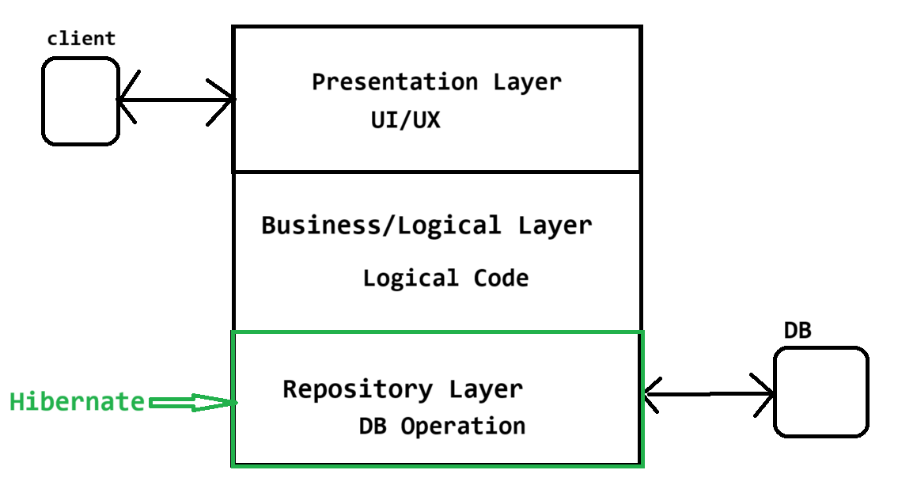
**Hibernate Framework**

What is Framework?

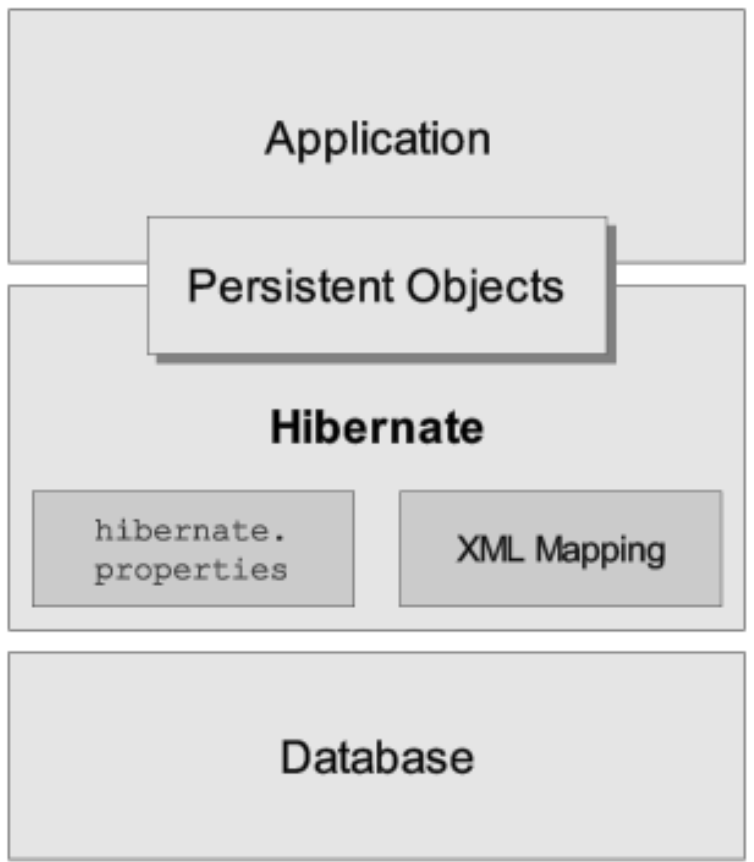
1. Framework is a predefine functionalities developed and provided by some communities.
2. Using framework you can achieve faster development by reducing a boiler plate code.
3. The functionalities are provided in the form of jar files which needs to include into application to integrate with framework.

What and Why Hibernate?

1. Hibernate is a ORM (Object Relational Mapping) tool.
2. Hibernate is mainly used for database operations.
3. Using hibernate you can make you database interaction easier.
4. Hibernate is used at the repository layer.
5. Using Hibernate can develop the database agnostic application.
6. Hibernate can execute DDL and DML queries internally.
7. Hibernate supports the caching to provide faster execution.
8. Hibernate handled all the SQL checked exception internally and converted them into unchecked exception.
9. Hibernate connects with the database using the external configurations using xml/properties files.



**Hibernate architecture**



**State Of an Object**

**Transient Object:**

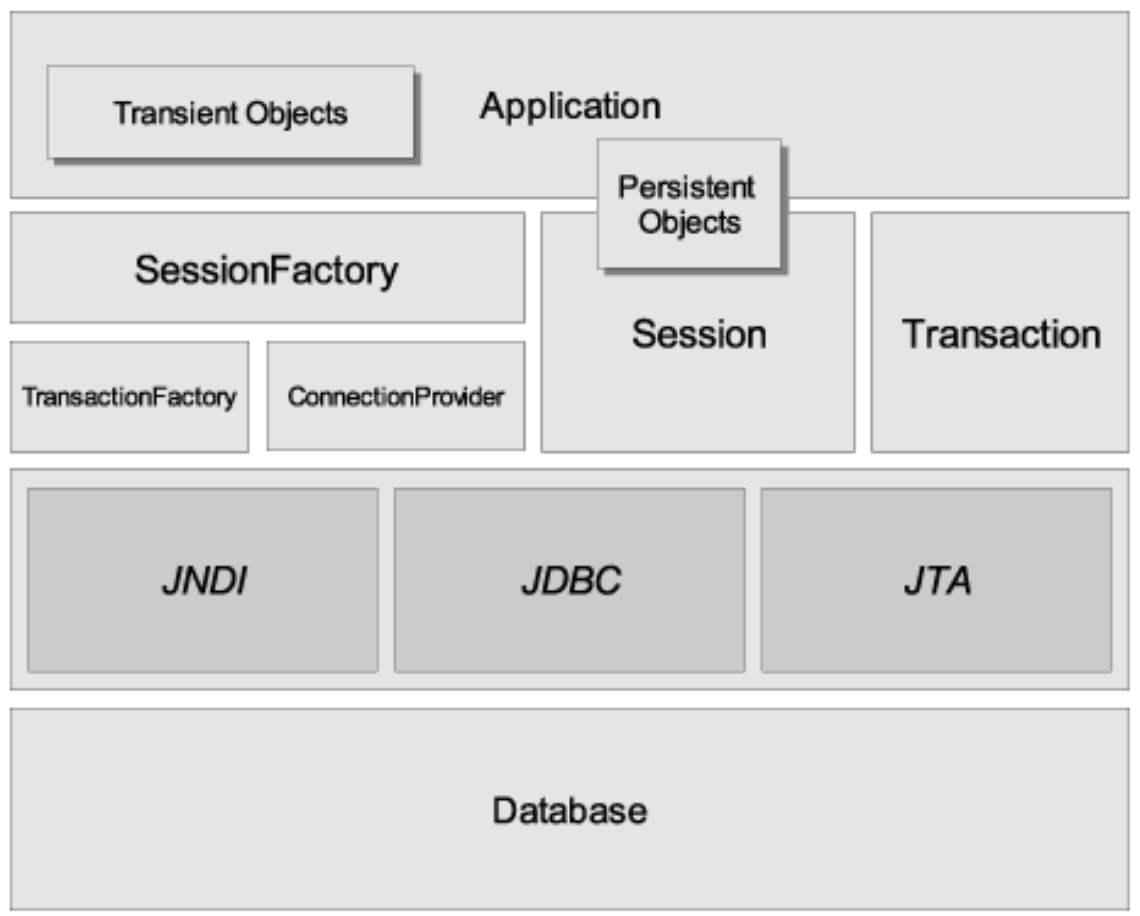
The entity class Object create inside Java application called as transient Object

**Persistent Object:**

The Entity class Object provide to a hibernate for DB operation is classed per persistent object

**Detached Object:**

The entity class object comes outside the hibernate after DB operation is called detached object.



Integrate hibernate with Java Application

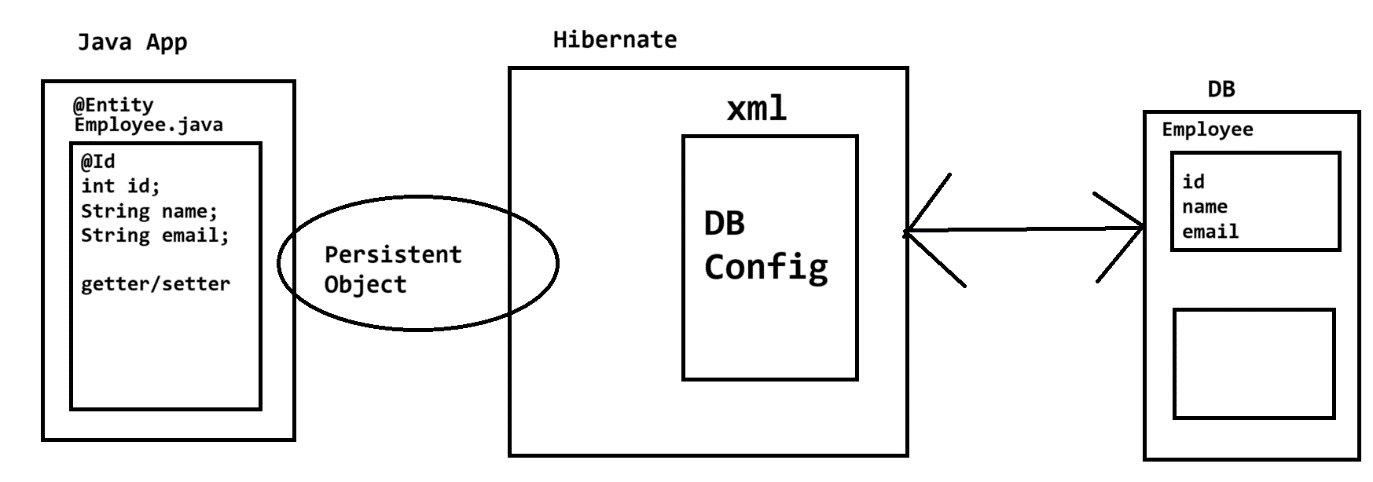
1. Add a jar file into application.
2. Add Hibernate Configuration in to application.
3. Use Hibernate APIs to perform the DB operations.
   1. **SessionFactory**
      1. It is use to connect with the Database and also Provides the Objects of the session.
      2. It the heaviest object in the application. And there is only one object of the SessionFactory per database.
      3. You have to provide the Database configuration to the session factory.
   2. **Session**
      1. Session is use to perform the DB operations (mostly a DML and DQL operations)
      2. Session Objects has short lifespan.
      3. These Object will be provided by the SessionFactory.
   3. **Transaction**
      1. One transaction is a combination of multiple queries.
      2. In transaction either all of the queries will be committed or all the operations will be rollback.
      3. There can be a multiple transaction in a session.

**Create hibernate Project**

1. Create java Application
2. Add Jar files
   1. The jar files from the required folder of the downloaded file.

Extracted Folder/lib/required.

1. Hibernate Configuration file
   1. Create an XML file inside src folder
      1. Right Click on src
      2. Go to “New” Option -> click on “Others”
      3. Search for “XML” into wizard
      4. Select the “XML File” option -> click on Next
      5. Provide the file (hbm-cfg.xml)
      6. Click on Finish
   2. Add the hibernate Configurations inside it.
2. Create Entity class
   1. Create Java class Which has private variables and getter and setter methods.
   2. This is same as a DTO class.
   3. You can use annotation @Entity at class level and @Id at variable.
   4. All the annotation has to import from the **javax.persistence package**
3. Make an entry of the entity class inside xml file.
4. Hibernate APIs to perform Operations
   1. Configuration
   2. SessionFactory
   3. Session
   4. Transaction



**CURD Operation using hibernate predefine methods**

**Insert** Data into table

\* save(entity\_object)

\* persist(entity\_object)

**Update** the data into table

\* update(entity\_object)

\* saveOrUpdate(entity\_object)

**Delete** the data from table

\* delete(entity\_object)

**Select** the data from table

\* get(EntityClassname, Primary\_Key\_Value)

\* load(EntityClassname, Primary\_Key\_Value)

**Different between get() and load()**

**Get()**

1. It is a EAGER fetch type
2. This return the actual object of the entity class.
3. The object return by this method will be available after session close.

**Load()**

1. It is a LAZY fetch type.
2. This return the proxy object of entity class.
3. The object return by this method will not be available outside session.

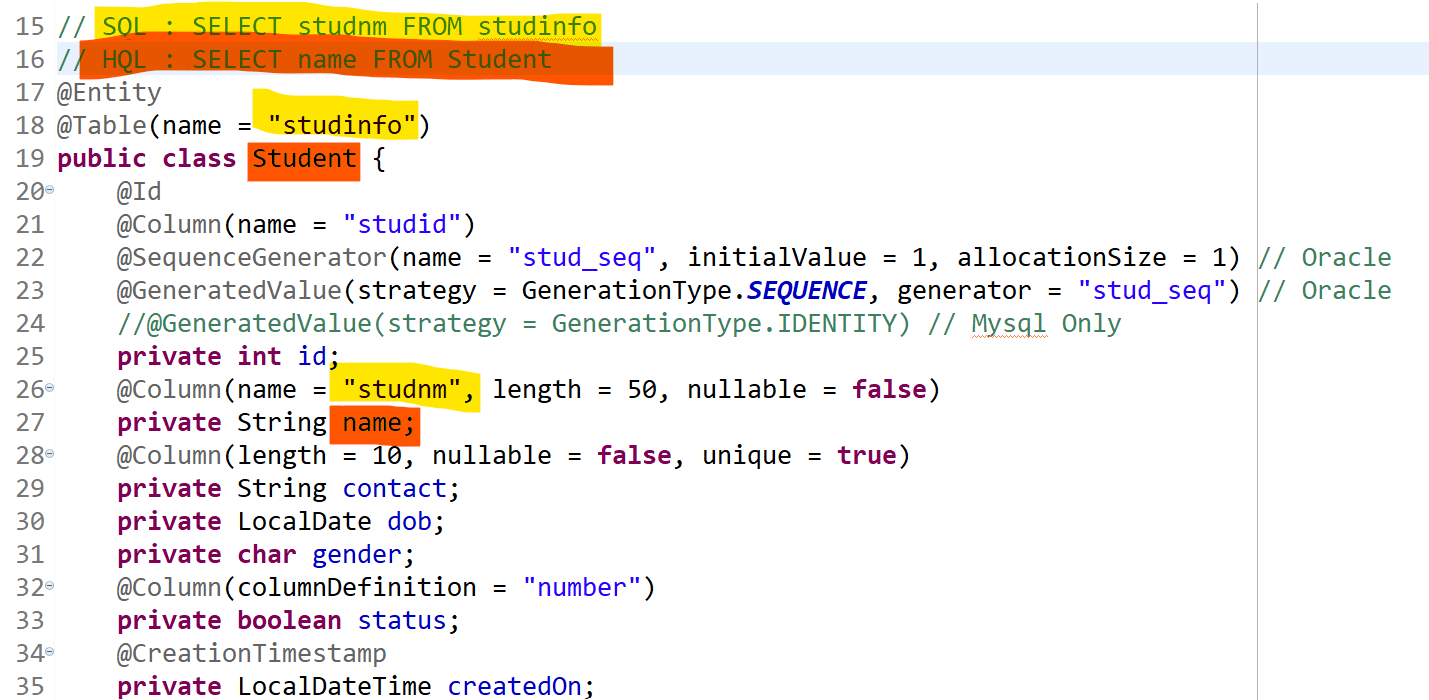
**Query Implementation**

1. Using this you will get a flexibility to write your own Query and execute it.
2. Hibernate provides this option to customize the query execution.
3. To execute the query in Hibernate you have to use **Query interface.**
4. In Hibernate you can represent the query by 2 ways.
   1. HQL
      1. HQL is Hibernate Query Language
      2. In this query language you have to use the Entity name and variable name instead of table and column name.
      3. Hibernate converts the HQL query into SQL before execution.
      4. Syntax:

session.createQuery(“HQL”);

* 1. SQL
     1. SQL is Structure Query Language
     2. This is the normal query patterns which use in the database.
     3. This need not any conversion in the hibernate.
     4. This type of queries is also known as native queries.
     5. Syntax:

session.createNativeQuery(“SQL”);



**Relation Mapping (HAS-A relation)**

1. If you are working on multiple tables and there is a Primary and Foreign Key relation between those table, then you can achieve the similar relation in the entity class using HAS-A relation.
2. HAS-A relation is creating reference/Object of one class into another class.
3. To Create a Foreign Key in hibernate you have to create reference of one class into another class.
4. You have to create a reference of class for which you wants to create foreign key, inside a class where you wants to create foreign key
5. In Entity classes you have to define the annotation on the HAS-A relation
   1. @OneToOne
      1. By Default Fetch type is EAGER
   2. @OneToMany
      1. By Default Fetch type is Lazy
   3. @ManyToOne
   4. @ManyToMany

