# Hibernate Class

(Faculty Omkar)

**Created by AZAM AMIR REZA (**azam20104u@gmail.com**)**

10-Dec-15:

**Frame Work**

* Frame work is a solution to commonly occurring problems
* Frame work can be categories into open source custom framework

**Example of open source Framework:**

1. Spring
2. Hibernate
3. JSF

Etc.

**Hibernate:**

* It is an open source ORM framework available in java
* Hibernate is used to persist data with relational database system
* Hibernate is an implementation of JTA(Java Transaction API)

Note: it is used to connect database and is not a replacement for JDBC

JDBC

DB

Java

JTA

**DTO :**( Data Transfer Object)

1. DTO is used to transfer data between layers while creating enterprise application
2. DTO preserve DP(Design Pattern)
3. DTO has 5 rules to be followed:
4. Implementation of Serializable
5. Properties must private
6. Access it throw getter and setter method
7. And non-final members
8. Define default constructor

Example

Class tree implements Serializable

Private String color;

Private String name;

Public tree(){}

Getr and seter()

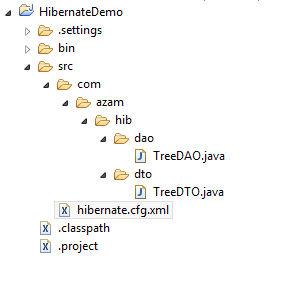
Public non-final variable

Note: A DTO should not have any methods to perform logic.

11-Dec-15:

Steps to save operation using Hibernate:

1. Create a java project



Persistence logic

1. Location is src  
   2. Create a lib folder and placed hibernate jar building the class path.
2. Create two folders dao and dto
3. Create hibernate configuration file by name hibernate.cfg.xml inside src folder.
4. In that xml file

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

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>tiger</property>

<property name=*"connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/jspider</property>

</session-factory>

</hibernate-configuration>

1. Write hibernate Entity class

**package** com.azam.hib.dto;

**import** java.io.Serializable;

@Entity

@Table(name="jspider")

**public** **class** TreeDTO **implements** Serializable{

@Id

@Column(name="u\_id")

**private** **int** uid;

@Column(name="tree\_name")

**private** String name;

@Column(name="tree\_color")

**private** String color;

@Column(name="life\_spane")

**private** **int** lifeSpan;

//generate setters and getters

**public** TreeDTO(){

}

**public** **int** getUid() {

**return** uid;

}

**public** **void** setUid(**int** uid) {

**this**.uid = uid;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **int** getLifeSpan() {

**return** lifeSpan;

}

**public** **void** setLifeSpan(**int** lifeSpan) {

**this**.lifeSpan = lifeSpan;

}}

1. Write DAO and save performance

**package** com.azam.hib.dao;

**import** com.azam.hib.dto.TreeDTO;

**public** **class** TreeDAO {

**public** **static** **void** main(String[] args) {

TreeDTO td=**new** TreeDTO();

td.setColor("green");

td.setLifeSpan(20);

td.setUid(1);

td.setName("Mango");

//component 1

Configuration cfg=**new** Configuration();

cfg.configure();

cfg.addAnnotatedClass(TreeDTO.**class**);

//component 2

SessionFactory factory=cfg.buildSessionFactory();

//component 3

Session session=factory.openSession();

//component 3, sub-component 1

Transaction tx=session.beginTransaction();

session.save(treeDTO);//table name

tx.commit();

session.close();

factory.close();

}

}

Hibernate🡪connect DB

TreeDTO🡪Entity Mapping file

TreeDAO🡪persistance logic

Om.bn@outlook.com

14-Dec-15:

**O R M**

**(Object Relational Mapping/Model):**

* Is a guideline to map object model with relational model
* ORM allows objects to the persisted into data base without converting object into values and values to object
* ORM has identified 5 problems between object relational models.

Note: Hibernate is an ORM tools and ibates, toplink they are also ORM tools

5 problems:

1. Problem of Relations
2. Problem of Inheritance
3. Problem of identify
4. Problem of navigation
5. Problem of Grains

**Hibernate Configuration:**

1. Hibernate config can be done in 3 ways
2. Hibernate config is used to configure are database with application

org.hibernate.cfg.Configuration:

* Configuration component is the basic object in hibernate system
* This component is used to load or parse configuration details to configure a data base with application.
* Configure() method is used to parse hibernate.cfg.xml from class path.

Configuration cfg=**new** Configuration();

cfg.configure(“Mysql.cfg.xml”);

15-Dec-15:

Hibernate Mapping:

* Hibernate mapping is used to map a java class with a data base table
* There are two ways for mapping

1. Xml
2. @notation or java configuration

Configuration is required to build SessionFactory.

**SessionFactory:**

* SessionFactory is the main component of hibernate System
* SessionFactory holds connection and mapping information which is required for application.
* SessionFactory is an abstraction to build session objects
* SessionFactroy has to be created only once for an application or data base
* SessionFactory is immutable
* SessionFactory is thread safe

Session:

* Session is the component which is required to perform CURD operation
* Session can be created n times depending on the application
* Session is used to obtain transaction criteria and query component
* Session is mutable
* Session is not thread safe.

Difference between session and factory:

**Transaction:**

* Transaction is a component which is used to manipulate TCL(commit and rollback)
* Transaction is required only for right operations (create update delete)

Note: transaction is not required for read operation

16-Dec-15:

**Architecture:**

Hibernate config

configure(); buildSessionFactory(); openSession();

Configuration

Session

Session factory

Hibernate mapping

Criteria

Transact

Query

beginTransaction();

createQuery(qry);

**update operation:**

BarDTO br=session.load(BarDTO.class, new Integer(2));

Br.setContactNo(12334);

session.update(br);

tx.commit();

session.close();

factory.close();

Step:

1. Load
2. Change or set
3. Update

Delete:

Load first and then session.delete(br); no need to change or set () method call

@notation note: go for this url docs.jboss.org/hibernate/jpa/2.1/api

**Generator:**

* Generator is a strategy which generates primary key or id for the entity

Syntax,

@Id

@GenericGenerator(name=”auto”,strategy=”increment”)

@GeneratedValue(generator=”auto”)

* Generator can be specified only with at id
* Generator is dependent on the data base

Example of generator Increment, sequence, identity, forIn

Generator strategy search google

**Mapping Entity’s:**

In the xml file we can add the class for mapping

<mapping class=”com.azam.hib.dto.BarDTO”/>

<mapping class=”com.azam.hib.dto.TreeDTO”/>

* Mapping tag should be used only after property tag

Secret property <property name=”show\_sql”>true</property> it helps to show the query in the console.

One session but operation with two table

Ex,

BarDTO br=session.load(BarDTO.class, new Integer(1));

String name=br.getName();

If(“Yes Bar”.equals(name)){

//save a tree

TreeDTO coco=new TreeDTO();

Coco.setName(“Coconut”);

Coco.setLifeSpane(100);

Coco.setColor(“Gray”);

Session.save(coco);

Home Task;

Public class ComputerDTO{

Private int cid;

Private String brandName;

Private String noOfProcs;

Private String type;

Private Boolean working;

}

1. Design hibernate for above class
2. Perform crud operations
3. Use increment generator

17-Dec-15:

Note: Assign is the default generator

**Handling Session and Transaction:**

**try** {

session.save(cd);

tx.commit();

} **catch** (HibernateException e) {

e.printStackTrace();

tx.rollback();

}**finally** {

session.close();

factory.close();

}

Note:

Transaction.commit must be last line in try block

Transaction.rollback should be in catch block

Session.close() must be in finally block

21-Dec-15:

**Hibernate Entity:**

It’s have three states

1. Transient
2. Persistence
3. Detached

An entity is a java class which is map to a data base table

1. **Transient State:**

* This is the state entity where it is not associated with session
* In this state entity will not have primary key

1. **Persistence State:**

* This is the state entity where it is associated with session
* In this state entity will have primary key

1. **Detached state:**

* This is the state entity where it was associated with session
* In this state entity may have primary key or may not

save(obj) delete(obj);

Detached

Persistent

Transient

persist(obj) commit();

SaveORpudate(obj)

commit();

Note: hibernate manages two level of catch memory

1. Associated with session and it can’t be managed by application
2. Second level associated with session factory and application can manage this

**Important method in the session:**

1. save(obj)
2. update(obj);
3. delete(obj)
4. load(.class,pk);
5. get(.class,pk);
6. saveOrUpdate(obj);
7. clear();
8. evict(obj);
9. flush();
10. persist(obj);
11. beginTransaction();

SessionFactory Transaction

commit();

rollback();

openSession();

close();

Configuration

Configure();

Configure(String);

addAnnotatedClass(.class)

buildSessionFactory();

**HQL:**

* Hibernate query language is a way writing query based on object model
* Query’s return HQL is database independent
* Syntax of writing an HQL is same as SQL
* There are two steps to work with HQL

1. Prepared the query
2. Process the query

22-Dec-15:

**try** {

//step 1

String hql="From FamilyDTO where name=’Patil’";

Query query=session.createQuery(hql);

//step 2

FamilyDTO ctdto=( FamilyDTO)query.uniqueResult();

System.***out***.println(ctdto.getname());

SQL = there mendatary selection part (selet\* ) mandatary entity part (from table\_name) optional is (where or condition )

HQL = we need only entity part like from table\_name

* To prepare query we need session object

Query query=session.createQuery(query);

* To process the query we need query object

ComputerDTO ctdto=(ComputerDTO)query.uniqueResult();

HQL support read update and delete operations only

23-Dec-15:

**When single type of object you want to retrieve:**

**try** {

//step 1

String hql="select cid From ComputerDTO where bname='Hp'";

Query qry=session.createQuery(hql);

//step 2

Integer cdt=(Integer) qry.uniqueResult();

System.***out***.println(cdt);

//session.save(cd);

tx.commit();

}

When multiple or different type of Object you want to retrieve:

**try** {

//step 1

String hql="select type,noOfProccess From ComputerDTO where bname='Hp'";

Query qry=session.createQuery(hql);

//step 2

Object[] cdt=(Object[]) qry.uniqueResult();

System.***out***.println(cdt[0]);

System.***out***.println(cdt[1]);

//session.save(cd);

tx.commit();

}

Processing multiple rows

Query qry=session.createQuery(“From ComputerDTO”);

List list=qry.list();

Syso(list.size());

Note: HQL doesn’t support select\*

String hql="From ComputerDTO";

Query qry=session.createQuery(hql);

List<ComputerDTO> list=qry.list();

**for**(ComputerDTO com: list){

System.***out***.println(com.getBname());

}

Ex,

String hql="select bname From ComputerDTO as sdto";

Query qry=session.createQuery(hql);

List<String> list=qry.list();

**for**(String com: list){

System.***out***.println(com);

}

Ex,

String hql="select type,noOfProccess From ComputerDTO";

Query qry=session.createQuery(hql);

List<Object[]> list=qry.list();

**for**(Object[] com: list){

System.***out***.println(com[0]);

System.***out***.println(com[1]);

}

Ex,

String hql="update ComputerDTO set bname='apple' where cid=2";

Query qry=session.createQuery(hql);

qry.executeUpdate();

Ex,

String hql="update ComputerDTO set bname=?"+ "where cid=?";

Query qry=session.createQuery(hql);

qry.setString(0, "azam");

qry.setInteger(1, 2);

qry.executeUpdate();

Set parameter:

Ex, String hql="update ComputerDTO set bname=:nam"+ "where cid=:id";

Query qry=session.createQuery(hql);

qry.setParameter("nam", "kzim");

qry.setParameter("id", 2);

qry.executeUpdate();

25-Dec-15:

**Problem of Relation:**

Has a relationship:

* Has a relation is an object is related to another object
* Has a can be categories into 4 types based on number and directions

1. One to one
2. One to many
3. Many to one
4. Many to many

Ex, relate

// create cricket

CricketDTO cdto=**new** CricketDTO();

cdto.setCid(1);

cdto.setNoOfMatch(30);

cdto.setNoOfTeam(20);

cdto.setType("One Day");

//create WC

WCDTO wcdto=**new** WCDTO();

wcdto.setHost("null");

//relate

cdto.setWc(wcdto);

1. **One to one:**

* In this relation an object is refer to another object exactly once.
* Mapping this into database is considered as problem

28-Dec-15:

**Cascade:**

* Cascade can be used problem with relations
* Case cade is hibernate for what operation it has consider associated entity
* Cascade by default is none
* Cascade have values

ALL, PERSIST, MERGE, REMOVE, REFRESH, DETACH;

Note:

* hibernate implements API standard
* JPA Standard API which Is used ORM

**Lazy loading:**

* Hibernate can fetch data only when used. Configure this we need to use FetchType
* Fetchtype have tow values a) EAGR b) LAZY

Example in the GunDTO class

@OneToMany(cascade=CascadeType.***ALL***,fetch=FetchType.***LAZY***)

By default fetch type is LAZY

Java equality contract search

**Problem of Identity:**

* Problem of identity occurs when working with one to many and many to many
* To solve this problem we have to override equals and hashCode();

29-Dec-15:

**Key word:**

1. ORM Problems
2. Hibernate config
3. Components OR Hibernate Architecture
4. CRUD operations
5. Generators
6. JPA
7. Four relations
8. HQL
9. 2 Levels and catch memory
10. Criteria.