**Hibernate Framework (ORM)**

It is ORM (Object Relational Mapping) created by Gavin King of Sun Systems after finding some issues with EJB-Entity. EJB-Entity had dependencies on Application Server and need to have EJB-Session for interaction. These dependencies have been eliminated in Hibernate Framework. This framework is developed for deploying DAO (Data Access Object) of Model Component of MVC Software Architecture. Hibernate do not require any application server like weblogic, Jbose etc. if there is some service required. It can independently handle the database operations which has features of HQL which is inbuilt Query Language. It can directly communicate with controller class if there is no security or service is required. And can communicate with EJB-Session beans also. All it requires is Hibernate libraries and JRE environment.

It has following features:

1. Auto DDL: It has the feature to perform DDL commands like create, modify, alter, drop and also builds the requires relationships among the tables as It uses Object Relational Mapping approach unlike Table Relation Mapping in EJB-Entity.
2. HQL: It has its own Query language called HQL (Hibernate Query Language) which converts the given HQL query in relevant SQL query of the resident database. So it does not have any problem when database migration happens. Let’s say an application uses Oracle Database and after one year the application has to be migrated to some other database say MySQL. At that it becomes a problem if we use conventional data connectivity software as different database may have different query for same operation. But in case of ORM, a standard query language is designed to tackle this problem.
3. Cache Support: Unlike JDBC, Hibernate provides us the cache support. It is very important as it reduces the number hits in the database for record fetching thus improving the performance. Consider an example, Everyday employee coming to the company has to swipe the card for his attendance and employee information is stored in database which changes occasionally may be once in a month or a week. So the entire employee data we can store in Cache so that number of hits to the database decreases.
4. Primary Key generator support: It creates a unique field called primary key whenever user submits a registration form.
5. Validations: It performs validations using annotations.
6. Exception Free: We don’t need to handle any compile time exceptions as it provides exception handling features which is handles internally.
7. ORM Support: It provides object to object relational mapping unlike Table relational mapping. So we can maintain IS-A and HAS-A relationship by using object to object relationship.
8. OGM (Object Graph Mapping): It has support for Non-relational database support like HSQL which is accomplished by OGM. (Hibernate 4)
9. Hibernate Search: They have given support for index based search. (Hibernate 4)

If we want to have CRUD operations then we have to pass the objects to the hibernate. Different types of object passed to the hibernate are:

1. Transient
2. Persistent
3. Detached

|  |  |  |
| --- | --- | --- |
| Id | Name | Email |
| 1111 | abc | abc@gmail.com |

Hibernate

Id = 1111

Name = abc

Email: abc@gmail.com

Database state

Transient State Persistent State

Remove Object(Detached State)

Till the object is not attached to the Hibernate, it is in Transient state. Once it is attached to the Hibernate it is moved to the persistent state. Once it is saved in the database and commit has been executed it moves to the database state. If we remove the object from the hibernate then it moves in the detached state.

For Hibernate applications, following things are required:

1. POJO class
2. Mapping File
3. Configuration file
4. Test Class

How to write POJO classes?

Rules of the POJO Classes for Hibernate are:

1. POJO class need to contain required number of properties which are being stored in the table,.
2. In POJO class default constructor is mandatory.
3. POJO class should be declared with public access modifiers.
4. For every property we need getters and setters.
5. Getters and setters method should have public access modifiers.
6. Along with these we can have extra variables and methods, constructors according to our needs.

For example

Public class student {

int id;

String name;

String email;

Int marks;

//getters and setters

}

Mapping file: When an object of student is attached to the hibernate, the same data has to be stored in the database with corresponding fields. So how the hibernate will know which data in the object is stored in which field of the database. So we have to configure the mapping for the same in mapping file. To map student bean object to the corresponding field in the database we need to create the Student.hbm.xml (mapping file).

It looks like

DTD definition

<hibernate-mapping>

<class name= “beans.Student” table=”Student007” schema=”scott”>

<id name=”id” column = “sid”/> (tag if primary key)

Or

<composite-id> (tag for composite primary key)

<key-property name=”id” column=”sid”/>

</composite-id>

(for non-key attributes)

<property name=”name” column=”sname”/>

<property name=”email” column=”email”/>

<property name=”marks” column=”marks”/>

</class>

</hibernate-mapping>

Configuration file is also required. This file name is hibernate.cfg.xml. It looks like

DTD definition

<hibernate-configuration>

Database Driver property Configuraion

Database connection pool Configuration

Hibernate commands configuration

Mapping files configuration

</hibernate-configuration>

To store data in the table we use have instantiate the POJO Class. And assign the data to the property.

Student st = new Student();

st.setId(111);

st.setName(abc);

st.setEmail([abc@gmail.com](mailto:abc@gmail.com));

st.setMarks(78);

the above bean object is transient and to store the same in the database we need to have test class. And the configuration class looks like as follows:

Configuration cfg= new Configuration(); (is a container class of Hibernate)

cfg.configure(); (loads configuration xml file which is hibernate.cfg.xml)

SessionFactory st = cfg.buildSessionFactory();

Session s= st.openSession(); //(Session is similar like JDBC connection object)

s.save(st); //(st object of Student POJO class now has persistent state)

s.beginTransact().Commit() ; //Now the st object has permanent state

s.remove(st); //Garbage collector will remove the data from memory

In POJO beans the data in in transient state. While it is attached to the hibernate then it is moved to persistent state and it will remain in the memory as long as we want. Java garbage collector will not remove it from the memory until it is detached from the hibernate.