HIBERNATE

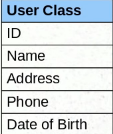
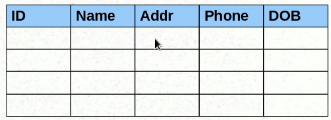
An ORM tool

Used in the data layer of applications

Implements JPA(java persistent API)

The problem that hibernate is trying to solve

Ex.

for now let’s have these many field has user\_class

To save the same information we will create an table and store all the object. With the help of JDBC, similarly to get the information we will use select.

Db.

The problem is we have object in java but not in database, so to convert it we right extra boiler plate code to covert java object to database rows and rows to object.

Pain point.

Mapping member variables to columns

Mapping relationship(user object to user table similarly for all)

Handling data types(example of handling Boolean)

Managing changes to object state(manual changes required)

L2-setup

L3- Using Hibernate

Saving the data without hibernate

Jdbc data configuration

The model object

Service method to create the model object

Database design

DAO method to save the object using sql queries

**The Hibernate Way**

JDBC database configuration- hibernate configuration

The model object – Annotations

Service method to create the model object – use the Hibernate API

Database design – Not needed!

DAO method to save the object using sql queries – not needed!

Steps:

Create the hibernate.cfg.xml -> include all the configuration ->

Ex



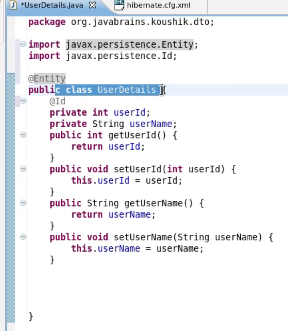
Dialect : is the language which will be used to connect to database by Hibernate

(org.hibernate.dialect.h2dialect)

Show\_sql=true(to generate the querries)

Lect 3 part 2—

Step2- Writing model Class with Annotations



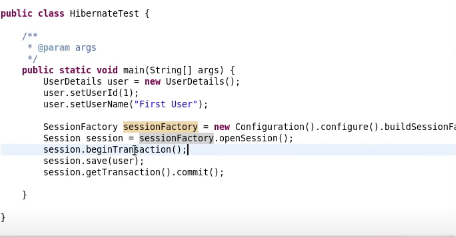
Id states primary key, how ever this class has to be configured in hibernate.cfg.xml file.

3rd step:

Saving objects using Hibernate APIs

Using the Hibernate API

1. Create a session factory (with the help of configuration)
2. Create a session from the session factory
3. Use the session to save model object



Tutorial-04: hbm2ddl Configuration and Name Annotations

Hbm2ddl.auto ->create: drop and create new table everytime

->update: it update in the previous table

Tutorial 05-> more Annotations

@table

@id

@column

Etc.

@Basic(hibernate persist and create other tables but it is not required unitl unless we want to

Specify as fetch or something in @basic

@Transient : exclude it from jpa/hibernate for heavy lifting

Date-> this will return date with time in se and milisecs

@Temporal(TemporalType.date): it will just save the date not time

@Lob : for large object( so it convert it into clob)

07)

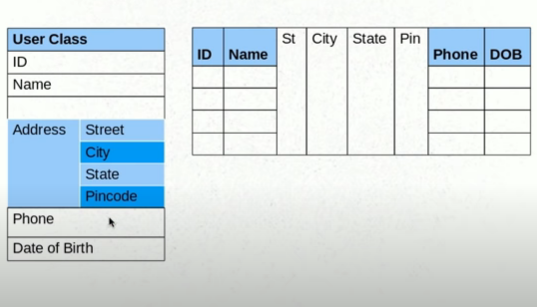
Primary keys: natural keys-> hibernate automatically generates the id.

@Id@GenratedValue(strategy=GenrationTYpe.AUTO) :it will generate the value automatically

It has strategy. And by default it is auto and other options are identiy,sequence,table,class

Surrogate key : it acts as primary key

8) Value types and Embedding Objects



Address object is a value object: there is no point of keeping address in a separate table

@Embeded : annotation will be used at variable

@Embedable : at class level

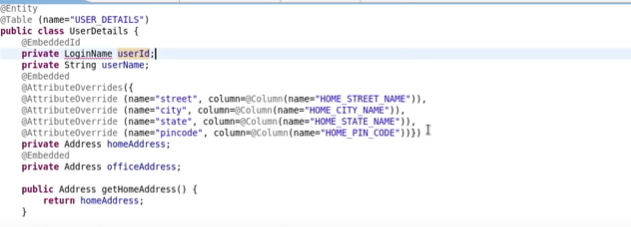
So one table will store all information.

9) Attribute Overrides and Embedded Object keys

@Column(name=”column\_name”)

@AttributeOverride(name = “stree”, column = @Column(name=”home\_stree”)

for multiple override refer below



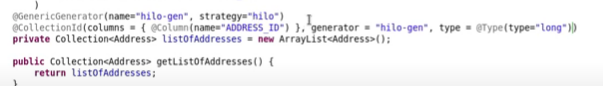
@EmbededId: this annotation is used for embedded primary key

10.) Saving collection

@ElementCllection

Private Set<Address> listOfAddresses = new HashSet();

11) Configuring Collections and Adding Keys



These feature is provided by hibernate, if you see the imports it is getting imported from hibernate

It will generate the

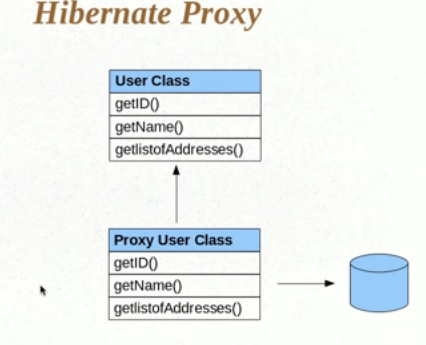
12 - Proxy Objects and Eager and Lazy Fetch Types

Say a user has travelled 1000 places or he has 1000 addresses ,

And it is stored in a list so when you do a get on list of address it will return entire list

So you do a lazy call on object to get one level of data nad eager when you want to get entire data

Hibernate uses proxy objects to achieve this.



13 – one to one

Entity inside another entity (while design database table we will look at these relationship)

@OneToOne

@JoinColumnn(name=”vehicle ID”

Example: one user has one vehicle

14 - one to many

Example: one user has many vehicle

@OneToMany

@joincolumn(name=”user\_id”)

@InverseJoinColumn

@tableTable(name=”user\_vehicle”, joinColumns=@JoinColumn(name=”user\_id”) ,inverseJoinColumn=@JoinColumn(name=”Vehivle\_id”))

Private List<vehicle> vehicle;

@ManyToOne

: to get the vehicle’s owner

Vehicle to user (it is reverse relationship)

Vehicle.user

Private users user;

15)Mapped and many to many

Mapped by

@OneTomany(mappedBy=”user”)

Many to Many

Example: one user can rent many vehicle and one vehicle can be rented by many users

16. Cascade type and other things

If there is no relationship

@Manytoone

@NotFound(action=NotFouncAction.IGNORE) /provided by HIBERNATE not jpa

Hibernate collections

Bag semantic : similar like a bag we insert and extract anyway. –List/ArrayList

Bag semantic with ID- List /Array List

@OneToMany(cascade=cascadetype.PERSIST)

Cascade on particular object. For example when saving user cascade the vehicle as well)

Persist this collection when you are saving this entity.

17) implementing inheritance

18) implementing inheritance single table strategy(not normalized)

@DiscriminatorColumn ? (to discriminate the object type , to identify record of project): since all the objects information I stored in one class

19) implementing inheritance with table per class strategy(more normalized)

It is not required to keep descriminator if we have table for each class

@Inheritance(strategy=inheritanceType.Table\_per\_Class)

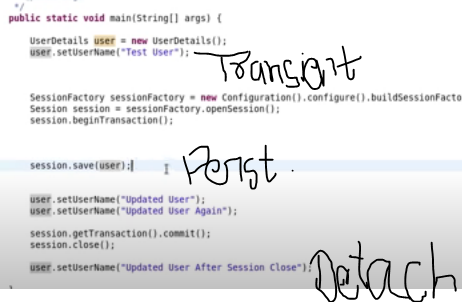
20) Implementing Inheritance with Joined strategy(in above still few rows being repeated)

21) crud operation

22) Transient,persistent and Detached Objects

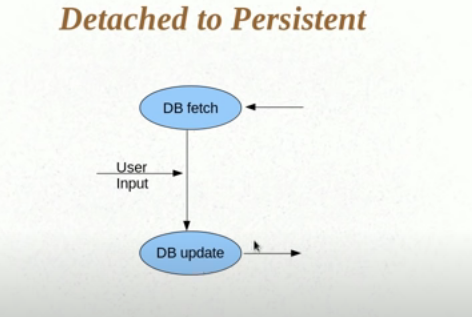
Before handling to hibernate it is a transient object , after hibernate takes that object this

Becomes Persistent and after session gets close. The objects becomes detached object



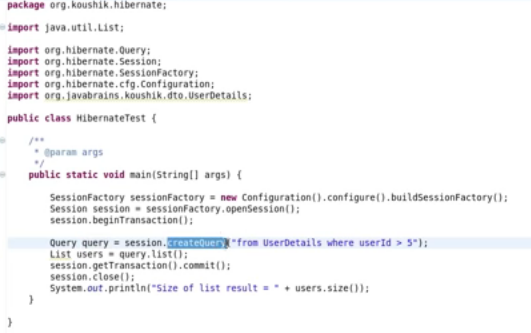
23) Understanding state changes

24) Persisting Detached objects



25)HQL and the Query Object

->hibernate query language : it is similar to sql but here we talk about object/classes rather than tables, and instead of column name use field name , query will start form (from) as shown in fig.



26) Select and Pagination in HQL

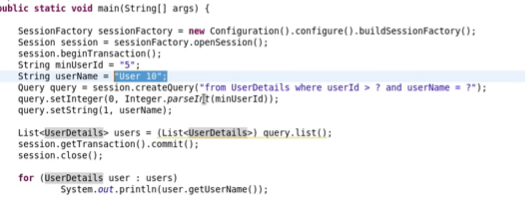
Show only required/requested data.

27) Understanding parameter binding and SQL Injection

Pass the parameter instead of doing concatenation

?

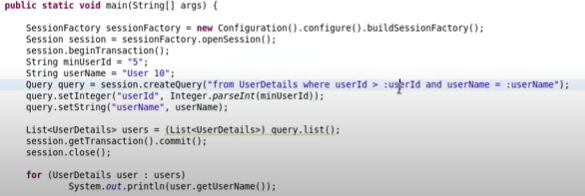
Ex.



Instead of using question mark we can also pass the variable name with colon and name

: variable name

Advantage of this is we did not need to track the position of parameter



Hence it is always advisable to use binding parameter instead of appending the parameter theat may

Lead to sql injection.

28- Named Queries

@NamedQuery works with hql where as @NamedNativeQery uses hql

Named queries is a way to consolidate all the query at one place.

Named query is used in @Entity with the @NamedQuery(name=”userDetails.byId”, query =”from userDetails where userId = ?”) with two parameter

Query query = session.getNamedQuery(“userDetails.byId”)

Query.setInteger(0,2)



29-Criteria API

As querries gets bigger it is a problem to solve this we have a Criteria API(repolacement o query)

//create the criteria nad apply the restriction

Criteria criteria = session.createCriteria(userDetails.class);

//return all the username which equals to user 10

Criteria.add(Restrictions.eq(“username”,”user 10”);

List<userDetails> users = (List<UserDetails>) criteria.list();

session.getTransactional().commit();

session.close()

Tutorial 30- Understanding Restrictions

We have all other restrictions like equal, less than, greater than.

We can append as many as criteria we want with .add



We can use all the conditions here , by default is use the and conjunction, example of different criteria



31-Projections and Query By Example(feature of criteria API)

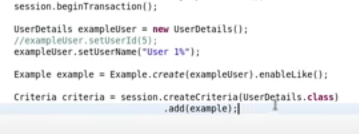
1. Use of projections is to implement the grouping functions like max,count ,order by etc.



Once you set a projection on property the end result won’t be list.

1. Querying by example

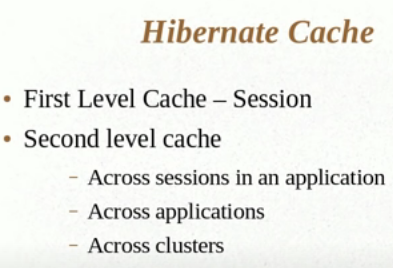
Let’s I have an object which has 10 fields and we want to fetch the details based on half of the fields ,to write this much criteria is pain so better version of this is use query be example



32- Caching in Hibernate

As soon as one session closes the cache data also get’s destroyed- level 1 caching.

If session is open it won’t fetch the data again from database since all the details are still available in cache memory.



33- Configuring Second Level Cache.

Change the configuration of cache to enable which is disabled by default.

Eh-cache, os cache,

Configure the eh cache(provider name) and tell the hibernate to use second level cache.



Download the eh cache and add the jar.



34- Using Query Cache

Add one more config line in xml as query cache and add

In code addd query.setCacheable(true) : this not only cache the query but also enables the second level cache…

We have to use it very carefully because unnecessary caching will make system inefficient