Hibernate(ORM tool/framework)
==========
Language(java)
Technology(JDBC)
Framework(ORM tool)

HibernateArchitecture
==========
refer:******.png

Hibernate Configuration File will provide all the configuration details like driver class name, driver URL,
Database User name , Database password,.... which we required to establish connection with database and
to setup JDBC environment.

Hibernate Mapping file will provide all the mapping details like Bean Class name and Database table name, ID property and
Primary key column , Normal Properties and Normal Columns,....

1) Client Application will perform the following three actions in Hibernate applications mainly.
2) Activating Hibernate Software, in this case, Hibernate Software will take all configuration details from hibernate
configuration file and Hibernate Software will set up the required JDBC Environment to perform database operations.
3) Prepare Persistence Object with the persistence data.
4) Perform Persistence Operation.

When Client Application perform persistence operation, Hibernate Software will perform the following actions.
1) Hibernate Software will take persistence method call and identify persistence Object.
2) Hibernate Software will take all mapping details from hibernate mapping file like database table name and all
column names on the basis of Persistence object.
3) Hibernate Software will prepare database dependent sql query on the basis of table names and column names and
with the persistence object provided data.
4) Hibernate Software will execute the generated database dependent sql query and perform the required persistence
operation.

Steps to prepare Hibernate Application:
1) Prepare Persistence Class or Object.
The main intention of Persistence class or object in Hibernate applications is to manage Persistence data which we want to
store in Database or by using this we want to perform the database operations like select, update, delete.

In Hibernate applications, to prepare Persistence classes we have to use the following Guidelines
1) In Hibernate applications Persistence classes must be POJO classes [Plain Old Java Object], they must not extend or
implement predefined Library.
2) In hibernate Applications Persistence classes must be public, Non abstract and non-final. Where the main intention to
declare persistence classes as public is to bring persistence classes scope to

Hibernate software in order to create objects.
Where the main intention to declare persistence classes as Non abstract is to allow to create Objects for Persistence classes
Where the main intention to declare persistence classes as Non final is to allow to extend one persistence class to another
persistence class as per the requirement.
3) In Persistence classes all Properties must be declared as per database table provided columns, where names are not
required to be matched, but, data types must be compatible.
4) In Persistence classes, all properties must be declared as private in order to improve Encapsulation.
5) In Persistence classes , we must define a separate set of setXXX () and getXXX () methods for each and every property
6) In persistence classes, we must declare all methods are public.
7) In Persistence classes, if we want to provide any constructor then it must be public and 0-arg constructor, because,
while creating object for persistence class Hibernate software will search and execute only public and 0-arg constructor.
8) In Hibernate applications , if we want to provide our own comparison mechanisms while comparing Persistence objects
then it is suggestible to Override equals(--) method.
9) In Hibernate applications, if we want to provide our own hash code values then it is suggestible to Overrid hashCode ()
method.
10) In Hibernate applications, we will use POJO classes, which are not extending and implementing predefined library, but,
it is suggestible to implement java.io.Serializable marker interface in order to make eligible Persistence object for
Serialization and Deserialization.
11) In Persistence classes, we have to declare a property as an ID property, it must represent primary key column in the
respective table.

2) Prepare Mapping File.
The main intention of mapping file in Hibernate applications is to provide mapping between a class,id property and
normal properties from Object Oriented Data Model and a table, primary key column and normal columns from
Relational data model.

In Hibernate applications, mapping file is able to provide the mapping details like Basic OR mapping, Component mapping,
inheritance mapping, Collections mapping, Associations mapping,
To prepare mapping file in Hibernate applications we have to provide mapping file name with the following format.
"POJO_Class_Name.hbm.xml".
The above format is not mandatory, we can use any name but we must provide that intimation to the hibernate software.
In Hibernate applications, we can provide any no of POJO classes, w.r.t each and every POJO class we can define a separate
mapping file.
refer: *****.hbm.xml

3) Prepare Hibernate Configuration File
The main purpose of hibernate configuration file is to provide all configuration details of hibernate application which includes
Jdbc parameters to prepare connection , Transactions configurations,Cache mechanisms configurations, Connection pooling

configurations,.....
In Hibernate applications, the standard name of hibernate configuration file is "hibernate.cfg.xml", it is not fixed,
we can provide any name but that name must be given to Hibernate software.
In Hibernate applications, we are able to provide more than one configuration file , but, for each and every database, that is,
in hibernate applications if we use multiple databases then we are able to prepare multiple configuration files.
To prepare hibernate configuration file with basic configuration details we have to use the following XML tags.

refer: hibernate.cfg.xml

4) Prepare Hibernate Client Application
The main intention of Hibernate Client application is to activate Hibernate Software, creating persistence objects and
performing Persistence operations.
To prepare Client Application in hibernate applications we have to use the following steps.

1. Create Configuration class object
2. Create Session Factory object
3. Create Session Object
4. Create Transaction object if it is required.
5. Perform Persistence operations
6. Close Session Factory and Session objects.

1. Create Configuration class object
In Hibernate, the main intention of Configuration object is to store all the configuration details which we provided in
hibernate configuration file.
To represent Configuration object Hibernate has provided a predefined class in the form of "org.hiberante.cfg.Configuration".
To create Configuration class object we have to use the following constructor from Configuration class.

```
public Configuration()
```

EX: Configuration cfg = new Configuration();
If we use the above instruction in Hibernate applications then we are able to get an empty Configuration object in heap
memory, it will not include any Configuration details.

If we want to store Configuration details from Configuration file we have to use either of the following methods.
1. public Configuration configure()
This method will get configuration details from the configuration file with the name hibernate.cfg.xml.
2. public Configuration configure(String config_file_Name)
This method can be used to get configuration details from hibernate configuration file with an name, it will be used
when we change configuration file name from hibernate.cfg.xml file to some other name .
3. public Configuration configure(File file)
This method can be used to get configuration details from a file which is represented in the form of java.io.File class object.

```
public Configuration configure(URL url)
```

This method can be used to get Configuration details from a file which is available in network represented in the form of
java.net.URL .
EX: Configuration cfg = new Configuration();

```
cfg.configure();
```

When we use configure() method then Hibernate Software will search for

hibernate.cfg.xml file, if it is available then
Hibernate software will load the content of hibernate.cfg.xml file, parse it and read content from configuration file to
Configuration object.

2. Create Session Factory object:
In Hibernate, the main intention of Session Factory object is to manage Connections, Statements, Cache levels, ... and it able
to provide no of Hibernate Session objects.
To represent Session Factory object Hibernate has provided a predefined interface in the form of "org.hibernate.Session Factory".
To get Session Factory object we have to use the following method from Configuration class.

```
            public Session Factory buildSessionFactory()
EX: SessionFactory sf = cfg.buildSessionFactory();
```

In Hibernate applications, if we use multiple Databases then we have to prepare multiple Configuration files,
multiple Configuration Object, w.r.t this, we have to prepare multiple Session Factory objects.
Session Factory object is heavy weight and it is thread safe upto a particular Database, because, it able to allow more than
one thread at a time.

3. Create Session Object:
In Hibernate, for each and every database interaction a separate Session will be created.
In Hibernate, Session is able to provide no of persistence methods in order to perform persistence operations.
To represent Session object, Hibernate has provided a predefined interface in the form of "org.hibernate.Session".
To get Session object, we have to use the following method from Session Factory.

```
            public Session openSession()
EX: Session s =sf.openSession();
```

In Hibernate, Session object is light weight and it is not thread safe, because, for each and every thread a separate Session
object will be created.

Difference b/w Session Object vs SessionFactory object?

SessionFactory

- It is a heavy weight object containing multiple other objects like DataSource,Dialect,TxFactory,generators etc.
- It is a immutable Object.
- ThreadSafe Object is by default.
- Created by using buildSessionFactory() method which is designed based on "builder" design pattern.
- Long Lived Object of the application.
- It maintains Level-L2 cache.

Session

- It is a Light Weigth Object(con++)
- It is a mutable Object.
- Not ThreadSafe by default.
- Created by using openSession() which is designed based on 'factory' design pattern.
- Short lived Object of the application.
- It maintains Level-L1 cache.

4. Create Transaction Object:
Transaction is a unit of work performed by Front End applications on Back end systems.
To represent Transactions, Hibernate has provided a predefined interface in the form of "org.hibernate.Transaction".

To get Transaction object we will use either of the following methods.
1. public Transaction getTransaction()
It will return Transaction object with out begin, where to begin Transaction we have to use the following method.
public void begin()

2. public Transaction beginTransaction()
It will return Transaction and begin Transaction.
In Hibernate applications, after performing persistence operations we must perform either commit or rollback operations
inorder to complete Transactions, for this, we have to use the following methods from Transaction.

5. Perform Persistence Operations:
In Hibernate applications, to perform persistence operations Session has provided the following methods.

Through hibernate we can perform
a. SRO(Single Row Operation)
a. save()/persist() ========> Insert Operation
b. get()/load() ========> Select Operation
c. update()/saveOrUpdate() => Update Operation
d. delete() ==========> Delete Operation
b. Bulk operation(working with mulitple records)
a. HQL/JPQL
b. NativeQuery
c. Criterian API