Annotations requried
==========

=> @Inheritence(strategy=InheritenceType.SINGLE_TABLE) ->For Table per class hierarchy
=> @Inheritence(strategy=InheritenceType.JOINED) ->For Table per joined class
=> @Inheritence(strategy=InheritenceType.TABLE_PER_CLASS) ->For Table per concrete class

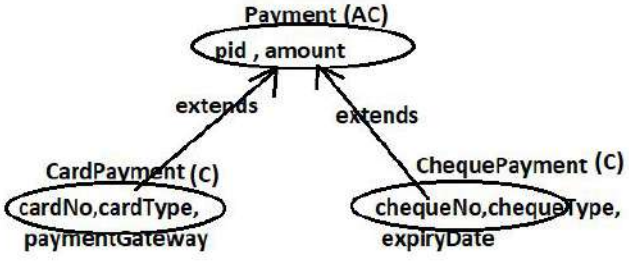
JPA annotation

note : default strategy is SINGLE_TABLE
(This annotation must be applied on the top of root class in the inheritence hierarchy)

@DiscriminatorColumn --->To cfg Discriminator column in Table per class hierarchy
@DiscriminatorValue --->To cfg Discriminator value in Table per class hierarchy
@PrimaryKeyJoinColumn --->To cfg FK column in Table per sub class.

JPA Annotations

Annotation driven Table per class Hierarchy (Single table)
==========



While working with inheritence mapping .. it is recomanded to take the root class of inheritence hierarchy as an abstract class becoz that class is there only to hold common properties ..not there to participate in Persistence operations..

PAYMENT_TPCH
==========

pid(pk)	amount	cardNo	cardType	paymentGateway	chequeNo	chequeType	expiryDate	(Discriminator column) payment_mode
1234	80000	456788	debit	visa	...	...	...	card
1235	86000	....	.....	....	564324	self	10-OCT-21	cheque

Paymemt.java
==========

```
@Data
@Inheritence(strategy=InheritenceType.SINGLE_TABLE)
@DiscriminatorColumn(name="payment_mode",type=DiscrminatorType.STRING,length=10)
@Table(name="PAYMENT_TPCH")
@Entity
public abstract class Payment{
  @Id @GeneratedValue
  private Integer pid;
  private Float amount;
}
```

CardPayment.java
==========

```
@Data
@Entity
@DiscriminatorValue("card")
public class CardPayment extends Payment{
  private Long cardNo;
  private String cardType;
  private String paymentGateway;

}
```

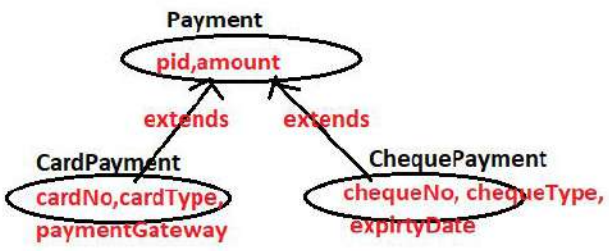
ChequePayment.java
==========

```
@Data
@Entity
@DiscriminatorValue("cheque")
public class ChequePayment extends Payment{
  private Long chequeNo;
  private String chequeType;
  private LocalDate expiraryDate;

}
```

if @Inheritence is not sepcified on the top root class or @Inheritence is specified but its Inheritence strategy is not specified then The table per class hierarchy will be taken as default Inheritence mapping strategy..

Annotation driven Inheritence Mapping (Table per sub class)

=> Here the super class db table and and the db tables of sub classes will maintain 1-1 realationship becoz the super class db table contains common columns



=>Here Discriminator column is optional.. but recomanded to take..

PAYMENT_TPSC (parent db table)

pid(pk)	amount	payment_mode (discriminator column)
123	5555	CARD
124	6666	CHEQUE

CARDPAYMENT_TPSC

cardNo	cardType	paymentgateWay	payment_id (fk)
45677	debit	visa	123

CHEQUEPAYMENT_TPSC

chequeNo	chequeType	expiryDate	payment_id(fk)
456333	self	20-OCT-21	124

Entity classes

Payment.java (super class)

```
@Entity
@Table(name="PAYMENT_TPSC")
@Inheritence(strategy=InheritenceType.JOINED)
@DiscriminatorColumn(name="payment_mode",
@Data                          length=10)
public class Payment implements Serializable{
 @Id @GeneratedValue
 private Integer pid;
 private Float amount;
}
```

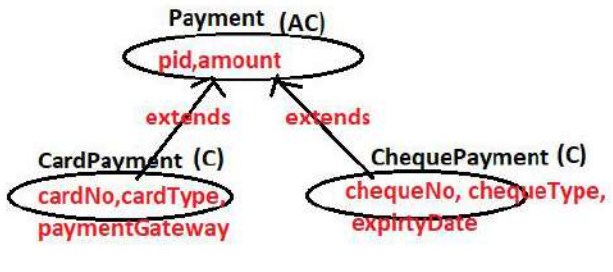
CardPayment.java (sub class1)

```
@Entity    @Data
@Table(name="CARDPAYMENT_TPSC")
@PrimaryKeyJoinColumn(name="payment_id", referencedColumnName="pid")
@DiscriminatorValue ("CARD")
public class CardPayment extends Payment{
 private Long cardNo;
 private String cardType;
 private String paymentGateway;
}
```

ChequePayment.java (sub class2)

```
@Entity
@Table(name="CHEQUEPAYMENT_TPSC")
@PrimaryKeyJoinColumn(name="payment_id", referencedColumnName="pid")
@Data
@DiscriminatorValue("CHEQUE")
public class ChequePayment extends Payment{
 private Long chequeNo;
 private String chequeType;
 private Local Date exiparyDate;

}
```

Annotation Driven Inheritence Mapping (Table Per concrete class)



note:: since we are taking super calss as abstract class and we are not going to do any persistence operations using super class ,So no db table is required for super class.

CARDPAYMENT_TPCC

pid(pk)	amount	cardNo	cardType	paymentGateway
123	5555	554546	debit	visa

CHEQUEPAYMENT_TPCC

pid(pk)	amount	chequeNo	chequeType	expiryDate
124	5667	45466	self	21-OCT-22

Payment.java (super class)

```
@Entity
@Data
@Inhitence(strategy=InheritenceType.TABLE_PER_CLASS)
public abtract class Payment implements Serializable{
  @Id
  @GenerdatedValue
  private Integer pid;
  private Float amount;
}
```

CardPayment.java (sub class1)

```
@Entity
@Table(name="CARDPAYMENT_TPCC")
@Data
public class CardPayment extends Payment{
 private Long cardNo;
 private String cardType;
 private String paymentGateway
 }
```

ChequePayment.java (sub class2)

```
@Entity
@Table(name="CHEQUEPAYMENT_TPCC")
@Data
public class ChequePayment extends Payment{
 private Long chequeNo;
 private String chequeype;
 private LocalDate expirtyDate;
 }
```

note:: since every sub class is having seperate DB table.. so there is no need of taking discriminator column here..

=>In Annotation driven Inheritence mapping .. there is no possibility to implement "Implicit Polymorphism or Explicit Polymorphism " based Table per Concreate class Inheritence mapping strategy..

What is the difference b/w table per concrete class and Table per concreate class with Implicit polymorphism?

Ans) In Table Per concrete class, the HQL query that is developed based on " super class name" generates single SQL Query with union option to get super class db table, sub classes db table columns.

```
select
   pid, amount
 from
   ( select  , pid, amount,chequeNo, chequeType, expiryDate)
   from    CHEQUEPAYMENT_TPCC
   union
   all (select   pid,  amount, cardNo, cardType, paymentGateway) from   CARDPAYMENT_TPCC ;
```

In Table Per concrete class with implicit Polymorphism, the HQL query that is developed based on " super class name" generates multiple SQL Queries to get records super class db table row and sub class db table rows...

```
SELECT * FROM PAYMENT_TPCC;
SELECT * FROM CARDPAYMENT_TPCC;
SELECT * FROM CHEQUEPAYMENT_TPCC;
```

note:: In Annotation driven Inheritence mapping , there is no provision to work with both "implicit " and "explicit" polymorphism... using default setup..

Collection Mapping
==========

=>It is all about taking collection type properties with Strings/wrappers in Entity class .
=> For Every Collection property one seperate child table will be created having FK pointing PK col of Entity class db table (parent table)
=>This child table for collection property (List/Set/Map) will have min 2 colums (Set) and max 3 columns(List/Map)

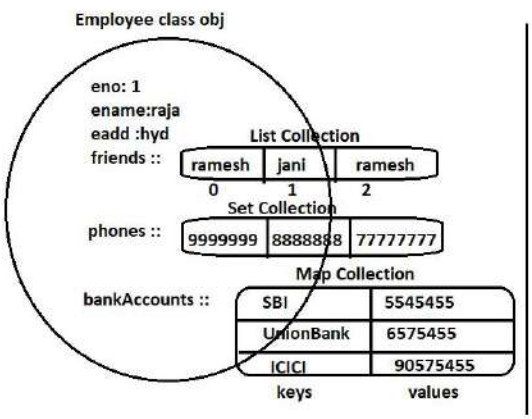
The Set Collection child table contains two columns a) element column b)foregin key column (No index column)
The List,Mapp Collection child tables contain 3 columns a) element column b)foregin key column and c) index column

(In List index column in numberic column, In Map index column can be of any type becoz key values of map collection elements act as index column values)

Collection are two types
a) Non-indexed colections :: Set collection
b) Indexed Collection :: List,Map

=>On every collection property we must add
=> @ElementCollection specifying element column name
=> @CollectionTable specifying child db table name and FK colum name
=> @OrderColumn (List) to specify index column name
@MapKeyColumn (Map) to specfify index column name

```
@Data
@Entity
@Table(name="EMPLOYEE_COLLECTION")
public class Employee implements Serializable{
  @Id
  @GeneratedValue
  private int eno;

  private String ename;
  private String eadd;
  @ElementCollection  @Column(name="friend")
  @CollectionTable(name="EMP_FRIENDS",
                   joinColumns=@JoinColumn(name="emp_id",referencedColumnName="eno")
  @OrderColumn(name="friend_no")  @ListIndexBase(value = 1)
  private List<String> friends;

  @ElementCollection  @Column(name="MOBILENUMBER")
  @CollectionTable(name="EMP_PHONES",
                   joinColumns=@JoinColumn(name="emp_id",referencedColumn="eno")
  private Set<Long> phones;

  @ElementCollection  @Column(name="BANK_ACCOUNTS")
  @CollectionTable(name="EMP_ACCOUNTS",
                   joinColumns=@JoinColumn(name="emp_id",referencedColumn="eno")
  @MapKeyColumn(name="BANK_NAME")
  private Map<String,Long> bankAccounts;

}
```



EMPLOYEE_COLLECTION (parent table)

eno(pk)	ename	eadd
1	raja	hyd

EMP_FRIENDS (Child table1-List Collection)

friend_no	friend	emp_id(fk)
1	ramesh	1
2	jani	1
3	ramesh	1

EMP_ACCOUNTS (Child table3-Map Collection)

bank_name	account_number	emp_id(fk)
SBI	5545455	1
UnionBank	6575455	1
ICICI	90575455	1

EMP_PHONES (child table2-Set Collection)

mobileNumber	emp_id (fk)
99999999	1
88888888	1
77777777	1

There the parent db table and child db tables are having one to many relationship (1 ... *)

=>The JPA Annotations required for the Collection mapping are @ElementCollection , @CollectionTable and etc..
=>In xml driven cfgs ,the following tags are required for collection mapping
<list>/<bag>,<set> ,<map> -->To cfg collection properties (alternate @CollectionTable)
<list-index> --> to cfg index column for List collection (altername to @OrderColumn,@ListIndexBase)
<key> -- To cfg FK column (alternate to @JoinColumn)
<element> --> To cfg element column (alternate to @ElementCollection)
<map-key> --> To cfg key of map collection as index column (alternate to @MapKeyColumn)

=>we can cfg java.util.List type Collection property by using either <list> or <bag> tag..
while configuring with <list> tag, the index column configuration is mandatory and while configuring with <bag> tag the index column cfg is not required.

Entity class
==========

```
@Data
public class Employee implements Serializable {
        private Integer eno;
        private String ename;
        private String eadd;

         private  List<String> friendsList;
         private List<String> relativesList;
         private Set<Long>phones;
         private  Map<String,Long> bankAccounts;
}
```

Employee.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping  package="com.nt.entity">
  <class name="Employee"  table="EMPLOYEE_COLLECTION">
     <id name="eno" type="int">
        <generator class="increment"/>
     </id>
     <property name="ename"  type="string" length="15"/>
     <property name="eadd" type="string" length="15"/>
     <list name="friendsList"  table="EMP_FRIENDS">
       <key column="EMP_ID"  />
       <list-index column="FRIEND_NO" base="1"  />
       <element column="FRIEND"  type="string" length="10"/>
     </list>
     <bag  name="relativesList"  table="EMP_RELATIVES">
       <key column="EMP_ID"/>
       <element column="RELATIVE" type="string" length="10"/>
     </bag>
     <set name="phones"  table="EMP_PHONES">
       <key column="EMP_ID"/>
       <element  column="MOBILENUMBER" type="long"  />
     </set>
     <map name="bankAccounts"  table="EMP_ACCOUNTS">
       <key column="EMP_ID"/>
       <map-key  column="BANK_NAME" type="string" length="10"/>
       <element column="BANK_ACCOUNT"  type="long" />
     </map>

  </class>
</hibernate-mapping>
```

Q) How to bring the effect of <bag> tag while working with annotation driven collctionmapping

Ans) write all annotations except @OrderColumn annotation.

```
@ElementCollection
        @Column(name="RELATIVE")
        @CollectionTable(name = "EMP_RELATIVES",
                         joinColumns = @JoinColumn(name="emp_id",referencedColumnName = "eno"))
        private  List<String> relativesList;
```

Working with Large Objects (files)
==========

=>images, audio files ,video files and etc... are called Binary large objects (BLOB) becoz they internally manage data as binary information (0,1).
=>text files,rich text files and etc... are called Character large objects (CLOB) becoz they internally manage data as character information (ASCII Or UniCode characters).

usecases:: while developing matrimony Apps, job portal Apps, profile mgmt apps as standalone Apps then store large objects (files) directly into DB table cols.

usecases:: while developing matrimony Apps, job portal Apps, profile mgmt apps as web Apps then store large objects (files) in server machine file system and add their locations/paths to the DB table as string col values.

BLOB=byte[]+ LOB(file)
CLOB =char[]+LOB (file)

add @Lob annotation on the top of Entiy class byte[] or char[] type bean properties.
eg::

```
@Lob
private  byte[]  photo;

@Lob
private  char[]  resume;
```

Sample Entity class
==========

```
@Entity
@Data
public class JobSeeker implements Serializable {
        @Id
        @GeneratedValue
        @Type(type="int")
        private Integer jsId;
        @Column(length=10)
        @Type(type="string")
        private String jsName;
        @Column(length=10)
        @Type(type="string")
        private String jsAddrs;

        //LOB property
        //@Type(type="binary")
        @Lob
        private  byte[] photo;


        //LOB property
        //@Type(type="character")
        @Lob
        private  char[] resume;
}
```

=>The JPA Annotations required for the Collection mapping are @ElementCollection , @CollectionTable and etc..
=>In xml driven cfgs ,the following tags are required for collection mapping
<list>/<bag>,<set> ,<map> -->To cfg collection properties (alternate @CollectionTable)
<list-index> --> to cfg index column for List collection (altername to @OrderColumn,@ListIndexBase)
<key> -- To cfg FK column (alternate to @JoinColumn)
<element> --> To cfg element column (alternate to @ElementCollection)
<map-key> --> To cfg key of map collection as index column (alternate to @MapKeyColumn)

=>we can cfg java.util.List type Collection property by using either <list> or <bag> tag..
while configuring with <list> tag, the index column configuration is mandatory and while configuring with <bag> tag the index column cfg is not required.

Entity class
==========

```
@Data
public class Employee implements Serializable {
    private Integer eno;
    private String ename;
    private String eadd;

    private List<String> friendsList;
    private List<String> relativesList;
    private Set<Long>phones;
    private Map<String,Long> bankAccounts;
}
```

Employee.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.nt.entity">
  <class name="Employee" table="EMPLOYEE_COLLECTION">
    <id name="eno" type="int">
      <generator class="increment"/>
    </id>
    <property name="ename" type="string" length="15"/>
    <property name="eadd" type="string" length="15"/>
    <list name="friendsList" table="EMP_FRIENDS">
      <key column="EMP_ID" />
      <list-index column="FRIEND_NO" base="1" />
      <element column="FRIEND" type="string" length="10"/>
    </list>
    <bag name="relativesList" table="EMP_RELATIVES">
      <key column="EMP_ID"/>
      <element column="RELATIVE" type="string" length="10"/>
    </bag>
    <set name="phones" table="EMP_PHONES">
      <key column="EMP_ID"/>
      <element column="MOBILENUMBER" type="long" />
    </set>
    <map name="bankAccounts" table="EMP_ACCOUNTS">
      <key column="EMP_ID"/>
      <map-key column="BANK_NAME" type="string" length="10"/>
      <element column="BANK_ACCOUNT" type="long" />
    </map>

  </class>
</hibernate-mapping>
```

Q) How to bring the efflect of <bag> tag while working with annotation driven collctionmapping

Ans) write all annotations except @OrderColumn annotation.

```
@ElementCollection
    @Column(name="RELATIVE")
    @CollectionTable(name = "EMP_RELATIVES",
                     joinColumns = @JoinColumn(name="emp_id",referencedColumnName = "eno"))
    private List<String> relativesList;
```

Working with Large Objects (files)
==========

=>images, audio files ,video files and etc... are called Binary large objects (BLOB) becoz they internally manage data as binary information (0,1).
=>text files,rich text files and etc... are called Character large objects (CLOB) becoz they internally manage data as character information (ASCII Or UniCode characters).

usecases:: while developing matrimony Apps, job portal Apps, profile mgmt apps as standalone Apps then store large objects (files) directly into DB table cols.

usecases:: while developing matrimony Apps, job portal Apps, profile mgmt apps as web Apps then store large objects (files) in server machine file system and add their locations/paths to the DB table as string col values.

BLOB=byte[]+ LOB(file)
CLOB =char[]+LOB (file)

add @Lob annotation on the top of Entiy class byte[] or char[] type bean properties.
eg::

```
@Lob
private byte[] photo;

@Lob
private char[] resume;
```

Sample Entity class (annotation driven cfgs)
==========

```
@Entity
@Data
public class JobSeeker implements Serializable {
    @Id
    @GeneratedValue
    @Type(type="int")
    private Integer jsId;
    @Column(length=10)
    @Type(type="string")
    private String jsName;
    @Column(length=10)
    @Type(type="string")
    private String jsAddrs;

    //LOB property
    //@Type(type="binary")
    @Lob
    private byte[] photo;


    //LOB property
    //@Type(type="characters")
    @Lob
    private char[] resume;
}
```

//Sample HBM file for LOB cfgs (xml driven cfgs)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.nt.entity">
  <class name="JobSeeker" table="JOBSEEKER">
    <id name="jsId" type="int">
      <generator class="increment"/>
    </id>
    <property name="jsName" type="string" length="15"/>
    <property name="jsAddrs" type="string" length="15"/>
    <property name="photo" type="binary"/>
    <property name="resume" type="characters"/>

  </class>
</hibernate-mapping>
```

Entity class
==========

```
@Data
public class JobSeeker implements Serializable {
    private Integer jsId;
    private String jsName;
    private String jsAddrs;
    private byte[] photo;
    private char[] resume;
}
```

=>if the Entity class property type is boolean property .. it can have either getXxx() /isXxx() ... and DB table colum value will be stored as 0(false)/1 (true),

InEntity class

```
private boolean active;
```

⇩

```
public void setActive(boolean active){
  this.active=active;
}

public boolean isActive(){   (or)   public boolean getActive(){
  return active;                    return active;
}                                   }
```

AssociationMapping in hibernate (Relationships in Hibernate) (Multiplicity in Hibernate)

=>If we keep db tables in relationship .. their relevant Entity classes should kept relationshop by mapping them through Association mapping..

Need of keeping Db tables in relationship

=>Storing multiple entities info/details in single db table is bad pratice.. which many give
- a) Data redundecny problem
- b) Data Maintenance Problem

Problem:

user_phone_number_details (db table)

userId	uname	addrs	phoneNumber	numberType	provider
101	raja	hyd	99999999	office	airtel
101	raja	hyd	88888888	residence	vi
102	rajesh	vizag	77777777	office	airtel
102	rajesh	vizag	66666666	personal	jio

user details are duplicated i.e
data redundency problem is raised.

Solution1: take two different db table to maintain user details and phone number details (Bad)

user_details (db table1)

userId	uanme	addrs
101	raja	hyd
102	rajesh	hyd

Phone_numbers (db table2)

phone(pk)	numberType	provider
99999999	office	airtel
88888888	residence	vi
77777777	office	airtel
66666666	personal	jio

=>Data redundency problem is solved.. but data navigation problem is raised i.e means we can not access phone number details from user table and vice-versa becoz the db tables are not relationship.

Solution2: keep two db table in foreign key relationship (bad)

parent table — user_details (db table1)

userId	uanme	addrs	(FK) phone
101	raja	hyd	99999999
101	raja	hyd	88888888
102	rajesh	hyd	77777777
102	rajesh	hyd	66666666

child db table — Phone_numbers (db table2)

phone(pk)	numberType	provider
99999999	office	airtel
88888888	residence	vi
77777777	office	airtel
66666666	personal	jio

data redundency poblem

=>Becoz foreign key colum , the data navigation possible b/w two db tables but data redundency poblem is raised db table1(parent table)

Solution3: take two db tables in relation using foreign key column that is placed in child db table

parent table — user_details (db table1)

userId (pk)	uanme	addrs
101	raja	hyd
102	rajesh	hyd

childtable — Phone_numbers (db table2)

regNo(pk)	phone	numberType	provider	user_id(fk)
1001	99999999	office	airtel	101
1002	88888888	residence	vi	101
1003	77777777	office	airtel	102
1004	66666666	personal	jio	102

=>Here there is no data redundency problem and data navigation problem.. i.e it is the perfect way of maintaining data in db table for multiple entities..

note:: As part of good Db tables designing.. it is recomanded to take foreign column in child db tables..

Using FK column support ,we can keep db tables in 4 types relationships
- a) One to One [1 --- 1] (eg:: Person <-----> Passport)
- b) One To many [1 ----*] (eg: PErson <----> Visa) (eg: Person <-------> PhoneNumber) (eg: Dept <----------> Employee)
- c) Many To One [* .... 1] (eg:: Visa <-----> Person) (eg: PhoneNumber <---------->Person) (eg: Employee<----------> Dept)
- d) Many To Many [* ... *] (eg:: Patient <-----> Doctor) (eg: Student <----------> Course) (eg:: Teacher <-------->Student)

=>if Db tables in relationship/multiplicitly then the associated Entity classes should also be kept relationship.. By designing them with special properites..

=> we can keep db table in relationship using fk colum.. but there is no fk column in java , so we need to keep entity classes in relation through compositon by taking reference type properties (non-collection type) or reference type properties (collection type)

This indicates mismatch b/w Java and DB s/w.. becoz the way we desing db tables relationship is different from the way we design java classes in relationship .. This mismatch is called assocation mismatch .. and we can solve this by association mapping..

Db tables in oracle
==========
same as solutions3

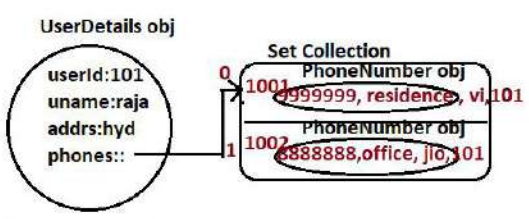
Entity classes
==========

parent class

```
public class UserDetails{
  private Integer userId;
  private String uname;
  prviate String addrs;
  private Set<PhoneNumber> phones;
  //setter... && getters
  ...
}
```

child class

```
public class PhoneNumber{
  private Integer regno;
  private Long phone;
  private String numberType
  private String provider;
  private Integer user_id; //optional :: taking property in child class
                           //for the foreign key column is optional
  //setters && getters
    ....
    ....
}
```



Hibernate supports two modes assocation mapping
- a) Uni-Directional Association mapping (P -----> C or C-----> P)
  - =>Here we can access child objs from parent objs or parent objs from child objects (not both navigations at a time)
- b) Bi-Directional Association Mapping (P <------> C)
  - =>Here we can access childs from parent object and vice-versa (both navigations at a time)

we can implement Entity classes level assoication in two ways
- a) Using Collection (One to Many, many to many)
  - => Entity should hold multiple associated entity objects
- b) With out using Collection (one to one , many to one) (Taking Entity type reference)
  - =>Here Entity should hold only one associated object

To build AssociationMapping in xml driven cfgs we can use <one-to-many> , <many-to-one> ,<one-to-one> ,<many-to-many> tags.

To build AssociationMapping in annotation driven cfgs we can use @OneToOne ,@OneToMany ,@ManyToOne ,@ManyToMany

One To Many Uni-Directional Assoication using xml driven configurations (parent to child)

==========

User ====> Phonenumber (one user can have 1 or more phone numbers , so it is called One to many assoication)

=>since , we want to implement it as one to many uni-directional assocation having parent to child navigation.. So parent Entity class should have special collection type property to hold multiple objects of child class (multiple phone numbers)

User_Info (parent db table)
- |-->userId (n) (pk)
- |-->username (vc2)
- |--->addrs (vc2)

PhoneNumber (child db table)
- |--->regNo (n)(pk)
- |--->phone (n)
- |--->type (vc2)
- |-->provider(vc2)
- |-->user_id(fk)

=>while entty classes for association based on db table in assocation.. taking property in entity class for fk property is optional..

UserInfo.java

==========

```
@Data
public class UserInfo implements Serializable {
    private int userId;
    private String username;
    private String addrs;
    private Set<PhoneNumber> phones; // special property to build 1..* assocication (By holding muliple objects of child class)
}
```

PhoneNumber.java

```
@Data
public class PhoneNumber implements Serializable {
    private int regNo;
    private long phone;
    private String type;
    private String provider;

}
```

UserInfo.hbm.xml

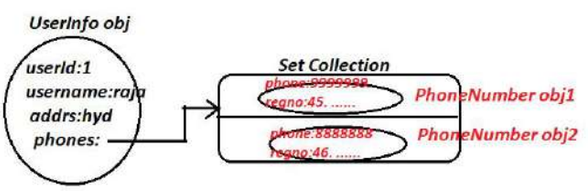
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.nt.entity">
  <class name="UserInfo" table="USER_INFO">
    <id name="userId" type="int">
      <generator class="increment"/>
    </id>
    <property name="username" type="string" length="15"/>
    <property name="addrs" type="string" length="15"/>
    <set name="phones" cascade="all" >
      <key column="user_id"/> <!-- fk column -->
      <one-to-many class="com.nt.entity.PhoneNumber"/>
    </set>
  </class>
</hibernate-mapping>
```

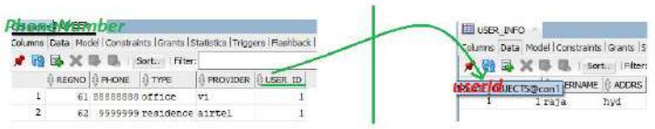
cascade="all" indicates any non-select persistence operation perfomed on prarent object also takes place on the child objs

The set collection phones hold multiple PhoneNumber class objs as child objs based on the fk col user_id

phoneNumber.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.nt.entity">
  <class name="PhoneNumber" table="PHONENUMBER">
    <id name="regNo" type="int">
        <generator class="sequence"/>
    </id>
    <property name="phone" type="long"/>
    <property name="type" type="string" length="10"/>
    <property name="provider" type="string" length="15"/>
  </class>
</hibernate-mapping>
```





while saving object (ses.save(user))

==========

a) inserts record to parent table by collecting data from parent obj.

b) inserts records to child table by collection data from childs objects that are associated with parent obj.

c) udates child table fk colum with parent table pk col vlaues.

note:: if any prorblem/exception raises in parent /child table record insertion. then automatically other db table record insertion falls.. becoz we are executing this code in TxMgmt..

Select operation in one to many Uni-Directional Assocation (paren to child)

==========

=>Here we can access child objects (PhoneNumber objs) from Parent obj (UserInfo object) but reversse is not possible..

=> if we load parent objs.. the associated child objects will be load lazily and automatically...

```
//Load parent object
    Query query=ses.createQuery("from UserInfo");
    List<UserInfo> list=query.getResultList();
    list.forEach(user->{
        System.out.println("parent::"+user.getUserId()+" "+user.getUsername()+" "+user.getAddrs());
        //get associated childs of a parent
        Set<PhoneNumber> childs=user.getPhones();
        childs.forEach(ph->    System.out.println("child :"+ph));
    });
```

Select Operation in One To Many Uni-directional Association

=> In this assoication by default lazy loading is enabled.. i.e the parent objs will be loaded normally but the associated child objects will be lazily i.e they will be loaded on demand basis. To control this behavoir we can use "lazy" attribute of <set>/<list>/<bag>/<map> and etc.. tags where the assocation is built.

possible values for "lazy" attribute are
- lazy="true" (default)
- lazy="false"
- lazy="extra" (best)

```
<set name="phones" cascade="all" lazy="extra">
    <key column="user_id"/> <!-- fk column -->
    <one-to-many class="com.nt.entity.PhoneNumber"/>
</set>
```

lazy="true" (default)

=>Loads the parent/main objects normally, but loads the assoicated childs object lazily on demand basis.. if we try to perform even aggregate operations like counting no.of childs by getting size of childs collection then also all the child objects will be loaded by hitting the DB s/w.

```
Query query=ses.createQuery("from UserInfo");
    List<UserInfo> list=query.getResultList();
        list.forEach(user->{
            System.out.println("parent::"+user.getUserId()+" "+user.getUsername()+" "+user.getAddrs());
            //get associated childs of a parent
            Set<PhoneNumber> childs=user.getPhones();
            System.out.println(childs.isEmpty()); ⇨
            //childs.forEach(ph->    System.out.println("child :"+ph));*/
    });
```

For this aggregate operation on child collection.. it will try to load all child objs of a parent by hitting the Db s/w.

lazy="extra" (best)

=>same as lazy="true".. but does not hit the Db s/w to load the child objs of parent objs if we perform aggragate opeations on childs collection.. for that it generate seperate aggregate query instead of loading all childs objs of parent objects.

lazy="false"

=>perform eager/early loading .. i.e childs objects will be loaded along with parent objs irrespective of wheather they utilized or not..

"cascade" attribute related "cascading" operations are purely related to non-select persistnece operations(insert,update,delete) that we are performing.. it is no way related select operations. similary "lazy" attribute related lazy/eager loading are purely related to select operations and does not effect non-select operations..

Can explain how can we perform lazy loading in hibernate ?

=> In single row operation by using ses.load(-,-) method
=> In single db table HQL Query by using query.iterate() method (deprecated)
=> In One To many and Many To Many Using lazy="true" or lazy="extra"
=> In Many ToOne and One to One Using lazy="proxy" or lazy="no-proxy"

=>lazy loading hibernate makes HB App hitting the DB s/w only when need is there.. i.e it delays the process of hitting Db s/w.. as long as it can..

Adding new child to exising parent

```
//get Session
    Session ses=HibernateUtil.getSession();
        Transaction tx=null;
            boolean flag=false;
            try {
                if(!ses.getTransaction().isActive())
                    tx=ses.beginTransaction();
                //Load existing Parent object
                UserInfo user=ses.get(UserInfo.class,1);
                //get all existing childs of a parent
                Set<PhoneNumber> childs=user.getPhones();
                //create new Child
                PhoneNumber ph=new PhoneNumber(6888888, "personal", "jio");
                //add existing childs
                childs.add(ph);
                flag=true;
            }//try
            catch(HibernateException he) {
                he.printStackTrace();
                flag=false;
            }
            finally {
                //Tx Mgmt
                if(flag) {
                    tx.commit();
                    System.out.println("new child is added to existing parent");
                }
                else {
                    tx.rollback();
                    System.out.println("problem in adding new child to existing parent");
                }
                    //close session factory
                HibernateUtil.closeSessionFactory();
            }//finally
```

Deleting a exising parent and its childs

=> If we delete parent obj by loading that object.. then its associated child objs will be deleted automatically becoz while loading parent object the associated child object will be loaded automatically..

```
//get Session
    Session ses=HibernateUtil.getSession();
    Transaction tx=null;
    boolean flag=false;
    try {
        if(!ses.getTransaction().isActive())
            tx=ses.beginTransaction();
        //Load a parent
        UserInfo user=ses.get(UserInfo.class, 2);
        //delete the parent
        ses.delete(user);
        flag=true;
    }//try
    catch(HibernateException he) {
        he.printStackTrace();
        flag=false;
    }
    finally {
        //Tx Mgmt
        if(flag) {
            tx.commit();
            System.out.println(" parent and its childs are deleted successfully ");
        }
        else {
            tx.rollback();
            System.out.println("Problem in deleting a parent and its childs");
        }
            //close session factory
        HibernateUtil.closeSessionFactory();
    }//finally
```

Deleting all parents and their childs

```
Session ses=HibernateUtil.getSession();
    Transaction tx=null;
    boolean flag=false;
    try {
        if(!ses.getTransaction().isActive())
            tx=ses.beginTransaction();
        //Load all parent objs.. so assocatied child objs of those parent objects will be loaded automatically
        Query query=ses.createQuery("from com.nt.entity.UserInfo");
      List<UserInfo> list=query.getResultList();
      list.forEach(user->{
        ses.delete(user);
      });
  flag=true;
    }//try
        finally {
            //Tx Mgmt
            if(flag) {
                tx.commit();
                System.out.println(" all parents and their childs are deleted successfully ");
            }
            else {
                tx.rollback();
                System.out.println("Problem in deleting all parent and their childs");
            }
                //close session factory
            HibernateUtil.closeSessionFactory();
        }//finally
```

=>Do not use delete HQL Query directly to delete parent/main records when they are having associated child records becoz that throws SQLIntegrity Exception

```
Query query=ses.createQuery("delete from com.nt.entity.UserInfo ");          //throws Exception::
java.sql.SQLIntegrityConstraintViolationException: ORA-02292: integrity constraint
(SYSTEM.FKTDTSU7E61OWMWCPRRSQ3GPLC) violated - child record found
```

In One to Many Assocication While deleting all childs of a parent or one child from collection of childs belonging to a parent the Hibernate will not delete child records completely.. rather it makes the child records orphan by emptying the FK column values.. The child table record that has lost link with parent db table record with null value in FK column is called orphan record..

USER_INFO (parent table)

	USERID	USERNAME	ADDRS
1	1	suresh	hyd
2	2	suresh	hyd

PHONENUMBER (child table)

	REGNO	PHONE	TYPE	PROVIDER	USER_ID
1	73	68888888	officee	vi	(null)
2	72	7999999	residence	airtel	(null)

orphan records..

To instruct hibernate not to give any orphan records.. in any kind deletion process, we need to use cascade="all-delete-orphan" in relavent assocication mapping tags..

```
<set name="phones" cascade="all-delete-orphan" lazy="true">
    <key column="user_id"/> <!-- fk column -->
    <one-to-many class="com.nt.entity.PhoneNumber"/>
  </set>
```

Code to delete all childs of a parent

```
Session ses=HibernateUtil.getSession();
        Transaction tx=null;
        boolean flag=false;
        int count=0;
        try {
            if(!ses.getTransaction().isActive())
                tx=ses.beginTransaction();
            //Load parent
            UserInfo user=ses.get(UserInfo.class, 2);
            //get All childs of a loaded parent
            Set<PhoneNumber> phones=user.getPhones();
            phones.removeAll(phones);
            flag=true;
        }//try
        finally {
            //Tx Mgmt
            if(flag) {
                tx.commit();
                System.out.println(" all childs of parent are deleted successfully ");
            }
            else {
                tx.rollback();
                System.out.println("Problem in deleting all childs of a parent");
            }
                //close session factory
            HibernateUtil.closeSessionFactory();
        }//finally
```

Deleting one child from collection of childs belonging to parent

==========

note:: Set collection can not delete its elements based on the indexes.. So we need to pass object as the argument value to Setcollection.remove(-)

=>If Set Collection is used to built One To many assocation.. To delete one child from collection of childs beloging to a parent.. we must load that child object from DB in order to pass to setcollection.remove(-) method as argument value.

```
Session ses=HibernateUtil.getSession();
        Transaction tx=null;
        boolean flag=false;
        int count=0;
        try {
            if(!ses.getTransaction().isActive())
                tx=ses.beginTransaction();
            //Load parent object(UserInfo) from DB
            UserInfo user=ses.get(UserInfo.class,2);
            //get All childs of a parent
            Set<PhoneNumber> childs=user.getPhones();
            //Load specific child(PhonerNumber) from Db
            PhoneNumber ph=ses.get(PhoneNumber.class, 73);
            childs.remove(ph); //removing specific child from parent
            flag=true;
        }//try
        finally {
            //Tx Mgmt
            if(flag) {
                tx.commit();
                System.out.println(" one child from collection of childs Belonging to parent deleted ");
            }
            else {
                tx.rollback();
                System.out.println(" one child from collection of childs Belonging to parent not deleted");
            }
                //close session factory
            HibernateUtil.closeSessionFactory();
        }//finally
```

cascade operations in Associations

=> cascading is all about propagating/passing the non-select persistence operations performed on the parent/main objs to child / associated objects..
=>casacading is no way related to Loading of obejcts..

In <set>/<map>/<list> and etc..tags of one to many / many to many association mapping is having "cascade" attrbute supporting the possible values..

cascade="all" :: propagates all non-select persitence operations to associated objects
cascade="save-update" :: propagates insert,update persitence operations to associated objects
cascade="delete" :: propagates delete persitence operations to associated objects
cascade="persist" :: propagates insert persitence operations to associated objects
cascade="merge" :: propagates merge persitence operations to associated objects
cascade="delele-orphan" ::propagates delete persitence operation to associated objects but also destroys the orphan records.
cascade="all-delele-orphan" (best) ::propagates all non-select persitence operations to associated objects but also destroys the orphan records.
cascade="none" (default):: does not propagate any non-select persistence operation to associated objects

we can give multiple multiple values cascade attribute as comms seperated list of vlaue like cascade="persist, delete"

note:: cascade="save-update" allows to use ses.save(-) or ses.persist(-) method to perform save object operation where as cascade="persist" allows us to use only ses.persist(-) .

note: cacade="delete" makes ses.delete(-) method not to associated records completely. rather it makes the associated/child records as orphan records by emptying the the FK column.. To avoid this use cascade="all-delete-orphan" or cascade="delete-orphan"

Taking Collection property as java.util.List in One To many association mapping

=>Since set collection does not maintain indexes for elements.. So we can not maintain child record indexes with respect to parent records. To solve this problem use the colelction property as List collection property

=> java.util.List type property must be cfg using <list> having the configuration of index column (index column is mandatory) This index column default base value 0 and it can be changed any other numeric value (geernally we take 1)

UserInfo.java (parent class)
==========

```
@Data
@NoArgsConstructor
@RequiredArgsConstructor
public class UserInfo implements Serializable {
    private int userId;
    @NonNull
    private String username;
    @NonNull
    private String addrs;
    private List<PhoneNumber> phones;
    ....
}
```

PhoneNumber.java (child class)

```
@Data
@RequiredArgsConstructor
public class PhoneNumber implements Serializable {
    private int regNo;
    @NonNull
    private long phone;
    @NonNull
    private String type;
    @NonNull
    private String provider;
}
```

UserInfo.hbm.xml

```
<hibernate-mapping package="com.nt.entity">
  <class name="UserInfo" table="USER_INFO_LIST">
    <id name="userId" type="int">
      <generator class="increment"/>
    </id>
    <property name="username" type="string" length="15"/>
    <property name="addrs" type="string" length="15"/>
    <list name="phones" cascade="all-delete-orphan" lazy="true">
      <key column="user_id"/> <!-- fk column -->
      <list-index base="1" column="PHONE_INDX" />
      <one-to-many class="com.nt.entity.PhoneNumber"/>
    </list>
  </class>
</hibernate-mapping>
```

PhoneNumber.hbm.xml

```
<hibernate-mapping package="com.nt.entity">
  <class name="PhoneNumber" table="PHONENUMBER_LIST">
    <id name="regNo" type="int">
      <generator class="sequence"/>
    </id>
    <property name="phone" type="long"/>
    <property name="type" type="string" length="10"/>
    <property name="provider" type="string" length="15"/>
  </class>
</hibernate-mapping>
```

USER_INFO_LIST (parent db table)

	USERID	USERNAME	ADDRS
1	1	suresh	hyd
2	2	suresh	hyd

PHONENUMBER_LIST (Child db table)

	REGNO	PHONE	TYPE	PROVIDER	USER_ID	PHONE_INDX
1	89	7999999	residence	airtel	1	1
2	90	68888888	officee	vi	1	2
3	91	7999999	residence	airtel	2	1
4	92	68888888	officee	vi	2	2

index column

=> while working with Set collection bassed special property.. we must load child object from DB ... for deleting that object from collection of childs belonging to a parent becoz Set collection does not allow to work with indexes for the elements. but if the special propety is ListColection we can access and delete elements through index so there is no need of loading child object from Db s/w..

Code with List Collection
==========

```
//Load parent object
UserInfo user=ses.get(UserInfo.class,2);
    //get All childs of a Parent
    List<PhoneNumber> childs=user.getPhones();
    //delete specific child element through index from the list colleciton
     childs.remove(0);
```

---

code with Set Collection

```
//Load parent object
 UserInfo user=ses.get(UserInfo.class,2);
 //get All childs of a Parent
     Set<PhoneNumber> childs=user.getPhones();
 //load child object
 PhoneNumber ph=ses.get(PhoneNumber.class,75);
//delete specific child element through index from the list colleciton
 childs.remove(ph);
```

=>To use the benifits List Collection type special property with out having the configuration of index column , we must cfg that List Collection property by using <bag> tag.

example code
==========
=>All are same execept mappings in UserInfo.hbm.xml

```
<hibernate-mapping package="com.nt.entity">
  <class name="UserInfo" table="USER_INFO_BAG">
    <id name="userId" type="int">
      <generator class="increment"/>
    </id>
    <property name="username" type="string" length="15"/>
    <property name="addrs" type="string" length="15"/>
    <bag name="phones" cascade="all-delete-orphan" lazy="true">
      <key column="user_id"/> <!-- fk column -->
      <one-to-many class="com.nt.entity.PhoneNumber"/>
    </bag>
  </class>
</hibernate-mapping>
```

No index column is required

Taking property for FK column of child /parent db table is optional, if u take then cfg that property having insert="false" and update="false" in the mapping file.. to avoid direct insertion/update operations on Db table columns.

In PhoneNumber.hbm.xml

```
<property name="unid" column="user_id" type="int" insert="false" update="false"/>
```

USER_INFO_BAG (parent db table)

	USERID	USERNAME	ADDRS
VORKLOAD@con1	1	suresh	hyd

PHONENUMBER_BAG (child db table)

	REGNO	PHONE	TYPE	PROVIDER	USER_ID
1	97	7999999	residence	airtel	1
2	98	68888888	officee	vi	1

no index column here..

Taking java.util.Map collection as special property type in One to Many Association

=>if the special property type is List collection , then we can get numeric indexes for the childs of parent.

=>if the special property type is Map collection , then we can take the keys of Map collection as the indexes for childs of parents. This allow us to take different types of keys as the indexes.

UserInfo.java (parent class)

```
@Data
//@NoArgsConstructor
@RequiredArgsConstructor
public class UserInfo implements Serializable {
        private int userId;
        @NonNull
        private String username;
        @NonNull
        private String addrs;
        private Map<String,PhoneNumber> phones;
}
```

PhoneNumber.java (child class)

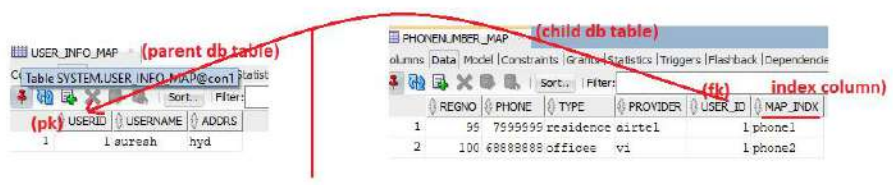
```
@Data
@RequiredArgsConstructor
public class PhoneNumber implements Serializable {
        private int regNo;
        @NonNull
        private long phone;
        @NonNull
        private String type;
        @NonNull
        private String provider;
        private int unid;
}
```

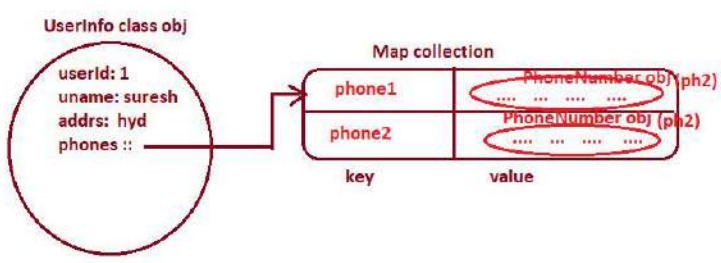
UserInfo.hbm.xml

```
<hibernate-mapping package="com.nt.entity">
  <class name="UserInfo" table="USER_INFO_MAP">
    <id name="userId" type="int">
      <generator class="increment"/>
    </id>
    <property name="username" type="string" length="15"/>
    <property name="addrs" type="string" length="15"/>
    <map name="phones" cascade="all-delete-orphan" lazy="extra">
      <key column="user_id"/> <!-- fk column -->
      <map-key column="MAP_INDX" type="string"/>
      <one-to-many class="com.nt.entity.PhoneNumber"/>
    </map>
  </class>
</hibernate-mapping>
```

PhoneNumber.hbm.xml

```
<hibernate-mapping package="com.nt.entity">
  <class name="PhoneNumber" table="PHONENUMBER_MAP">
    <id name="regNo" type="int">
        <generator class="sequence"/>
    </id>
    <property name="phone" type="long"/>
    <property name="type" type="string" length="10"/>
    <property name="provider" type="string" length="15"/>
    <property name="unid" column="user_id" type="int" insert="false" update="false"/>
  </class>
</hibernate-mapping>
```





Taking special property of one to many Assocation as array type property

=> Not recomanded to take becoz
- ->arrays are not dynamically growable
- -> we can not delete array elements
- -> we can not add dyamic elements to the arrays
- -> lazy loading is not possible

UserInfo.java (parent class)

```
@Data
//@NoArgsConstructor
@RequiredArgsConstructor
public class UserInfo implements Serializable {
        private int userId;
        @NonNull
        private String username;
        @NonNull
        private String addrs;
        private PhoneNumber[] phones;
  ...
}
```

PhoneNumber.java (child class)

```
@Data
@RequiredArgsConstructor
public class PhoneNumber implements Serializable {
        private int regNo;
        @NonNull
        private long phone;
        @NonNull
        private String type;
        @NonNull
        private String provider;
        private int unid;
}
```

UserInfo.hbm.xml

```
<hibernate-mapping package="com.nt.entity">
  <class name="UserInfo" table="USER_INFO_ARRAY">
    <id name="userId" type="int">
      <generator class="Increment"/>
    </id>
    <property name="username" type="string" length="15"/>
    <property name="addrs" type="string" length="15"/>
    <array name="phones" cascade="all-delete-orphan">
      <key column="user_id"/> <!-- fk column -->
      <!-- <index column="ARRAY_INDX" type="int" length="10"/> -->
      <list-index column="ARRAY_INDX" base="1"/>
      <one-to-many class="com.nt.entity.PhoneNumber"/>
    </array>
  </class>
</hibernate-mapping>
```

<array> tag does not have lazy attribute.

we can <index> or <list-index> to cfg index column

PhoneNumber.hbm.xml

same as previous

Table showing differences among the array , List , Set ,Map type special properties in 1 ... * ,*...* assocation mapping

operation	array	java.util.List	java.util.Set	java.util.Map
Stroing duplicates	yes	yes	no	yes
Mainating index	yes (0 based)	yes (0 based)	no	yes (key based)
To cfg special/ assocation property	<array>	<list>/<bag>	<set>	<map>
Index colunm cfg	mandatory	mandatory with <list> optional with <bag>	not rerquired	mandatory to cfg key values based index column
tag to cfg index colum	<index>/<list-index>	<list-index>	not required	<map-key>
Enabling lazy loading	not possible	possible using lazy attr	possible using lazy attr	possible using lazy attr
Deleting one child from multiple childs beloging to a parent	not possible	can be done through index, so no need of loading child obi	we must load child object form DB	possible by taking key as criteria value

Many To One UniDirectional Association

==========

=>Multiple objects of class will be mapped with single object of parent class..

=>It is reverse of one to many association..

=> It Is non collection based association

=>Generally it is developed as chid to parent association by taking Parent class ref variable in child class to hold the associated parent object (Actually multiple objects of child class will hold single object of parent class)

Example :: Employee ------> Department (Multiple employees belong to same dept So it many to One Association)

Student ---------->College

Kid ----------> Parents

Faculty ---------->college

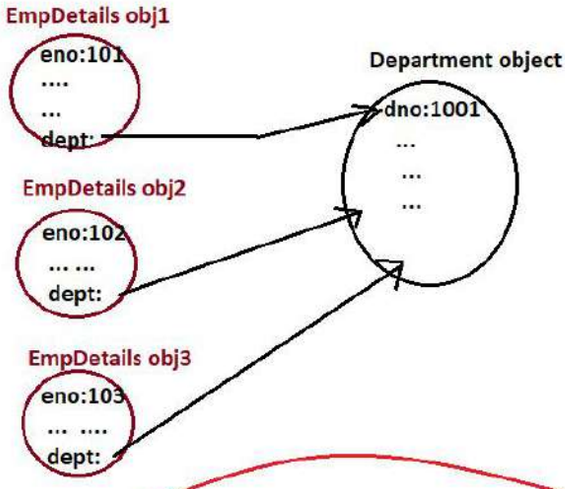
PhoneNumber --------->User

Department.java (parent class)

```
@Data
public class Department implemetns Serializable{
  private int dno;
  private String name;
  private String location;
  private int size;

}
```

EmpDetails.java (child class)

```
@Data
public class EmpDetals implements Serializable{
  private int eno;
  private String ename;
  private String eadd;
  private float salary;
  private Department dept;  //To build many to one association
  }
```



Department (parent db table)

dno(pk)	dname	loacation	size
1001	IT	hyd	10

EmpDetails (child db table)

eno(pk)	ename	eadd	salary	dept_no(fk)
101	raja	hyd	90000	1001
102	ravi	vizag	10000	1001
103	suresh	delhi	80000	1001

Department.hbm.xml

```
<hibernate-mapping>
    <class name="pkg.Department" table="DEPARTMENT">
      <id name="dno">
        <generator class="sequence">
          <param name="sequence-name">dno_seq</param>
        </generator>
      </id>
      <property name="dname"  lenght="15"/>
      <property name="location" length="15"/>
      <property  name="size" />
    </class>
</hiernate-mapping>
```

EmpDetails.hbm.xml

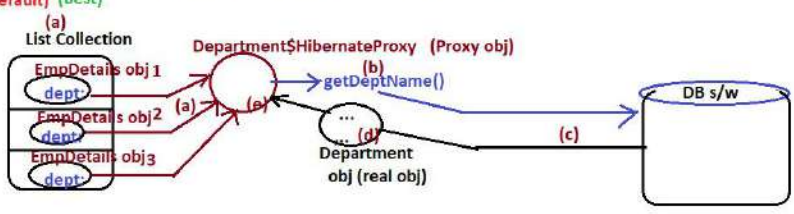
```
<hibernate-mapping>
   <class name="pkg.EmpDetails"   table="EMPDETAILS">
      <id name="eno">
         <generator class="increment"/>
      </id>
     <property name="ename" length="15"/>
     <property name="eadd" length="15"/>
     <property name="salary"/>
     <many-to-one name="dept"
                  class="pkg.Department"
                  column="dept_no"  //fk columnn
                  cascasde="all"/>
   </class>
</hibernate-mapping>
```

"cascade" attrbute of many-to-one tag does not have support of "delete-orphan" and "all-delete-orphan" values becoz there is no possbility of getting orphan records in many to one association.So best value for "cascade" is "all".

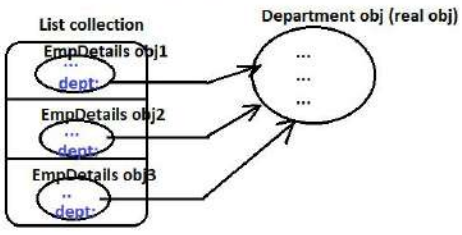
SElect operation in ManyTo One Assocation
==========

=> if we perform load using child entity class .. first it loads all objects but the associated parent objects will be load lazily by taking the support of proxy objects i.e first Proxy objects will be created for the associated parent objects.. when child objs really start accessing parent object by calling non-indentifier methods on proxy objectof parent then parent records will be loaded from Db by creating real parent objects.. The possible vlaues of lazy attribute in <many-to-one> tag are "proxy", "no-proxy" , "false"

lazy="proxy" (Loads chid objects normally but loads the associated parent objects lazily
========== with the support of its proxy objects) (Lazy loading)
(default) (best)



lazy="false" (Performs earger Loading.. associated parent objs will be loaded with childs objects
========== normally irrespective of wheather u r using non-indentifier methods of parent objs or not



=Here no Proxy objects will be created..

lazy="no-proxy" (Perform Lazy loading with out having Proxy objs.. by taking support of byte code instrutmentation -> an extra plugin
========== that we need to add to hibernat libraries)

note:: if we can not arrnage that byte code instrutmentatiom.. lazy="no-proxy" behaves like lazy="proxy".

```
//Load objects
Query query=ses.createQuery("from EmpDetails");
List<EmpDetails> list=query.getResultList();
list.forEach(emp->{
 System.out.println("Child::"+emp.getEno()+" "+emp.getEname()+" "+emp.getEadd()+" "+emp.getSalary());
 //get Associated parent object
 Department dept=emp.getDept();
 System.out.println("Parent::"+dept.getDno()+" "+dept.getDname()+" "+dept.getLocation()+" "+dept.getCapacity());
});
```

Delete operation on <ManyToOne>Assoication
==========

=>if we can delete child objs of a parent then there will be no problem to delete the associated parent obj. otherwise exception will be raised.. java.sql.SQLIntegrityConstraintViolationException: ORA-02292: integrity constraint (SYSTEM.FKCK5BS516FTDQT6DALO0W7W3RX) violated - child record found

Deleting all childs and its parents

```
//get Session
Session ses=HibernateUtil.getSession();
Transaction tx=null;
boolean flag=false;
try {
    if(!ses.getTransaction().isActive())
        tx=ses.beginTransaction();
    //Load all child object
    Query query=ses.createQuery("from EmpDetails");
    List<EmpDetails> list=query.getResultList();
    list.forEach(emp->{
        ses.delete(emp);
    });
    flag=true;

}//try
catch(HibernateException he) {
    he.printStackTrace();
    flag=false;
}
finally {
    //perform TxMgmt
    if(flag) {
        tx.commit();
        System.out.println("All chids and its parents are deleted ");
    }
    else {
        tx.rollback();
        System.out.println("All chids and its parents are not deleted ");
    }
    //close HB objs
    HibernateUtil.closeSessionFactory();
}
```

Deleting one child from collection of childs belonging to a parent *(Bad code)*
==========

```
//get Session
Session ses=HibernateUtil.getSession();
Transaction tx=null;
boolean flag=false;
try {
    if(!ses.getTransaction().isActive())
        tx=ses.beginTransaction();
    //Load child obj
    EmpDetails emp=ses.get(EmpDetails.class, 3);
    ses.delete(emp);
    flag=true;
}
finally {
    //perform TxMgmt
    if(flag) {
        tx.commit();
        System.out.println("One chid of a parent is deleted ");
    }
    else {
        tx.rollback();
        System.out.println("One chid of a parent is not deleted
");
    }
    //close HB objs
    HibernateUtil.closeSessionFactory();
}//finally
```

This code throws exception if cascade="all" ... becoz parent obj of Loaded child object is also having other childs obeject we get java.sql.SQLIntegrityConstraintViolationException: ORA-02292: integrity constraint (SYSTEM.FKCK5BS516FTDQT6DALO0W7W3RX) violated - child record found

To solve this problem::
a) change cascade="none" (this is not applicable in realtime becoz we can not cascade attribute value in the middle of the application development)

b) Use HQL Delete Query to delete spefic child record directly

---

Deleting One child from collection childs belonging to parent using HQL delete
==========

```
@Override
public void deleteOneChildFromParent() {
    //get Session
    Session ses=HibernateUtil.getSession();
    Transaction tx=null;
    boolean flag=false;
    try {
        if(!ses.getTransaction().isActive())
            tx=ses.beginTransaction();
        Query query=ses.createQuery("DELETE FROM EmpDetails WHERE eno=:no");
        query.setParameter("no",2);
      int count=query.executeUpdate();
      flag=true;
    }
    finally {
        //perform TxMgmt
        if(flag) {
            tx.commit();
            System.out.println("One chid of a parent is deleted ");
        }
        else {
            tx.rollback();
            System.out.println("One chid of a parent is not deleted ");
        }
        //close HB objs
        HibernateUtil.closeSessionFactory();
    }//finally
}
```

One To Many /Many To One Bi-Directional Association

==========

=>In Bi-Directional Assoication parent to child and child to parent navigation is possible i.e we can access child objects from parent objs and parent obj from child objects.

=>To build one TO many Bi-Directional Association parent class should have collection property to hold bunch of child objects and child class should have parent class ref variable to hold associated parent object.

=>What is navigation mismatch problem in Hibernate?

=>Using single FK Column we can navigate from parent db table records to child db table records and vice-versa. But same thing is not possible while with working classes i.e Both parent and child classes should have special properties to build bi-directional Assoication...

UserInf(parent db table)

userId(pk)	uname	addrs
101	raja	hyd

PhoneNumber (child db table)

regNo(pk)	phone	numberType	provider	user_id(FK)
8901	99999999	office	airtel	101
8902	88888888	residence	gio	102

Here Single FK column is sufficient to naviate bi-directionally (parent to child and child to parent)

classes
=======

```
//parent class
public class UserInfo{
  private Integer userId;
  private String uname;
  private String addrs;
private Set<PhoneNumber> phones;

  //setters && getters
    ....
    ....
}
```

```
//child class
public class PhoneNumber{
  private Integer regNo;
  private Long phone;
  private String numberType;
  private String provider;
  private UserInfo user;
  //setters && getters
    ....
    ....
}
```

User.hbm.xml
==========

```
<hibernate-mapping package="com.nt.entity">
  <class name="UserInfo" table="USER_INFO_BI">
    <id name="userId" type="int">
      <generator class="increment"/>
    </id>
    <property name="username" type="string" length="15"/>
    <property name="addrs" type="string" length="15"/>
    <set name="phones" cascade="all-delete-orphan" lazy="extra" inverse="true" >
      <key column="user_id"/> <!-- fk column -->
      <one-to-many class="com.nt.entity.PhoneNumber"/>
    </sety>
  </class>
</hibernate-mapping>
```

PhoneNumber.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.nt.entity">
<class name="PhoneNumber" table="PHONENUMBER">
  <id name="regNo" type="int">
      <generator class="sequence"/>
  </id>
  <property name="phone" type="long"/>
  <property name="type" type="string" length="10"/>
  <property name="provider" type="string" length="15"/>
  <many-to-one name="user" class="pkg.UserInfo" column="user_id" cascade="all" lazy="proxy">
 </class>
</hibernate-mapping>
```

What is inverse="true" or inverse="false"?

Ans) we can use this only in Bi-Directional Assoication.. Hibernate internally generates multiple SQL queries to build the association from scratch level . In Bi-Directional generated similar SQL queries to build the opposite side of relationship is unneccassary and waste of time. Keeping this in mind if we use inverse="true" that indicateds The current property is owner of the relationship and while build other side of relationship use this side generated SQL queris to improve the perfomence.

In One to many association we should inverse="true" at many side (collection side) , Many To Many association we can place inverse="true" any side. In OneToOne Assocation we can not take inverse="true" (becoz it does not maintain FK column)

note:: "inverse" attribute is avaialble only collection tags like <set> /<list>/<bag>/<map> and etc..

=>The default value of "inverse" attribute is false..i.e each side of relationship should build from scracth level seperately.

HQL Joins
==========
=> Joins are given to data from two connected tables by applying some implicit conditions...
=> While working with SQL Joins the db tables need not be in relationship.. where as while working with HQL Joins the Db tables and their entity classes must be there in relationship
=> As of now HQL supports 4 types of joins
a) Inner join (gives common data of both db tables)
b) Right join /right outer join (gives common data of both db tables
and also give un common data of right side db table/parent db table)
c) Left join /Left outer join (gives common data of both db tables
and also give un common data of left side db table/child db table)
d) full join (gives both common and uncommon data of both db tables)

USER_INFO (parent db table/right side table)

user_id	username	address
101	raja	hyd
102	sunil	vizag

PHONENUMBER (child db table/left side table)

phone_id	phone	number_type	phone_user_id (fk)
991	9999999	office	airtel	101
992	8888888	home	vi	101
993	7777777	office	jio	102

Venn Diagram

Inner join :: gives only "B"
left join :: gives A+B
right join :: gives B+C
full join :: gives A+B+C.

HQL Join Syntax (parent to child)
==========
select [parent class properties, child class properties] from <Parent class> <aliasname> <join type>
parent class HAS-A relation variable <alias name>
Parent class (represents parent/let table)
eg1: select u.userid,u.username,u.addrs,ph.regno,ph.phone,ph.number‡type,ph.provider from UserInfo u
Inner join u.phones ph
left join Parent class special property
right join (represents child/right table)
full join

HQL Join Syntax (child to parent)
==========
select [child properties, parent properties] from <Child class> <aliasname> join type child class HAS-A relation variable <alias name>
eg: select ph.regno,ph.phone,ph.number‡type,ph.provider , u.userid, u.username,u.addrs from PhoneNumber ph
Inner join ph.user u
left join Child class special property
right join
full join

Example code
==========

```
@Override
public void loadDataUsingParentToChildHQLJoins() {

    // get Session
    Session sen=HibernateUtil.getSession();
    Transaction tx=null;
    try {
        if(!sen.getTransaction().isActive())
            tx=sen.beginTransaction();
        //execute HQL query
        // Query query=ses.createQuery("select u.userid,u.username,u.addrs,ph.regno,ph.phone,ph.type,ph.provider from  UserInfo u inner join u.phones ph");
        // Query query=ses.createQuery("select u.userid,u.username,u.addrs,ph.regno,ph.phone,ph.type,ph.provider from  UserInfo u left join u.phones ph");
        // Query query=ses.createQuery("select u.userid,u.username,u.addrs,ph.regno,ph.phone,ph.type,ph.provider from  UserInfo u right join u.phones ph");
        Query query=ses.createQuery("select u.userid,u.username,u.addrs,ph.regno,ph.phone,ph.type,ph.provider from  UserInfo u full join u.phones ph");

        List<Object[]> list=query.getResultList();
        list.forEach(row->{
            for(Object val:row)
                System.out.print(val+"  ");
            System.out.println();
        });

    }//try
    catch(HibernateException he) {
        he.printStackTrace();
    }
    finally {
        //close  ohbs
        HibernateUtil.closeSessionFactory();
    }

}//method

@Override
public void loadDataUsingChildToParentHQLJoins() {
    // get Session
    Session sen=HibernateUtil.getSession();
    Transaction tx=null;
    try {
        if(!sen.getTransaction().isActive())
            tx=sen.beginTransaction();
        //execute HQL query
        //Query query=ses.createQuery("select ph.regno,ph.phone,ph.type,ph.provider,u.userid,u.username,u.addrs from  PhoneNumber ph inner join  ph.user u");
        //Query query=ses.createQuery("select ph.regno,ph.phone,ph.type,ph.provider,u.userid,u.username,u.addrs from  PhoneNumber ph left join  ph.user u");
        //Query query=ses.createQuery("select ph.regno,ph.phone,ph.type,ph.provider,u.userid,u.username,u.addrs from  PhoneNumber ph right join  ph.user u");
        Query query=ses.createQuery("select ph.regno,ph.phone,ph.type,ph.provider,u.userid,u.username,u.addrs from  PhoneNumber ph full join  ph.user u");

        List<Object[]> list=query.getResultList();
        list.forEach(row->{
            for(Object val:row)
                System.out.print(val+"  ");
            System.out.println();
        });

    }//try
    catch(HibernateException he) {
        he.printStackTrace();
    }
    finally {
        //close  ohbs
        HibernateUtil.closeSessionFactory();
    }

}
```

What is 1-m SELECT Problem and how to solve this problem?

While working with one to many or many to one association , the hibernate f/w generates "1" select SQL query to get All parent obj/records and generates "n" select queries to get the childs of "n" parents i.e total 1-m Select HQL Queries will be generated to get all parents and its child records. Generating more SQL queries means...hitting the DB u/w for more times.. This degrades the performance.. So 1-m select Queries generation looks like a problem...

Let us assume 2 parents are there each having 2childs

parent table

child table

One To many Association : 1-m select problem
Many To One Assocation :: m+1 select problem

select * from User_Info (1 select query to get all parent records)
---- //gives two parent records
---
select * from PhoneNumber where user_id=? (?)

---
select * from PhoneNumber where user_id=0 (?)

---

2 select queries to get the child records
two parent records (n queries)

Solution:: Generate Single Select SQL query to get both parent and child records...

The possible solutions for 1m/1n-1 select problem is
a) GBC/PCBC with fetch type JOIN
b) Using Fetch HQL Join (Entity query)
c) Using HQL Join As scalar query
d) @KxsToMany(fetch=FetchType.EAGER)
One
Many

we can not specify fetch type with HQL/JPQL or Native SQL

annotation driven cfgs

a) GBC/PQBC with fetch type JOIN
==========
Using GBC (old style)
==========

```
            // get Session
            Session sen=HibernateUtil.getSession();
            Transaction tx=null;
            try {
                if(!sen.getTransaction().isActive())
                    tx=sen.beginTransaction();
                //prepare Criteria object
                Criteria criteria=sen.createCriteria(UserInfo.class);
                criteria.setFetchMode("phones",FetchMode.JOIN);
                List<UserInfo> list=(List<UserInfo>)criteria.list();
                list.forEach(user->{
                    System.out.println("parent :: "+user);
                    Set<PhoneNumber> childsuser=user.getPhones();
                    childs.forEach(ph->{
                        System.out.println("child:: "+ph);
                    });
                });
            }//try
            catch(HibernateException he) {
                he.printStackTrace();
            }
            finally {
                //close  ohbs
                HibernateUtil.closeSessionFactory();
            }
```

// Instread of writing fetch mode in java code
we can also write in hibernate mapping file
using "fetch" attribute of collection tag <set></dist></map></bag> and etc...

```
<set name="phones"  lazy="extra"  ...fetch="join" />
  ...
  ...
</set>
```

Using JPQBC /JPCriteria API
==========

```
// get Session
            Session sen=HibernateUtil.getSession();
            Transaction tx=null;
            try {
                if(!sen.getTransaction().isActive())
                    tx=sen.beginTransaction();
                //create Criteria Builder
                CriteriaBuilder builderuser=getCriteriaBuilder();
                //create CriteriaQuery object
                CriteriaQuery<UserInfo> criteryuserbuilder=builderuser.createQuery(UserInfo.class);
                //create root object
                Root<UserInfo>root=criteryuser.from(UserInfo.class);
                root.fetch("phones");
                //add Root object CriteriaQuery object
                criteryuser.select(root);
                //prepare Query object
                Query query=ses.createQuery(criteryuser);
                List<UserInfo> list=query.getResultList();
                list.forEach(user->{
                    System.out.println("parent :: "+user);
                    Set<PhoneNumber> childsuser=user.getPhones();
                    childs.forEach(ph->{
                        System.out.println("child:: "+ph);
                    });
                });
            }//try
            catch(HibernateException he) {
                he.printStackTrace();
            }
            finally {
                //close  ohbs
                HibernateUtil.closeSessionFactory();
            }
```

note:: Here fetch="join" attribute kept in collection tags like <set></dist></map></bag> does not work

b) Using Fetch HQL Join (Entity query)

```
            // get Session
            Session sen=HibernateUtil.getSession();
            Transaction tx=null;
            try {
                if(!sen.getTransaction().isActive())
                    tx=sen.beginTransaction();
                //execute HQL Join Query
                Query query=ses.createQuery("select u from UserInfo  u inner join fetch u.phones ph ");
                List<UserInfo> list=query.getResultList();
                list.forEach(user->{
                    System.out.println("parent :: "+user);
                    Set<PhoneNumber> childsuser=user.getPhones();
                    childs.forEach(ph->{
                        System.out.println("child:: "+ph);
                    });
                });
            }//try
            catch(HibernateException he) {
                he.printStackTrace();
            }
            finally {
                //close  ohbs
                HibernateUtil.closeSessionFactory();
            }
```

c) Using HQL Join As scalar query

```
// get Session
            Session sen=HibernateUtil.getSession();
            Transaction tx=null;
            try {
                if(!sen.getTransaction().isActive())
                    tx=sen.beginTransaction();
                //execute HQL Join Query
                Query query=ses.createQuery("select u.userid,u.username,u.addrs ,ph.regno,ph.phone,ph.type,ph.provider from UserInfo u inner join  u.phones ph ");
                List<Object[]> list=query.getResultList();
                list.forEach(row->{
                    for(Object val:row)
                        System.out.print(val+"  ");

                    System.out.println();
                });

            }//try
            catch(HibernateException he) {
                he.printStackTrace();
            }
            finally {
                //close  ohbs
                HibernateUtil.closeSessionFactory();
            }
    }
```

Can u explain different Fetch Types/Modes supported by the Hibernate

Ans) Hibernate supports the following fetch types
a) SELECT (default)
b) SUBSELECT
c) JOIN

we can specify these fetch types
by using "fetch" attribute of collection tags
or by using criteria.setFetchMode(..) (HB GBC) or
root.fetch() (JPA GBC) or using "fetch" param of
@KxsToMany Annotation

fetch="select "

=>Generates 1 SQL query to get all parent records and
generates n SQL queries to get childs records of n parent records... i.e It generates
1+n select queries. It called 1 +n select problem.

fetch="subselect"

=>generates 1 SQL query to get all parent records and
generates another 1 SQL query to get child records of all parent records i.e generates 1+1 queries
totally... This indicates It does not solve 1+n select problem..

fetch="join" (best)

=>Generates 1 SQL query to get All parent records that child records ... Solves 1+n select problem.

While working with criteria.setFetchMode(..) method we specify the following fetch modes
=>FetchMode.SELECT is equal to fetch="select"
=>FetchMode.JOIN is equal to fetch="join" (best)
=>FetchMode.DEFAULT uses the fetch mode specified in "fetch" attribute of collection tag like <set></dist></ /map> and etc...
note:: There is no FetchMode.SUBSELECT ... But we can bring that effect by taking
FetchMode.DEFAULT with fetch="subselect" in collection tag of Hibernate mapping file
If we specify two different modes in collection tag fetch attribute and in criteria.setFetchMode(..) then what will be taking place?
Ans) The fetch mode specified in criteria.setFetchMode(..) takes place.

What is the difference fetch mode/type and lazy/eager loading ?
Ans) Lazy/eager loading speaks about how child object should loaded with respect to parent objs...
If lazy loading is enabled then parent objects will loaded normaly ..but the childs will be loaded lazily on demand
If lazy loading is disabled then both parent and child objects will loaded normaly.

=> fetch type/mode is no way related to Loading child objects with respect parent objs. It is all about the no.of SQL queries
that are generated to get parent and child records

note: If we specify both "fetch" and "lazy" attributes asking for two different opposite operations... then fetch attribute
action will take palce.
for example lazy="extra" fetch="join" ->Still generates single query to both parent and childs
i.e childs are loaded with parents eargerly though
we specify lazy="extra".

=>PCBC supports only two fetch modes/type that is indirectly.
note:: If root.fetch() /root.fetch(..) is called then it behaves fetch type JOIN... otherwise (if not called) It behaves like
fetch type "SELECT" i.e fetch type "SUBSELECT" is not possible here.
note: JPQBC, HQL ,Native SQL and etc... does not work fetch types specified in "fetch" attribute of collection tags like <set></dist></map> and etc...

note:: In HQL we can use fetch JOINS to work Fetch type/mode JOIN.

Many To Many Association
==========

=> It should be always developed as Bi-Directional Association have

=> Here Both Parent and child classes will have special collection type properties So parent can set of child objs and child can hold set of parent objects.

=> It is One To Many Association from parent and also one to many Assocation from child . i.e one or more parents can have one or more childs and vice-versa.

eg::
Programmer <---------->◇Prjoect
teacher <---------->◇student
Student <---------->◇course
Doctor <--------->◇Patient

=> To build this association we need to Db tables
a) Table to hold parent records
b) Table to hold child records
c) Join table -To build the Assoication

3 tables

DOCTOR (table1) (To hold parent records)

did (pk)	name	hospital	qualification
101	raja	Apollo	MD
102	rajesh	KIMS	MD

PATIENT (table2) (To hold child records)

pid(pk)	name	problem
1001	jani	heart surgery
1002	mahesh	mental-instability
1003	anitha	cancer
1004	suresh	corona

DOCTORS_PATIENTS (table3- Join table) (To build the association)

doctor_id (f k)	patient_id(f k)
101	1001
101	1002
102	1001
102	1002
102	1003
102	1004

Composite PK
⇩
i.e primary key is applied on both colums togather (doctor_id and patient_id)

2 Entity classes to build the Many To Many Association

```
//parent class
@Data
public class Doctor implements Serializable{
  private Integer did;
  private String name;
  private String hospital;
  private String   qualification;
  private Set<Patient> patients;
 }
```

```
//child class
@Data
public class  Patient implements Serializable{
 private  Integer  pid;
 private  String name;
  private String problem;
  private Set<Doctor> doctors;
 }
```

Doctor.hbm.xml
---

note:: since join table is having Composite PK on both FK columns So there is not possibility of getting orphan records in many to many assoication.. so the best value for "cascade" is "all" .. not the "all-delete-orphan".

```
<hibernate-mapping>
  <class name="pkg.Doctor"   table="DOCTOR">
     <id name="did"  type="int" >
        <generator class="increment"/>
     </id>
    <property name="name" length="20" type="string"/>
    <property name="hospital" length="20" type="string"/>
     <property name="qualification" length="10" type="string"/>
    <set name="patients"  table="DOCTORS_PATIENTS"  cascade="all" lazy="extra"   inverse="true">
      <key column="doctor_id"/> <!--owing side FK column-->
      <many-to-many class="pkg.Patient"  column="patient_id"/>
    </set>
                                        Non-owing side FK column
  </class>
</hibernate-mapping>
```

Patient.hbm.xml

```
<hibernate-mapping>
  <class name="pkg.Patient"  table="PATIENT">
      <id name="pid"  type="int">
      <generator    name="sequence">
            <param name="sequence">pat_id_seq</param>
       </generator>
      </id>
      <property  name="name" type="string" length="20"/>
      <property name="problem" type="string" length="20"/>
       <set    name="doctors" table="DOCTORS_PATIENTS"  cascade="all" lazy="extra">
        <key column="patient_id"/> owning side FK column
        <many-to-many class="pkg.Doctor"  column="doctor_id"/>
       </set>
                                        Non-owning side FK column
  </class>
</hibernate-mapping>
```

In Many To Many Assication, if we try to perform delete operation.. on child or parent .. it will not delete the record from table1 or table2 directly. becoz that parent/child may have other depedencies.. but delete the association related entries from the 3rd table join table. (In simple it will not delete main records.. it just breaks the relation in the 3rd db table)

```
@Override
    public void deleteAllChildsOfAParent() {
        //get Session
        Session ses=HibernateUtil.getSession();
        Transaction tx=null;
        boolean flag=false;
        try {
                if(!ses.getTransaction().isActive())
                    tx=ses.beginTransaction();
                //Load parent
                Doctor doctor=ses.get(Doctor.class,2);
                //get childs
                Set<Patient> childs=doctor.getPatients();
                childs.removeAll(childs);
                flag=true;

        }
        catch(HibernateException he) {
            he.printStackTrace();
            flag=false;
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        finally {
            if(flag) {
                tx.commit();
                System.out.println(" deleted");
            }
            else {
                tx.rollback();
                System.out.println(" not deleted");
            }
            //close SessionFactory
            HibernateUtil.closeSessionFactory();
        }//finally

    }
```

This code does not effect the Doctor records whose id is 2 but it deletes all entries from 3rd table whose doctor_id is 2 also the associated patient ids from 3rd db table will be deleted.

```
@Override
    public void deleteSpecificChildFromSpecifcParent() {
        //get Session
        Session ses=HibernateUtil.getSession();
        Transaction tx=null;
        boolean flag=false;
        try {
                if(!ses.getTransaction().isActive())
                    tx=ses.beginTransaction();
                //Load parent
                Doctor doctor=ses.get(Doctor.class,6);
                //get childs
                Set<Patient> childs=doctor.getPatients();
                //Load specific child
                Patient patient=ses.get(Patient.class,1006);
                childs.remove(patient);
                flag=true;

        }
        catch(HibernateException he) {
            he.printStackTrace();
            flag=false;
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        finally {
            if(flag) {
                tx.commit();
                System.out.println(" deleted");
            }
            else {
                tx.rollback();
                System.out.println(" not deleted");
            }
            //close SessionFactory
            HibernateUtil.closeSessionFactory();
        }//finally
    }//method
```

=>In Many To Many Assocation, 3rd table is called join db table and it is not having any unique values.. i.e there is no uniqueness in the join db table (3rd table) . To add that uniqueness in 3rd DB table to allow people to access the records 3rd db table directly.. we need to for <idbag> tag based configuration by taking the collection property as List Collection.

Procedure
=========

step1) keep Many To Many Application ready

step2) Take Parent /child class colection property as java.util.List collecitom

In Doctor.java

```
private List<Patient> patients;
```

step3) In Parent Entity mapping file cfg collection property using <idbag> tag.
<bag> with the generator generated id value.

In Doctor.hbm.xml
==========

```
<idbag name="patients" table="DOCTORS_PATIENTS1" cascade="all" lazy="extra" >
    <collection-id type="int" column="relationId">
        <generator class="increment"/>
    </collection-id>
    <key column="doctor_id"/>
    <many-to-many class="com.nt.entity.Patient" column="patient_id" />
</idbag>
```

column maintaing the uniqueness

any generator can be specified other than assigned

step4) keep inverse="true" in the collection tag <set> of child mapping file (Patient.hbm.xml)

step4) perform the regular many to many save object operation.

```
@Override
    public void saveDataUsingParents() {
        //get Session
        Session ses=HibernateUtil.getSession();
        //prepare objects
        Doctor d1=new Doctor("raja","apollo","MD");
        Doctor d2=new Doctor("rajesh","KIMS","MD");

        Patient p1=new Patient("jani","heart");
        Patient p2=new Patient("anil","kidney");
        Patient p3=new Patient("anitha","cancer");
        Patient p4=new Patient("suresh","corona");
        //set patients to doctors and vice-versa
        d1.setPatients(List.of(p1,p2));
        d2.setPatients(List.of(p1,p2,p3,p4));
        p1.setDoctors(Set.of(d1,d2));
        p2.setDoctors(Set.of(d1,d2));
        p3.setDoctors(Set.of(d2));
        p4.setDoctors(Set.of(d2));

        Transaction tx=null;
        boolean flag=false;
        try {
            if(!ses.getTransaction().isActive())
                tx=ses.beginTransaction();

            //save objects (childs to parent)
          ses.save(d1); ses.save(d2);
            flag=true;
        }//try
        catch(HibernateException he) {
            he.printStackTrace();
        }
        finally {
            //Transaction mgment
            if(flag) {
                tx.commit();
                System.out.println("Objects are saved");
            }
            else {
                tx.rollback();
                System.out.println("objects are not saved");
            }
            //close SessionFactory
            HibernateUtil.closeSessionFactory();
        }//finally

    }//method
```

step5) Observer the db tables

DOCTOR1

DID	NAME	HOSPITAL	QUALIFICATION
1	raja	apollo	MD
2	rajesh	KIMS	MD

PATIENT1

PID	NAME	PROBLEM
1016	jani	heart
1017	anil	kidney
1018	anitha	cancer
1019	suresh	corona

DOCTORS_PATIENTS1

DOCTOR_ID	RELATIONID	PATIENT_ID
1	1	1016
1	2	1017
2	3	1016
2	4	1017
2	5	1018
2	6	1019

Uniquess in 3rd db table

One To One Assoication

==========

=>Here 1 parent object holds one child object (for uni-directional)

=>Here 1 parent object holds one child object and 1 child object hold 1 parent (for bi-directional)

=> It is non-Collection type Assoication i.e The special property that builds the assocation is non-collection type (nothing But HAS-A relation variable is other class reference variable)

=> It can be implemented in two ways

a) Using One To One FK

=>if Parent and child objects want to have their own identity values and they are linked by using FK column then use One to One FK

eg:: Person and IndianPassport , Person And voterCard ,
Person and Driving License

b) Using One To One PK

=>if Parent and child objs want to use same id values then go for One To One PK ..

=>Here There is no need to FK colum .. The PK column itself acts as FK column.

eg:: Student and LibraryMembership (Student Id only will be used as LibraryMembershipId)

eg:: Student and HallTicket (Student Id only will be used as HallTicketNo)

eg:: Car and CarParkingAllotementDeatails

eg:: Employee and CateenCard

One To One FK

==========

=> Here Association is built using FK column

=> <one -to-one> tag is not having attribute to specify FK column.. So use <many-to-one>tag with unique="true", not-null="true" by specifying Fk column to built one to one association.

note:: <one-to-one> tag is give to built One to One PK Assoication , So there is no attribute to specify FK column.

=> Generally we use this assoicatin When child is having Parent.. but there is no guarantee that Parent will have child (One to One Uni-Directional Association from child to parent)

for example Every Passport will belong to a Person.. but every Person need not to have a passport.
Every DivingLicese will belong to a Person.. but every Person need not to have a Driving License.

note:: we prefer One to one PK for Bi-Directional Assoications.. One To One FK for uni-directional Assoications generally..

PErson(parent db table)

pid (pk)	pname	addrs
101	raja	hyd
102	ramesh	vizag

Passport (child db table)

psptNo(pk)	country	expiryDate(date)	person_id(FK)
2346655	india	20-OCT-30	101

Person.java

```
package com.nt.entity;

import java.io.Serializable;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@RequiredArgsConstructor
public class Person implements Serializable{
 private Integer pid;
 @NonNull
 private String pname;
 @NonNull
 private String addrs;
 public Person() {
     System.out.println("Person.0-param constructor");
}
@Override
public String toString() {
     return "Person [pid=" + pid + ", pname=" + pname + ", addrs=" + addrs + "]";
}

}
```

Passport.java

```
import java.io.Serializable;
import java.time.LocalDate;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Setter
@Getter
@RequiredArgsConstructor
public class Passport implements Serializable {
    private Long psptNo;
    @NonNull
    private String country;
    @NonNull
    private LocalDate expiryDate;
    private Person personDetails;

    public Passport() {
        System.out.println("Passport: 0-param constructor");
    }

    @Override
    public String toString() {
        return "Passport [psptNo=" + psptNo + ", country=" + country + ", expirtyDate=" + expiryDate + "]";
    }

}
```

Peson.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.nt.entity">
  <class name="com.nt.entity.Person" table="PERSON_OTOFK">
    <id name="pid" type="int">
      <generator class="increment"/>
    </id>
      <property name="pname" type="string" length="20"/>
      <property name="addrs" type="string" length="20"/>
  </class>
</hibernate-mapping>
```

Passport.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.nt.entity">
  <class name="com.nt.entity.Passport" table="PASSPORT_OTOFK">
    <id name="psptNo" type="long">
        <generator class="sequence"/>
    </id>
    <property name="country" type="string" length="20"/>
    <property name="expiryDate"/>
    <many-to-one name="personDetails"
                 class="com.nt.entity.Person"
                 column="person_id"
                 unique="true"
                 not-null="true"
                  cascade="all"
                 lazy="proxy">

    </many-to-one>
  </class>
</hibernate-mapping>
```

One to One PK
==========

=>Here no FK colunm is required..
=> Use this when both parent and child objs are ready having same id value
=>Here PK column itslef acts as FK colum.. For this we need to use "foreign" generator.

eg:: Student <---> LibraryMemership
(Here Student Id itself can used as LibraryMemeship Id .. So
we can go for One to One PK

=>To build this assocation we can use <one-to-one> tag.. not the <many-to-one> tag
=> "foreign" generator is given collected the id value of associated object and make it as the id value of current object..

=>if we configure "foreign" generator for "LibraryMembership" then it collects the id value of assoicated parent Student object and uses it as the id value of LibraryMembership object.

Student(db table)

sid(pk)	sname	sadd	age
1	raja	hyd	30

LibraryMembership (db table)

lib (pk)	type	startDate	expiryDate
1	gold	20-OCT-11	21-OCT-31

(The child table pk column it self acts as fk column becoz its values are collected from parent table pk colum]

note:: <one-to-one> tag is given to build PK based association , So there is no provision to specify FK column here..
note:: Only FK based Bi-Directional Assocations can enable Inverse assocation to reduce the generation of SQL Queries .. Since <one-to-one> is given for PK based Assoication, So there is no inverse attribute in <one-to-one> tag.

Student.java (Parent class)

```java
package com.nt.entity;

import java.io.Serializable;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Setter
@Getter
@RequiredArgsConstructor
public class Student implements Serializable {
	private Integer sid;
	@NonNull
	private String sname;
	@NonNull
	private String sadd;
	private LibraryMembership libraryDetails;

	public Student() {
	 System.out.println("Student:: 0-param constructor");
	}

	@Override
	public String toString() {
		return "Student [sid=" + sid + ", sname=" + sname + ", sadd=" + sadd + "]";
	}

}
```

LibraryMembership.java

```java
import java.io.Serializable;
import java.time.LocalDate;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@RequiredArgsConstructor
public class LibraryMembership implements Serializable {
	private Integer lid;
	@NonNull
	private String type;
	@NonNull
	private LocalDate startDate;
	@NonNull
	private LocalDate endDate;
	private Student studentDetails;

	public LibraryMembership() {
		System.out.println("LibraryMembership:: 0-param constructor");
	}

	@Override
	public String toString() {
		return "LibraryMembership [lid=" + lid + ", type=" + type + ", startDate=" + startDate + ", endDate=" + endDate
		+ "]";
	}
}
```

Student.hbm.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.nt.entity">
  <class name="com.nt.entity.Student" table="STUDENT_OTOPK">
    <id name="sid" type="int">
      <generator class="increment"/>
    </id>
      <property name="sname" type="string" length="20"/>
      <property name="sadd" type="string" length="20"/>
      <one-to-one name="libraryDetails" class="com.nt.entity.LibraryMembership" cascade="all" lazy="proxy" />
  </class>
</hibernate-mapping>
```

LibraryMembership.hbm.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.nt.entity">
  <class name="com.nt.entity.LibraryMembership" table="LIBRARYMEMBERSHIP_OTOPK">
    <id name="lid" type="int">
      <generator class="foreign">
        <param name="property">studentDetails</param>
        <!-- uses studentDetails property to get associated Student obj(parent)
        and to use that Student object id value as current obj (LibraryMemership) id value. -->
      </generator>
    </id>
    <property name="type" type="string" length="20"/>
    <property name="startDate"/>
    <property name="endDate"/>
    <one-to-one name="studentDetails" class="com.nt.entity.Student" cascade="all" lazy="proxy"/>
  </class>
</hibernate-mapping>
```

note ::Since there is no FK column in one to one PK Association.. there no is possiblity of getting orphan records , So the best possible value for "cascade" attrbute in <one-to-one> association is "all".
note:: In One to One PK the "lazy" attribute possible vlaues are "false" ,"proxy" (best), " no-pxory"

Annotation driven AssociationMapping

=> prefer using annotations in the following order

a) JPA Annotations
b) Jdk Annotations
c) Hibernate Annotations
d) Third Party Annotations

Annotation driven OneToMany Unidirectional Assoication

@OneToMany :: To buid assocation and to specify cascade, fetchtype (lazy/eager/..) values (JPA)
@JoinColumn :: To specify FK column (JPA)
@Fetch :: To specity fetch mode (select/join /subslect) (HB)
and etc..

=>One CarCompany creates multiple models of cars..the The CarCompany and CarModel entity classes in OneToMany Association

CarCompany (parent db table)

compId(pk)	compName	location
1001	Maruthi	noida
1002	Toyota	chennai

CarModel (child db table)

modelId(pk)	modelName	type	fuelType	company_id(fk)
101	baleno	hatchpack	diesel	1001
102	alto	hatchpack	petrol	1001
201	innova	suv	diesel	1002
202	ethois	hatchback	petrol	1002

=> The cascade attribute of Assocation mappjng annotations allow the following values

ALL : CascadeType - CascadeType
DETACH : CascadeType - CascadeType
MERGE : CascadeType - CascadeType
PERSIST : CascadeType - CascadeType
REFRESH : CascadeType - CascadeType
REMOVE : CascadeType - CascadeType

The best value is :: ALL

All these are values of CacadeType enum.

(here no default value for "cascade" parameter/attrbute)

To enable orphanRecords removal process ,we need to use "orphanRemoval=true".

CarCompany.java

```
@Table
@Entity
@Setter
@Getter
@RequiredArgsConstructor
public class CarCompany implements Serializable {
    @Id
    @Type(type="int")
    @GeneratedValue   //takes default sequence hibernate_sequence
    private Integer compId;

    @Column(length=15)
    @Type(type="string")
    @NonNull
    private String compName;

    @Column(length=15)
    @Type(type="string")
    @NonNull
    private String location;

    @OneToMany(targetEntity = CarModel.class,cascade = CascadeType.ALL,fetch = FetchType.LAZY,orphanRemoval =true )
    @JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
    private Set<CarModel> models;

    public CarCompany() {
        System.out.println("CarCompany:: 0-param constructor");
    }

    @Override
    public String toString() {
        return "CarCompany [compId=" + compId + ", compName=" + compName + ", location=" + location + "]";
    }
}//class
```

CarModel.java

```
@Entity
@Setter
@Getter
@RequiredArgsConstructor
public class CarModel {
    @Id
    @Type(type="long")
    @GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
    @SequenceGenerator(name = "gen1",sequenceName = "car_model_seq",initialValue = 101,allocationSize = 1)
    private Long modelId;

    @Type(type="string")
    @Column(length = 20)
    @NonNull
    private String modelName;

    @Type(type="string")
    @Column(length = 20)
    @NonNull
    private String type;

    @Type(type="string")
    @Column(length = 10)
    @NonNull
    private String fuelType;

    public CarModel() {
        System.out.println("CarModel:: 0-param consturctor");
    }

    @Override
    public String toString() {
        return "CarModel [modelId=" + modelId + ", modelName=" + modelName + ", type=" + type + ", fuelType=" + fuelType
                + "]";
    }

}
```

=>JPA suppiled fetch attribute @OneToMany annotation support only values

a) fetch=FetchType.LAZY (default)
b) fetch=FetchType.EAGER

Here there is no provision to bring lazy="extra" effect. To use fetch types in in Hiberante mode (true/false/extra) then use @LazyCollection Annotation as shown below (HB annotation)

Enum values are FALSE(default), TRUE, EXTRA(best)

```
@LazyCollection(LazyCollectionOption.EXTRA)
@OneToMany(targetEntity = CarModel.class,cascade = CascadeType.ALL,fetch = FetchType.EAGER,orphanRemoval =true )
@JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
private Set<CarModel> models;
```

if specify fetch attribute of @OneToMany and @LazyCollection with two different options/value then the HB Annotation @LazyCollection fetchtype/lazy mode will apply.

=>To Specify Fetch Modes we can use @Fetch Annotation given by Hibernate

```
@LazyCollection(LazyCollectionOption.EXTRA)
    @OneToMany(targetEntity = CarModel.class,cascade = CascadeType.ALL,fetch = FetchType.EAGER,orphanRemoval =true )
    @JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
  @Fetch(FetchMode.SELECT)
private Set<CarModel> models;
```

The posible Fetch modes are

FetchMode.SELECT :: Generates 1+n select Queries to get records in one to many assocaiton (default)
FetchMode.SUBSELECT :: Generates 1+1 select Queries to get records in one to many assocaiton
FetchMode.JOIN :: Generates 1 select Query to get records in one to many assocaiton (best)

What is the difference b/w Fetch type and fetch mode?

In Assoication maoing fetch type specifies wheather child records should be loaded lazily or eargely with respect to parent records where as the fetch mode specifies the no.of SQL queries that are required to generate to load paren record and associated child records (like 1, 1+1 , 1+n SQL Quries)

note:: while deleting all childs of a Parent or specific child of parent , the hibernate will not execute delete child table records.. rather it makes child table records as orphan records. To avoid that is happening we need to place orphanRemoval=true

```
@LazyCollection(LazyCollectionOption.EXTRA)
@OneToMany(targetEntity = CarModel.class,cascade = CascadeType.ALL,fetch = FetchType.EAGER,orphanRemoval = true )
@JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
@Fetch(FetchMode.SELECT)
private Set<CarModel> models;
```

Taking Collection property as List Collection in one to many Association

=> While List collection type special property to build one to many association , we must take Index/Order column to maintain the indexes of child objs with respect to parent object.

In CarCompany.java (Parent class)

```
@LazyCollection(LazyCollectionOption.EXTRA)
@OneToMany(targetEntity = CarModel.class,cascade = CascadeType.ALL,fetch = FetchType.EAGER,orphanRemoval = true )
@JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
//@IndexColumn(name = "LST_INDX",base = 1) HB specific .. Depreated
@OrderColumn(name = "LST_INDX")
@ListIndexBase(value = 1)
@Fetch(FetchMode.SELECT)
private List<CarModel> models;
```

(or)

CARCOMPANY_LIST (parent db table)

COMPID	COMPANY	LOCATION
171	maruthi-suzuki	noida

CARMODEL_LIST (Child db table)

MODELID	FUELTYPE	MODELNAME	TYPE	COMPANY_ID	LST_INDX
123	diesel	Baleno	HatchBack	171	1
124	diesel	Brezza	HatchBack	171	2

but

To bring the effect of <bag> nothing taking special property as List Collection , not having index/order column in the db table .. For this do no take cfg Order/index column while working List Collection type Special property.

```
@LazyCollection(LazyCollectionOption.EXTRA)
@OneToMany(targetEntity = CarModel.class,cascade = CascadeType.ALL,fetch = FetchType.EAGER,orphanRemoval = true )
@JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
@Fetch(FetchMode.SELECT)
private List<CarModel> models;
```

CARCOMPANY_LIST_BAG (parent db table)

COMPID	COMPANY	LOCATION
172	maruthi-suzuki	noida

CARMODEL_LIST_BAG (child db table)

MODELID	FUELTYPE	MODELNAME	TYPE	COMPANY_ID
125	diesel	Baleno	HatchBack	172
126	diesel	Brezza	HatchBack	172

Taking the special Property as Map Collection in one to many Association

=>It allows to take our choice type values as the index values.beoz keys placed in map collection will becom the index values.

```
@LazyCollection(LazyCollectionOption.EXTRA)
@OneToMany(targetEntity = CarModel.class,cascade = CascadeType.ALL,fetch = FetchType.EAGER,orphanRemoval = true )
@JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
@MapKeyColumn(name = "MAP_INDX",length = 10)
@Fetch(FetchMode.SELECT)
private Map<String,CarModel> models;
```

CARCOMPANY_MAP (taget class)

COMPID	COMPANY	LOCATION
173	maruthi-suzuki	noida

CARMODEL_MAP (child class)

MODELID	FUELTYPE	MODELNAME	TYPE	COMPANY_ID	MAP_INDX
127	diesel	Baleno	HatchBack	173	first
128	diesel	Brezza	HatchBack	173	second

ManyToOne Uni-directional Association

=>This is always child to parent non-collection type Association

=> Here child class should have parent class reference variable to make multiple objs of child class refering to single object of parent class.

=>The Annotations required are ::

@ManyToOne (JPA) :: To build the association

@JoinColumn :: To specify FK column.

@LazyToOne :: To specify HB style fetch type for many to One like "false", "proxy", "no-proxy"

@Fetch :: To specify Fetch modes like "SELECT","SUBSELECT" and JOIN

example

Employee (child) ◇——— Department (parent)

PoliticalLeader ◇——— PoliticalPary

Minster ◇——— | ChiefMinister

=>One To many assocation JPA Style default fetch type LAZY

=>Many To One assocation JPA Style default fetch type EAGER

=>Here no need of taking collection property to build the Association.. we just need to place parent class ref variable in child class.

```
//PoliticalParty.java (parent class)
package com.nt.entity;

@Table
@Entity
@Getter
@Setter
@RequiredArgsConstructor
public class PoliticalParty implements Serializable {
    @GeneratedValue
    @Id
    private long partyId;
    @NonNull
    @Column(length = 20)
    @Type(type="string")
    private String partyName;
    @NonNull
    @Column(length = 20)
    @Type(type="string")
    private String flagColors;
    @NonNull
    @Column(length = 20)
    @Type(type="string")
    private String partySymbol;

    public PoliticalParty() {
        System.out.println("PoliticalParty:: 0-param constructor");
    }

    @Override
    public String toString() {
        return "PoliticalParty [partyId=" + partyId + ", partyName=" + partyName + ", flagColors=" + flagColors
                + ", partySymbol=" + partySymbol + "]";
    }

}
```

```
//PoliticalLeader(child class)
package com.nt.entity;

@Table
@Entity
@Getter
@Setter
@RequiredArgsConstructor
public class PoliticalLeader implements Serializable {
    @Id
    @GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
    @SequenceGenerator(name="gen1",sequenceName = "AADHAR_SEQ",initialValue = 1000,allocationSize = 1)

    private long leaderAadharNo;
    @Column(length = 20)
    @Type(type="string")
    @NonNull
    private String leaderName;

    @Column(length = 15)
    @Type(type="string")
    @NonNull
    private String leaderPosition;

    @Column(length = 15)
    @Type(type="string")
    @NonNull
    private String leaderLocation;

    @ManyToOne(targetEntity = PoliticalParty.class,cascade = CascadeType.ALL,fetch = FetchType.LAZY)
    @JoinColumn(name = "PARTY_ID",referencedColumnName = "PARTYID")
    //@LazyToOne(LazyToOneOption.PROXY)  //HB style fetch type
    private PoliticalParty party;

    public PoliticalLeader() {
        System.out.println("PoliticalLeader:: 0-param constructor");
    }


    @Override
    public String toString() {
        return "PoliticalLeader [leaderAadharNo=" + leaderAadharNo + ", leaderName=" + leaderName + ", leaderPosition="
                + leaderPosition + ", leaderLocation=" + leaderLocation + "]";
    }

}
```

POLITICALPARTY (Parent table)

Table SYSTEM.POLITICALLEADER(COM2)

PARTYID	FLAGCOLORS	PARTYNAME	PARTYSYMBOL
174	saffron-Green	BJP	Lotus
175	white	AAP	Broom

POLITICALLEADER (Child db table)

LEADERAADHARNO	LEADERLOCATION	LEADERNAME	LEADERPOSITION	PARTY_ID
1000	delhi	modi	PM	174
1001	UP	yogi	CM	174
1002	delhi	Aravind	CM	175
1003	delhi	prasanth	D-CM	175

One To many and Many to One BiDirectional Assocation
==========

=> Here Parent class should have collection property to hold one more child objects and child class should have Parent class ref variable to make multiple objects of child class pointing to single Parent object.

CarCompany------◊CarModel
InsuranceCompany ---->◊InsurancePolicy
BCCI ---------->◊Cricket player
Company ------->◊Project

Example code
==========

Company.java (parent java class)

```
package com.nt.entity;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Type;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
@Table(name="COMPANY_OTM_BI")
@Entity
@Getter
@Setter
@RequiredArgsConstructor
public class Company  implements Serializable {
        @Id
        @GeneratedValue
        @Type(type="int")
    private Integer compId;

        @Column(length=20)
        @Type(type="string")
        @NonNull
    private String   compName;

        @Column(length=20)
        @Type(type="string")
        @NonNull
    private  String  addrs;

        @Column(length=20)
        @Type(type="string")
        @NonNull
    private  String type;

        @OneToMany(targetEntity =Project.class, cascade = CascadeType.ALL,fetch = FetchType.LAZY,orphanRemoval =true,mappedBy="company")
        //@JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
    private Set<Project>  projects;

        public Company() {
                System.out.println("Company:: 0-param constructor");
        }

        @Override
        public String toString() {
                return "Company [compName=" + compName + ", addrs=" + addrs + ", type=" + type + "]";
        }

}
```

**** :: mappedBy can be used only in Bi-Directional Assoication indicating the FK that is required to build the assoication is specified at other side of assocation .. not the current side... So writing @JoinCoumn having FK column details at current side is optional.. becoz it will be collected from the specified other side property..

-> we can specify the "mappedBy" at many side in OneToMany Association and we can specify any side in ManyToMany and OneToOne Associations. It is some how behaves like inverse="true".

Project.java (child java class)

```
package com.nt.entity;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;

import org.hibernate.annotations.Type;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
@Entity
@Table(name="PROJECT_OTM_BI")
@RequiredArgsConstructor
@Getter
@Setter
public class Project  implements  Serializable {
        @Id
        @SequenceGenerator(name = "gen1",sequenceName = "PROJ_ID_SEQ",allocationSize = 1,initialValue = 1000)
        @GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
        @Type(type="long")
        private Long projId;

        @Column(length=20)
        @Type(type="string")
        @NonNull
        private String projName;

        @Column(length=20)
        @Type(type="string")
        @NonNull
        private String projType;

        @Type(type="int")
        @NonNull
        private Integer teamSize;

        @ManyToOne(targetEntity = Company.class, cascade = CascadeType.ALL,fetch = FetchType.LAZY)
        @JoinColumn(name = "COMPANY_ID",referencedColumnName = "COMPID")
        private Company company;

        public Project() {
                System.out.println("Project:: 0-param constructor");
        }

        @Override
        public String toString() {
                return "Project [projId=" + projId + ", projName=" + projName + ", projType=" + projType + ", teamSize="
                        + teamSize + "]";
        }

}
```

COMPANY_OTM_BI (parent db table)

Table SYSTEM.COMPANY_OTM_BI@con2

	(pk) COMPID	ADDRS	COMPNAME	TYPE
1	176	hyd	HCL	IT
2	177	mumbai	Wiport	IT

PROJECT_OTM_BI (child db table)

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies

	(pk) PROJID	PROJNAME	PROJTYPE	TEAMSIZE	COMPANY_ID (fk)
1	1000	citiBank	banking	20	176
2	1001	OpenFx	financial	10	176
3	1003	FFW	govt	10	177
4	1002	Aadhar	portfolid	10	177

Annotation driven ManyToMany Assocation
==========
@ManyToMany :: To build the assocation
@JoinTable :: To specify 3rd table info
@JoinColumn :: To specify fk Cols info

=> Here both parent and child classes should collection type special property to build the association..

eg:: DOCTOR <----> Patient
Teacher <----> Student
TV Channel <----> Subscriber

TVChanner.java(parent class)

```
package com.nt.entity;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Type;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Entity
@Table(name="TVCHANNEL_MTOM")
@Setter
@Getter
@RequiredArgsConstructor
public class TVChannel implements Serializable{
	@Id
	@GeneratedValue
	@Type(type="int")
	private Integer channelId;

	@Type(type="string")
	@Column(length=15)
	@NonNull
	private String channelName;

	@Type(type="string")
	@Column(length=25)
	@NonNull
	private String company;

	@Type(type="string")
	@Column(length=25)
	@NonNull
	private String location;

	@ManyToMany(targetEntity = Subscriber.class,cascade = CascadeType.ALL,fetch = FetchType.LAZY,mappedBy="channels")
	private Set<Subscriber> subscribers;

	public TVChannel() {
		System.out.println("TVChannel:: 0-param constructor");
	}

	@Override
	public String toString() {
		return "TVChannel [channelId=" + channelId + ", channelName=" + channelName + ", company=" + company
			+ ", location=" + location + "]";
	}


}
```

Subscriber.java

```
package com.nt.entity;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;

import org.hibernate.annotations.Type;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Entity
@Table(name="SUBSCRIBER_MTOM")
@Setter
@Getter
@RequiredArgsConstructor
public class Subscriber implements Serializable {
	@Id
	@SequenceGenerator(name = "gen1",sequenceName = "sub_id_seq",initialValue = 1000,allocationSize = 1)
	@GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
	private Integer subscriberId;
	@Column(length=20)
	@Type(type="string")
	@NonNull
	private String subscriberName;
	@Column(length=20)
	@Type(type="string")
	@NonNull
	private String subscriberType;
	@Column(length=20)
	@Type(type="string")
	@NonNull
	private String subscriberLocation;
	@ManyToMany(targetEntity = TVChannel.class,cascade = CascadeType.ALL,fetch = FetchType.LAZY)
	@JoinTable(name="TVCHANNEL_SUBSCRIBER_MTOM",
		joinColumns = @JoinColumn(name ="SUBSCRIBER_ID",referencedColumnName = "SUBSCRIBERID" ), //owning side fk col
		inverseJoinColumns = @JoinColumn(name ="CHANNEL_ID",referencedColumnName = "CHANNELID" ) // non-owning side fk col
		)
	private Set<TVChannel> channels;

	public Subscriber() {
		System.out.println("Subscriber:: 0-param constructor");
	}

	@Override
	public String toString() {
		return "Subscriber [subscriberId=" + subscriberId + ", subscriberName=" + subscriberName + ", subscriberType="
		+ subscriberType + ", subscriberLocation=" + subscriberLocation + "]";
	}

}
```

TVCHANNEL_MTOM (table1)

	CHANNELID	CHANNELNAME	COMPANY	LOCATION
1	178	ZEETV	ZEE Entertainment	India
2	179	STARTTV	Start india	India
3	180	SONY	SONY Entertainment	India
4	181	ETV	RGC	india

SUBSCRIBER_MTOM (table2)

	SUBSCRIBERID	SUBSCRIBERLOCATION	SUBSCRIBERNAME	SUBSCRIBERTYPE
1	1000	delhi	suresh	silver
2	1001	mumbai	jani bhasha	gold
3	1002	hyd	raja rao	gold
4	1003	delhi	anil	silver
5	1004	hyd	ramesh	gold
6	1005	mumbai	Teja	gold

TVCHANNEL_SUBSCRIBER_MTOM (Join table)

	SUBSCRIBER_ID (fk)	CHANNEL_ID (fk)
1	1000	179
2	1000	178
3	1001	179
4	1002	179
5	1002	178
6	1003	180
7	1003	181
8	1004	180
9	1004	181
10	1005	181

To the bring the effect of <idbag> for getting uniqueness in join table M2M Assocaiton we need to use @CollectionId annotation.

step1) keep previous M2M app ready as seperate copy

step2) Change special property of child class to List collection and specify @CollectionId with the increment generator to generate thd unique values in join db table..

```
@ManyToMany(targetEntity = TVChannel.class,cascade = CascadeType.ALL,fetch = FetchType.LAZY)
	@JoinTable(name="CHANNEL_SUBSCRIBER_IDBAG",
		joinColumns = @JoinColumn(name ="SUBSCRIBER_ID",referencedColumnName = "SUBSCRIBERID" ), //owning side fk col
		inverseJoinColumns = @JoinColumn(name ="CHANNEL_ID",referencedColumnName = "CHANNELID" ) // non-owning side fk col
		)
	@GenericGenerator(name = "gen1",strategy = "increment")
	@CollectionId(columns = @Column(name="ch_sub_id"),
		type =@Type(type="int") ,
		generator = "gen1")
	private List<TVChannel> channels;
```

CHANNEL_SUBSCRIBER_IDBAG (join table)

	SUBSCRIBER_ID	CHANNEL_ID	CH_SUB_ID
1	1015	188	1
2	1015	189	2
3	1016	188	3
4	1016	189	4
5	1017	189	5

Annotation driven One to One Association

Can be Implemented in two ways

a) One To One FK (@OneToOne with @JoinColumn)

b) One to One PK (@OneToOne with @PrimaryKeyJoinColumn)

One To One FK

=>Use this where parent and child obj are not ready to use same id value.. they want to have two different id values..

=> Here relationship is built using FK column

=> We need to use

@OneToOne

@JoinColumn annotations

eg:: Person ——◇DrivingLicense

eg:: Person ——◇Passport

eg:: Person ——◇PanCard

```
//Person.java (parent class)
package com.nt.entity;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;
import org.hibernate.annotations.Type;
import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Getter
@Setter
@RequiredArgsConstructor
@Entity
@Table(name="PERSON_ANNO_OTO")
public class Person implements Serializeble{
    @Id
    @Type(type="int")
    @GeneratedValue
 private Integer pid;
 @NonNull
 @Column(length=15)
 @Type(type="string")
 private String pname;

 @NonNull
 @Column(length=15)
 @Type(type="string")
 private String addrs;

 public Person() {
    System.out.println("Person.0-param constructor");
}
@Override
public String toString() {
    return "Person [pid=" + pid + ", pname=" + pname + ", addrs=" + addrs + "]";
}

}
```

(parent db table)

	PID	ADDRS	PNAME
1	150	hyd	1856
2	151	vizag	ravi

child db table

	PASSPNO	COUNTRY	EXPIRYDATE	PERSON_ID
1	11000	india	10-OCT-28	190

```
//Passport.java (child class)
package com.nt.entity;

import java.io.Serializable;
import java.time.LocalDate;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;

import org.hibernate.annotations.Type;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Setter
@Getter
@RequiredArgsConstructor
@Entity
@Table(name="PASSPORT_ANNO_OTO")
public class Passport implements Serializable {
    @Id
    @Type(type="long")
    @SequenceGenerator(name = "gen1",sequenceName = "PSPT_NO_SEQ",initialValue = 10000,allocationSize = 1)
    @GeneratedValue(generator = "gen1",strategy = GenerationType.SEQUENCE)
    private Long psptNo;

    @NonNull
    @Type(type="string")
    @Column(length=20)
    private String country;

    @NonNull
    private LocalDate expiryDate;
    @OneToOne(targetEntity = Person.class,cascade = CascadeType.ALL,fetch = FetchType.LAZY,orphanRemoval = true)
    @JoinColumn(name="PERSON_ID",referencedColumnName = "PID")
    private Person personDetails;

    public Passport() {
        System.out.println("Passport: 0-param constructor");
    }

    @Override
    public String toString() {
        return "Passport [psptNo=" + psptNo + ", country=" + country + ", expiryDate=" + expiryDate + "]";
    }

}
```

One To One PK

=>Use this when both parent and child class objs are ready to use same identity values

=> Here FK column not required but .. "foriegn" generator is required to collect associated object id value as the id value of current object.

=>We need to use

@OneToOne

@PrimaryKeyJoinColumn annotaitons here

eg:: Student <-----> LibraryMembership (student id will also be used as libraryMembership id)

eg:: Student <-----> ExamHallticket (student id will also be as used HallTicket No)

eg:: Employee <-----> CateenCard (Employee id will also be used as Canteenid)

What is the difference between @JoinColumn and @PrimaryKeyJoinColumn?

Ans)=> @JoinColumn creates seperate FK column in child table ponting to PK column parent table ..

This is required to build all Associations expect One to One PK

=>@PrimaryKeyJoinColumn makes child table PK column as FK column pointing to parent table PK column i.e

no seperate FK column will be taken. This will be used in a) Table per sub class inheritence mapping

b) One To One PK Assoication

```
//Employee.java (parent class)
package com.nt.entity;
import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Type;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;

@Entity
@Table(name="EMPLOYEE_ANNO_OTO")
@Getter
@Setter
@RequiredArgsConstructor
public class Employee implements Serializable {
    @Id
    @GeneratedValue
    private Integer eno;

    @Type(type="string")
    @Column(length=20)
    @NonNull
    private String ename;

    @Type(type="string")
    @Column(length=20)
    @NonNull
    private String eadd;

    @Type(type="float")
    @NonNull
    private Float salary;

    @OneToOne(targetEntity=CanteenCard.class,cascade =CascadeType.ALL, fetch = FetchType.LAZY,orphanRemoval =true )
    @PrimaryKeyJoinColumn(name ="CID",referencedColumnName = "ENO" )
    private CanteenCard cardDetails;

    public Employee() {
        System.out.println("Employee:: 0-param constructor");
    }

    @Override
    public String toString() {
        return "Employee [eno=" + eno + ", ename=" + ename + ", eadd=" + eadd + ", salary=" + salary + "]";
    }

}
```

(parent db table)

	ENO	EADD	ENAME	SALARY
1	193	hyd	raja	50003

(child db table)

	CARDID	CANTEENNAME	CARDTYPE	LOCATION
1	150	military	gold	hyd

```
//Canteen.java (child class)
package com.nt.entity;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;

import org.hibernate.annotations.GenericGenerator;
import org.hibernate.annotations.Parameter;
import org.hibernate.annotations.Type;

import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
@Entity
@Table(name="CANTEEN_ANNO_OTO")
@Getter
@Setter
@RequiredArgsConstructor
public class CanteenCard implements Serializable {
    @Id
    @GenericGenerator(name ="gen1",strategy = "foreign",
                      parameters = @Parameter(name="property" ,value ="empDetails" ) )
    @GeneratedValue(generator="gen1")
    @Type(type="int")
    private Integer cardid;

    @Column(length=15)
    @Type(type="string")
    @NonNull
    private String canteenName;

    @Column(length=15)
    @Type(type="string")
    @NonNull
    private String location;

    @Column(length=15)
    @Type(type="string")
    @NonNull
    private String cardType;
    @OneToOne(targetEntity = Employee.class,cascade = CascadeType.ALL,
                      fetch = FetchType.LAZY,orphanRemoval = true,mappedBy = "cardDetails")
    private Employee empDetails;

    public CanteenCard() {
        System.out.println("CanteenCard:: 0-param constructor");
    }

    @Override
    public String toString() {
        return "CanteenCard [cardid=" + cardid + ", canteenName=" + canteenName + ", location=" + location
                + ", cardType=" + cardType + "]";
    }

}
```

2nd Level cache

==========

=>It is associated with SessionFactory obejct , So it is called Global cache

=> App serches for entity objects in the following order

a) L1 cache of current session obj (if not there)

b) L2 cache of Session factory obj (if not there)

c) Collects from DB s/w and keeps L2 cache and L1 cache then gives to App.

=>It is Configurable Cache i.e we can enable or disable it

=> we have mulitple providers supporting L2 cache

a) EH cache b) swarm cache c) JbossTree Cache d) OS Cache e) Tangosol cache

(popularly used) (free) (commercial)

note:: EHCache supports both InMemory and Disk Caching

uses the some amount of HDD memory if RAM cache reached to max elements. (InMemory)

L2 Caching strategies

---

read-only: caching will work for read only operation.

nonstrict-read-write: caching will work for read and write but one at a time.

read-write: caching will work for read and write, can be used simultaneously.

transactional: caching will work for transaction.

Procedure to configure EH Cache as L2/Level2 cache in Hibernate App

==========

step1) Keep any Basic App ready..

step2) add ehcahe jar files to build path or libraries..

```
// https://mvnrepository.com/artifact/org.hibernate/hibernate-ehcache
implementation group: 'org.hibernate', name: 'hibernate-ehcache', version: '5.4.25.Final'
```

step3) Develop ehcache.xml(any name) file having ehcache related info..

ehcache.xml *(com/nt/cfgs)*

```
<ehcache>
  <diskStore path="java.io.tmpdir"/>
   <defaultCache
      maxElementsInMemory="100"
      eternal="false"
      timeToIdleSeconds="10"
      timeToLiveSeconds="30"
      overflowToDisk="true"
      />
</ehcache>
```

step4) write entries in hibernate.cfg.xml file to enable and configure level2 /l2 cache..

in hibernrate.cfg.xml

```
<!-- EH Cache cfg -->
<property name="cache.use_second_level_cache">true</property>
<property name="net.sf.ehcache.configurationResourceName">com/nt/cfgs/ehcache.xml</property>
<property name="hibernate.cache.region.factory_class">org.hibernate.cache.ehcache.EhCacheRegionFactory</property>
```

step5) Specify caching strategy ....

In Mapping file before <id> tag

```
<cache usage="read-only"/>
```

In Annotation driven env.. *At Entity class level*

```
@Cache(usage = CacheConcurrencyStrategy.READ_ONLY)
```

step6) Develop the client app.

```
public class SecondLevelCacheTest {

    public static void main(String[] args) {
        Session ses=null;
        InsurancePolicy policy=null;
        //get SEssion object
        ses=HibernateUtil.getSession();
        try {
            //Load object
            policy=ses.get(InsurancePolicy.class,9001L);  //gets from DB and keeps in L2 and L1 cache
            System.out.println(policy);
            System.out.println("......................");
            policy=ses.get(InsurancePolicy.class,9001L);  //gets from  L1 cache
            System.out.println(policy);
            System.out.println("......................");
            ses.evict(policy);  // removes from L1 cache
            policy=ses.get(InsurancePolicy.class,9001L);  //gets from  L2 cache and keeps L1 cache
            System.out.println(policy);
            System.out.println("......................");
            ses.clear(); // removes from L1 cache
            try {
                Thread.sleep(20000);  // removes from L2 cache (idletime out)
            }
            catch(Exception e) {
                e.printStackTrace();
            }
            policy=ses.get(InsurancePolicy.class,9001L);   //gets from DB and keeps in L2 and L1 cache
            System.out.println(policy);
        }//try
        catch(HibernateException he) {
            he.printStackTrace();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        finally {
            HibernateUtil.closeSessionFactory();
        }
    }
}
```

To control 2nd Level cache

==========

```
Cache cache=factory.getCache();
```

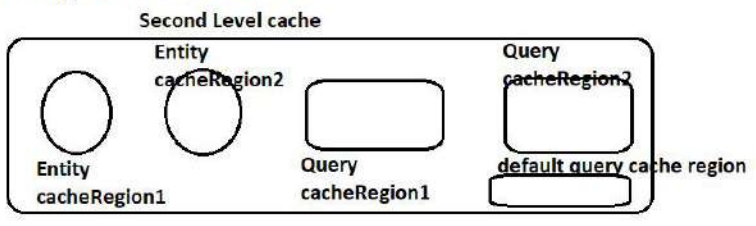
and invoke various methods on cache obj like

```
evict(Class cls) : void - Cache
evict(Class cls, Object primaryKey) : void - Cache
evictAll() : void - Cache
evictAllRegions() : void - Cache
evictCollection(String role, Serielizable ownerIdentifier) : void -
evictCollectionData() : void - Cache
evictCollectionData(String role) : void - Cache
evictCollectionData(String role, Serializable ownerIdentifier) : vc
evictCollectionRegion(String role) : void - Cache
evictCollectionRegions() : void - Cache
evictDefaultQueryRegion() : void - Cache
evictEntity(Class entityClass, Serializable identifier) : void - Cach
evictEntity(String entityName, Serializable identifier) : void - Ca
evictEntityData() : void - Cache
evictEntityData(Class entityClass) : void - Cache
evictEntityData(String entityName) : void - Cache
```

Query Cache

==========

=>Query Cache is part of SecondLevel cache to hold the entity objects related to HQL(or)JPQL Queries/Native SQL queries/ Criteria API logics. i.e it must be enabled along with Second Level cache.



To enable query cache

==========

step1) keep any app ready that is having second level cache

step2) enable query cache in hibernate cfg file

```
<property name="hibernate.cache.use_query_cache">true</property>
```

step3) Write the following code .. in the client App..

=>HQL quries/ native SQL queries / QBC logic queries will not placed in L1 and L2 caches by default.. they can be placed only in query caches.

```
public class QueryCacheTest {

    public static void main(String[] args) {
        //get SEssionFactory object
        SessionFactory factory=HibernateUtil.getSessionFactory();
        Session ses=factory.openSession();
        try {
            Query query=ses.createQuery("from InsurancePolicy");
            query.setCacheable(true);
            query.setCacheRegion("reg1");
            List<InsurancePolicy> list=query.getResultList(); //collects from DB and keeps in query cache of LI2 cache
            list.forEach(policy->{
                System.out.println(policy);
            });
            System.out.println("--------------------------");
             list=query.getResultList();  //collects from query cache .
            list.forEach(policy->{
                System.out.println(policy);
            });
            Cache cache=factory.getCache();
            cache.evictQueryRegion("reg1");   //removes the entties from query cache region "reg1"
            System.out.println("--------------------------");
             list=query.getResultList();  //collects from DB and keeps in query cache of LI2 cache
            list.forEach(policy->{
                System.out.println(policy);
            });


        }//try
        catch(HibernateException he) {
            he.printStackTrace();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        finally {
            HibernateUtil.closeSessionFactory();
        }
    }//main
}//class
```

How many caches are there in hibernate?

Ans) a) L1/session cache

b) L2 /Session factory cache

(Query cache is part of L2 cache)

Locking in Hibernate
==========

=>If multiple Apps or threads simultaenously accessing and manipulating records then there is a possibility of getting concurrency problems.

=>To avoid those problems , take the support of Locking of record in Hibernate

HB supports two types of lockings

a) Optimistic Locking

-->Allows second thread/app simultaenuolsy to access modify the record then 1 thread /app notices the modification and throws Exception.

=> If we enable versoning on the app , automatically this locking will be applied

b) Pessimistic Locking

-->Allows first App /thread itself to lock the record.. So if second App/ trhead tries access and modify the record simultaenously then we will get exception.

=> we need to call ses.get(-,-,-) having LockMode.UPGRADE_NOWAIT as the third argument value.

a) Optimistic Locking Example

Client App1
==========

```
package com.nt.test;

import java.util.List;

import org.hibernate.Cache;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.query.Query;

import com.nt.entity.InsurancePolicy;
import com.nt.utility.HibernateUtil;

public class LockingAppTest1 {

    public static void main(String[] args) {
        // get Session object
        Session ses=HibernateUtil.getSession();
        Transaction tx=null;
        boolean flag=false;
        try {
            tx=ses.beginTransaction();
         //Load object
            InsurancePolicy policy=ses.get(InsurancePolicy.class,9001L);
            System.out.println(policy);
             Thread.sleep(30000); // In This sleep time , run the second App
            //modify the object
             policy.setTenure(30);
             flag=true;
        }//try
        catch(HibernateException he) {
             flag=false;
            he.printStackTrace();
        }
        catch(Exception e) {
            flag=false;
            e.printStackTrace();
        }
        finally {
            //perform TxMgmt
            if(flag) {
                tx.commit();
                System.out.println("Object updated");
            }
            else {
                tx.rollback();
                System.out.println("Object not updated");
            }
            HibernateUtil.closeSession(ses);
            HibernateUtil.closeSessionFactory();
        }
    }//main
}//class
```

ClientApp2
==========

```
package com.nt.test;

import java.util.List;

import org.hibernate.Cache;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.query.Query;

import com.nt.entity.InsurancePolicy;
import com.nt.utility.HibernateUtil;

public class LockingAppTest2 {

    public static void main(String[] args) {
        // get Session object
        Session ses=HibernateUtil.getSession();
        Transaction tx=null;
        boolean flag=false;
        try {
            tx=ses.beginTransaction();
         //Load object
            InsurancePolicy policy=ses.get(InsurancePolicy.class,9001L);
            System.out.println(policy);
            //modify the object
             policy.setTenure(24);
             flag=true;
        }//try
        catch(HibernateException he) {
             flag=false;
            he.printStackTrace();
        }
        catch(Exception e) {
            flag=false;
            e.printStackTrace();
        }
        finally {
            //perform TxMgmt
            if(flag) {
                tx.commit();
                System.out.println("Object updated");
            }
            else {
                tx.rollback();
                System.out.println("Object not updated");
            }
            HibernateUtil.closeSession(ses);
            HibernateUtil.closeSessionFactory();
        }
    }//main
}//class
```

How to execute
==========

1) Run first App (after loading record 9001 having version "1" it goes to sleep state)

2) Run Second (it loads records 9001 and modifies the record , in that process version value changes "2"

Once the first App completes "sleep" state it reliases that the version value is changed to "2" from "1".. This makes Object as StaleState Object So StaleStateObjectException or OptimistickLockException will be thrown.

b) Pessimistic Locking Example App (No Versioning) is required.

ClientApp1
==========

```
public class LockingAppTest1 {

    public static void main(String[] args) {
        // get Session object
        Session ses=HibernateUtil.getSession();
        Transaction tx=null;
        boolean flag=false;
        try {
            tx=ses.beginTransaction();
         //Load object
            InsurancePolicy policy=ses.get(InsurancePolicy.class,9001L,LockMode.UPGRADE_NOWAIT);
            System.out.println(policy);
             Thread.sleep(30000); // In This sleep time , run the second App
            //modify the object
             policy.setTenure(30);
             flag=true;
        }//try
        catch(HibernateException he) {
             flag=false;
            he.printStackTrace();
        }
        catch(Exception e) {
            flag=false;
            e.printStackTrace();
        }
        finally {
            //perform TxMgmt
            if(flag) {
                tx.commit();
                System.out.println("Object updated");
            }
            else {
                tx.rollback();
                System.out.println("Object not updated");
            }
            HibernateUtil.closeSession(ses);
            HibernateUtil.closeSessionFactory();
        }
    }//main
}//class
```

ClientApp2

```
public class LockingAppTest2 {

    public static void main(String[] args) {
        // get Session object
        Session ses=HibernateUtil.getSession();
        Transaction tx=null;
        boolean flag=false;
        try {
            tx=ses.beginTransaction();
         //Load object
            InsurancePolicy policy=ses.get(InsurancePolicy.class,9001L,LockMode.UPGRADE_NOWAIT);
            System.out.println(policy);
            //modify the object
             policy.setTenure(24);
             flag=true;
        }//try
        catch(HibernateException he) {
             flag=false;
            he.printStackTrace();
        }
        catch(Exception e) {
            flag=false;
            e.printStackTrace();
        }
        finally {
            //perform TxMgmt
            if(flag) {
                tx.commit();
                System.out.println("Object updated");
            }
            else {
                tx.rollback();
                System.out.println("Object not updated");
            }
            HibernateUtil.closeSession(ses);
            HibernateUtil.closeSessionFactory();
        }
    }//main
}//class
```

How to execute

1) run the First App (loads 9001 record , gets lock on the record and goes sleep state)

2) run the second app (Loads 9001 record ,but noticates the lock is applied on that record So exception will be raised ..)

once the first App completes its sleep operation then it will complete other non-select persistence operation on the record and release the lock.