# **Hibernate**

All the documentation Is available here

Reference

Download the hibernate from internet and browse as per below path

E:\study softwares\hibernate-search-5.8.0.Final-dist\hibernate-search-5.8.0.Final\docs\reference\en-US\html\_single

### Links / projects for your reference

are available in below downloaded code

<https://hibernate.org/orm/releases/5.6/>

|  |  |
| --- | --- |
| main hibernate site | <https://hibernate.org/orm/documentation> |
| Full documentation | <https://docs.jboss.org/hibernate/orm/5.6/userguide/html_single/Hibernate_User_Guide.html> |
| spring data jpa | <https://docs.spring.io/spring-data/jpa/reference/index.html> |
| all spring properties found here | <https://docs.spring.io/spring-boot/appendix/application-properties/index.html> |
| Trainer git hub link – all his hib proj available here | <https://github.com/bharaththippireddy> |
|  |  |

All hibernate properties are available here

<https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/session-configuration.html>

<https://docs.jboss.org/hibernate/orm/current/quickstart/html_single/>

E:\study softwares\hibernate-search-5.8.0.Final-dist\hibernate-search-5.8.0.Final\project

## Inspiration

If they struggled and kept their blood in creating the software can’t we just read and understand

Main jar file is hibernate-core-5.2.11.Final

and jpa annotations are available in javax.persistence-api

### how to get most out of this

Read source code, read documents, read books, udemy course, read in internet misc sites, YouTube videos , Don’t study for interview, study for the survival of the job

### Hibernate points

1. If u don’t give any cfg file, by default hibernate looks for hibernate.cfg.xml from the class path

Sample hibernate.cfg.xml can be found in the hibernate software which you have downloaded from internet, you will have a folder called project in that u search for the file called hibernate.cfg.xml you will find sample projects

Features of hibernate

Lazy loading, it will create tables automatically, we have 12+ generators to generate primary key ,

gives callback interfaces to execute jdbc code, from hib 3.x we can call stored procedures

1. How does hibernate or any other ORM framework performs synchronization between objects in cache and database table rows?

It uses primary key column, based on that only it will fetch the records from database rows and based on that only it will update the table rows based on the modifications done on the objects

1. Hibernate **internally uses JDBC’s prepared statement** to execute the sql query as precompiled sql query, so that time will be saved to avoid parsing the query every time and because prepared statement solves the problem of sql injection.

## **Hibernate recommendation’s**

1. Make all entity classes implementing serializable
2. <property name=”connection.driver\_class” name=”oracle.jdbc.driver.OracleDriver”> this property is optional because java8 internally uses jdbc 4 which supports auto loading of jdbc driver so here after in jdbc also no need to do

Class.forname(‘oracle.jdbc……..)

1. Make sure all non select operations are running in transactions so that we can do commit and rollback , session.beginTRansaction() internally calls con.setAutoCommit(false);
2. Don’t hit database multiple times
3. Use session.update(p); only if u want to update entire record and this won’t perform

Any insert operation if record is not there instead it will throw exceptions for partial modifications pls prefer ses.load() and then modify some fields and do tx.commit() or ses.update()

### Hibernate important points

Session vs session Factory

Eager loading in hibernate

Design patterns in hibernate

Session.get() eager loading vs ses.load() lazy loading

In xml approach, If entity class name matches table name, then no need to mention table name in xml file

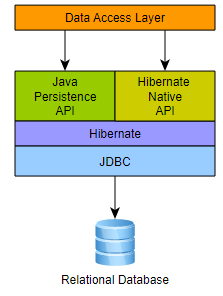
## **Hibernate concepts**

### What is the main advantage of hibernate

We don’t need to write sql query, it will only identify the changes made to that object and it will only create the insert or update query, that to after loading that object from db if u do any modifications and if u just commit that transaction (ses.update() is also not required) it will generate the update query

Simple point, when we put a property called show\_sql=true we can see sql that is being printed on the console. As simple as that sql can be executed only by jdbc. So hibernate uses jdbc only.

1. (Auto schema generation) It can create tables , it can create select query , insert ,update, delete query jdbc is database software dependent whereas hibernate is not,
2. Versioning, timestamping (automatically updates last modified by) , Global transactions ,
3. Supports lazy loading (it will hit the database lazily using ses.get()), we can call plsql procedures from hib 3.x, 12+ generators, callback interfaces to execute jdbc code.
4. Dynamic insertion and dynamic updation



Just add hibernate-core jar file remaining jar files will come automatically

* Hibernate internally uses jdbc only (JDBC prepared statement for select queries)

0.Eager vs lazy loading /ses.get() vs ses.load()

ses.**l**oad() for **l**azy loading internally uses proxy design pattern,

session.**ge**t() for **e**a**g**er loading

Whatever you do ses.get() or ses.load() both will query the database and keeps record in session level or l1 cache

ses.get() – only eager loading whether u use the data or not it will hit the database and fetch the data as and when this statement is executed- here no proxy class is created

because it is eager loading , when data is not there this method will return null

suitable when ur checking record is not there

use ses.get() if there is a guarantee that record is used immediately

ses.load() – this can do both eager loading(lazy=false or when lazy=true +entity class is final ) and lazy loading, by default it will do lazy loading, if you want this load() to do eager fetching then add lazy=false attribute to the class tag, when data is not there it will throw objectNotFoundException();

this method will return proxy class object as an implementation class object and hits the database only when non identifier methods are called in proxy class object

Even though u took lazy=true in below case it will perform eager loading

in case of lazy loading, the proxy class will be created as a child of entity, so if we made entity class as final then proxy can not be created by cglib or java assist libraries as a child of entity, so no lazy loading only eager even calling on ses.load() this also will behave like ses.get()

if we specify proxy interface name in the proxy attribute then proxy will be created as a child of that

prefer ses.load() when there is a layered application

1. how does hibernate know whether u are going to use the data or not?

If u call any setter method then hib understands that u need the data , so incase of lazy loading , until hibernate feels u need the data it won’t hit the database, so incase of lazy loading hibernate waits until a setter /getter method is called till then it will not hit the database and when u call database then it will hit the database.

It will delay how much it can delay

EX:-

Product product = session.load(Product.class, 1);  
System.*out*.println("load executed");  
System.*out*.println(product.getPname());

Console

load executed

Hibernate: select product0\_.pid as pid1\_0\_0\_, product0\_.pname as pname2\_0\_0\_, product0\_.price as price3\_0\_0\_, product0\_.qty as qty4\_0\_0\_ from PRODUCT product0\_ where product0\_.pid=?

realme-GT tv

observe from above that, after loading also query didn’t executed, query fired only when we called getter method(no @Id method)

1. l1 cache = session level cache

All the below methods will result in keeping an object in session/L1 cache

ses.get(Product.class,2L); will load from db and keeps in l1 cache

ses.update(p); ses.save(p); ses.delete(p); ses.saveOrUpdate(p); // will puts in cache

Product p=new product();

p.setId(2L);

ses.save(p); // this will also try to put an object into session cache ,but since an another object with same id is already present in cache it will throw non unique object exception

1. Sync from Entity to database

Load the object from database and make changes and commit the transaction and don’t even do session.update() it will automatically compare the changes and creates update query and syncs it with db

In O-R mapping software if u change objects data it will reflect those changes in table rows and vice versa. Solution:- this happens based on @Id field hibernate fetches the record using select query and based on this field only update query will be executed

public static void syncFromEntityObjectToDatabase(){  
 Session session = HibernateUtil.*openSession*();  
 *//* ***ses.load() or ses.get() will not only load's from db ,it will keep this in session or l1 cache***Product product = session.load(Product.class, 1);  
 product.setPname("realme-GT");  
 Transaction tx = session.beginTransaction();  
 *//* ***now we are modifying object present in l1 cache , it internally compares the previous and  
 // current object and generates the update query and changes are automatically synced without ses.update***tx.commit();  
}

1. **Internal code: internally uses JDBC only**

Session.get(InsurancePolicy.class,100);

Hibernate will talk to jdbc as below

ResultSet rs=con.createPreparedStatement(“select \* from InsurancePolicy”).executeQuery();

while(rs.next()){rs.next();}……………………….;

by default the cursor will be outside of result set , when u call rs.next() then it will point to 1st row.

1. Sync from database to entity ses.refresh

**In a transaction , after loading an object , If u modify the object and if u commit then synchronization will happen from object to db**

**Same after loading an object that object will be there in session cache or l1- cache , if u immediately went to database and modify that record and if u call session.refresh then db updates will be synced to your object automatically,**

**Assignment:-**

**open 10 sessions and load an object (which keeps into l1 cache), then go and modify in database table record which is same record which is in l1-cache, and if u click ses.refresh on 10 sessions then to all 10 sessions updates should come from database**

1. ses.update()

Ses.update() will update as it is, means if for that pojo 10 fields are there , if u populate 2 fields among 10,even if remaining 8 fields are null ,it will update all 10 fields with 8 null,

It will just update everything or all fields based on the id, It don’t care if something is null

1. What is dynamic update?

It will compare the all the field of object which we are passing with the object already present in cache

Internally it will compare which fields are modified

In case of ses.load() and ses.get(), the objects will be db loaded and kept in session cache

Now while ses.update(p) with dynamic-update=true, then since we are passing same object with little modifications which was already there in the l1 cache, as

dynamic-update= true , then then sql query will be generated only for the modified values instead of generating updating query with 100 columsn if it has,

But it will only do update it will not do insert, if no record with that id means runtime exception only

Caution!:-

If u don’t load using ses.load() /ses.get() and if are doing only ses.update() it will behave like ses,save(),

**Approach 1:- only update method Without load (use this for full object modification)**

In this case we will not load the entity from database instead since we know the id we

Populate and we send the object, in this case, whether dynamic-update=true or dynamic-update=false doesn’t make any difference, because since u didn’t loaded the object at all

There is nothing present in the cache to compare, that’s why in this case always all columns will be updated

|  |  |
| --- | --- |
| Don’t load using ses.load()/ses.get() +and do ses.update() with dynamic-update=false | **It is updating all columns with null cols too**  **update PRODUCT set pname=?, price=?, qty=? where pid=?** |
| Don’t load using ses.load()/ses.get() +and do ses.update() with dynamic-update=**true** | **Same as above all columns will be updated** |

session.update(p); // this will result in full object modification , if u want to update 1 field

U should to set remaining existing fields as it Is , but to set like that u should know all of them including with that id.

This is good if u want to update the complete record

and in this case, if record is not there it will throw exceptions-it wont insert

if u didn’t load using ses.load() and if u are simply calling the update() method, as hibernate didn’t loaded that object since it is not in l1 cache, whatever the data u give it will simply generate update query based on all fields, even if u gave same data , since hib doesn’t know as it (is not loaded by hib ,)it will simply generate the update query **with all fields even if dynamic-update is true and even if all fields are same**

1. Best partial modification approach- load and update

|  |  |
| --- | --- |
| Load using ses.load()/ses.get() +and do with **dynamic-update=false** | **Since this flag is false , stating no dynamic update , means don’t compare with the object present in session cache its updating all columns**  **update PRODUCT set pname=?, price=?, qty=? where pid=?** |
| Load using ses.load()/ses.get() +and do ses.update() with dynamic-update=**true** | **Since true, it will start comparing with the object present in cache, Now its updating columns only which we modified**  **update PRODUCT set pname=? where pid=?** |

<class name="com.nt.entity.Product" table="PRODUCT" dynamic-update="true">

Dynamic update means it is comparing previous value of that object which was there in session l1 cache with the current object based on that only it is identifying what are all the modifications made to that object

**Approach 2:-- load and modify (**This is good for partial object modification**)**

first load that object and modify what ever u want, but still it will create update query with all the fields, when u load and update , update query will be generated only when u modify the object, if u kept same values , simple update query will not be generated

**Approach 3:-- load and but modify automatically without ses.update**

After loading that object, it will be in session cache / L1 cache

After modifying u commit the transaction then automatically update query will be generated (but will fields / some fields?) This is good for partial modification

public static void partialModification\_loadAndUpdate(){  
 System.*out*.println("load and update");  
 Session session = HibernateUtil.*openSession*();  
 *// ses.load will not only load's from db ,it will keep this in session or l1 cache* Product product = session.get(Product.class, 1);  
 product.setPname("realme-GT fridge");  
 Transaction tx = session.beginTransaction();  
  
 session.update(product);  
 *// this way of load and update is suitable for partial object modification  
 // now we are modifying object present in l1 cache , it internally compares the previous l1 cache obj and  
 // current object and generates the update query* tx.commit();  
}

1. Ses.merge()

This is same like git merge , like when we raised the pr , our changes will be merged with main branch

Similarly when we do merge , our object will be merged with the object present in the session cache.

Assignment:-

1. What if in session this object is not there?
2. If the obj present in cache is having 10 values and if ur obj which is going to get merged is having only 2 values then will remaining 8 be lost? After merging check the return object
3. Ses.delete()

To delete the record from database , we have 2 approaches

1. Direct delete without loading , while deleting give only id to that entity

Product p=new Product();

p.setId(2);

ses.delete(p);

1. Load and delte- if u want u can load and delete too
2. Ses.saveorUpdate()

If record is present then it will do update else it will do insert fine

Product p=new Product();

p.setId(2);p.setName(“Trimmer”);

now since it will do load operation, suppose in database if we have record with all 10 fields , but now we are giving only 1 field , will remaining 9 fields be lost? Or these values are also be injected to the existing object? What will happen

Assnmt:- if record is present , anyways hib will load , if it have 10 fields in loaded obj and if u are passing an object with 2 fields will existing 8 fields be overridden with null value??

1. @Id vs @EmbeddedId

<id> tag vs <composite-id> tag in xml

In database table if we have a primary key column, in java class we should mark that field in entity class with @Id annotation

In db if we have composite primary key then use @EmbeddedId in annotation env use composite-id tag in xml approach,

It is not mandatory to have a pk column in db , but its mandatory to have @Id on entity class

1. Note:- taking primary key column is optional in table but taking @Id in entity class is mandatory JVM identifies every object uniquely using hashcode, lly hibernate identifies that entity record uniquely using @Id field
2. Ses.beginTransaction() means it internally calls

connection,setAutoCommit(false);

* session.save(employee) / **L1- cache**; will not immediately insert any record in database, Until you say tx.commit() {which is internally con.commit()} all the objects with modifications are present in hibernates L1 cache, once commit is issued then hibernate will use prep statement and execute the query, it might not be new ,because even In editors like sql developer until u say commit , all those instructions will be in cache

1. Session= connection++ object

Session.beginTransaction() will internally calls con.setAutocommit(false)

The cache associate with session cache is L-1 cache /session cache

When **session.close()** is called then associated session/L1 cache will also get cleared

And when this close method is called then it closes and releases this connection to the connection pool

1. sessionFactory.close()

when sessionFactory.close() Is called then it will release entire connection pool,

Releases the cache associated with the session factory ==could be L2 cache

Nullifies the generators, dialects and vanishes the in memory metadata of all loaded xml files and mapping files and deactivates the hibernate framework

When configuration class is created then hibernate framework is activated and when session factory is closed then hibernate is deactivated

1. tx.commit() vs tx.rollback()

these are equal to con.commit() and con.rollback()

1. session.save() vs session.persist()

save allows to work with generators and returns the id value whereas persists doesn’t allow us to work with generators and doesn’t return any id value

persist is method of jpa specification

save can return the generated

1. story of dialect –pre generated SQL query

these dialect is very eager as and when hibernate is getting started it will scan all entities and generates the sql queries (like select,create,read,update) with @ID column and keeps in session factory since all are based on primary key single row operations these are generated bit early to avoid the delay

in most of the cases the dialect will be picked up automatically based on the database url we provide

1. dynamic-update vs dynamic-insert -HQL

the problem is it will involve all columns , so if u are updating only 1 column with out this , hb will generate an update query with all columns to avoid this only we will use this dynamic update and insert

there is no dynamic delete because we will delete all columns with pk id , so dynamic delete is not required.

if u want dynamic select – means projections like selecting only some columns then u have to go with hql

* note all single row operations will use only primary key to fetch and hit the database , if u want to fetch based on the custom field then go with HQL
* in real time mostly we will not fetch with generated value, we may use another field to fetch

1. try with resources
2. diff between update and merge
3. 3 approaches of updating the object

Product p=new Product();