Hibernate-

Hibernate frameworks is mediator through which java application is communicated with database. It is open source frameworks.

Why?

* In JDBC, if we open a database connection we need to write in try, and if any exceptions occurred catch block will take care about it, and finally used to close the connections.
* We must close the connection, or we may get a chance to get connections error message.
* Actually if we didn’t close the connection in the finally block, then jdbc doesn’t responsible to close that connection.
* In JDBC we need to write Sql commands in various places, after the program has created if the table structure is modified then the JDBC program doesn’t work, again we need to modify and compile and re-deploy required, which is tedious.
* To overcome above draw backs we should go for Hibernate framework.

Advantages of Hibernate-

* 1. It is open source frameworks.
  2. Faster performance-

It uses cache concept hence the performance is fast.

* 1. Database independent query-

It generates the database independent query.

* 1. Automatic table creation-

It has facility to create the database tables automatically. There is no need to create the database tables manually.

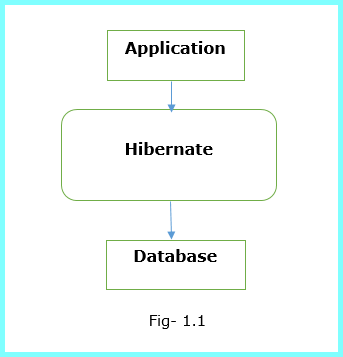
* 1. Simplifies the complex join-

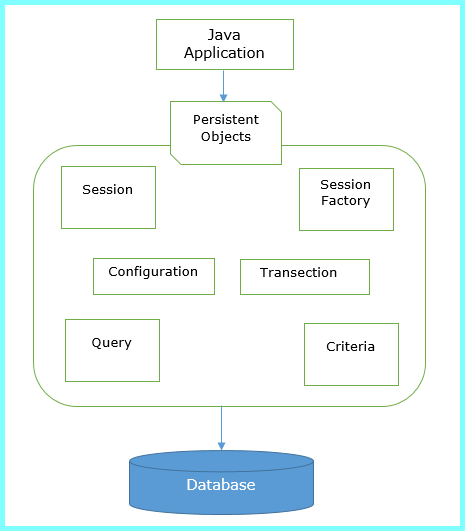
It is easy to fetch the data from multiple tables in hibernate framework.

Hibernate Architecture-

Hibernate architecture includes the many persistent objects such as session, session factory, transection factory, connection factory and transection etc. there are different types of layer in hibernate such as

* Java application layer
* Hibernate frameworks layer
* Database layer





Key points of Hibernate architecture-

1. Session Factory-

It is factory of session. It holds the second level cache. Session Factory interface provides the factory method to get the object of session.

1. Session-

It is the factory of transection, query and criteria. It holds first level cache. It provides the method to insert, update and delete the objects. It also provides the factory method for transection, query and criteria, etc.

1. Transection-

It is the interface that provides the method for transection management.

1. Connection provider-

It is the factory of JDBC connection. Which driver is used to connect to database.

1. TransectionFactory-

It is the factory of transection.

Note- Every hibernate program we must need two files for mapping.

# First Hibernate Application

Hibernate operation-Insert data into table.

1. Create java project
2. Add jar for hibernate
3. Create the persistent class or POJO class
4. Create the mapping file for persistent class
5. Create configuration file
6. Create the class that store persistent objects
7. Run application.
8. Create the java project

File menu->New->Project->Java Project->Specify Project name->next->finish.

1. Add Jar files

Right click on your project->build Path->Configure build path-

>Libraries->Add External Jar->Go to location where store the jar-> Select All jar-> click on open button->Apply and close button.

1. Student.java (POJO class)

**public** **class** Student {

**private** **int** id;

**private** String name;

**private** String city;

**private** String mobile;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getMobile() {

**return** mobile;

}

**public** **void** setMobile(String mobile) {

**this**.mobile = mobile;

}

}

Note-Hibernate mapping file is used by hibernate framework to get the information about the mapping of a POJO class and a database table.

1. student.hbm.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name=*"com.test.Student"* table=*"empl"*>

<id name=*"id"*>

<generator class=*"identity"*/>

</id>

<property name=*"name"*/>

<property name=*"city"*/>

<property name=*"mobile"*/>

</class>

</hibernate-mapping>

Note- if we don’t specify column tag then it will take property name as table column name into database. If we specify column tag then it will take that as column name into database.

Element of mapping file-

1. **hibernate-mapping:** It is the root element.
2. **Class:** It defines the mapping of a POJO class to a database table.
3. **Id:** It defines the unique key attribute or primary key of the table.
4. **generator:** It is the sub element of the id element. It is used to automatically generate the id.
5. **property:** It is used to define the mapping of a POJO class property to database table column
6. hibernate.cfg.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/test</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.connection.password"*>root</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQL5Dialect</property>

<property name=*"hbm2ddl.auto"*>create</property>

<property name=*"show\_sql"*>true</property>

<mapping resource=*"student.hbm.xml"*></mapping>

</session-factory>

</hibernate-configuration>

1. Test.java

package com.hibernate;

import org.hibernate.Session; import org.hibernate.SessionFactory; import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class Test {

**public** **static** **void** main(String[] args) {

Configuration cfg = **new** Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sessionFactory = cfg.~~buildSessionFactory~~();

Session session = sessionFactory.openSession();

Transaction t = session.beginTransaction();

// insert data into database

Student student = **new** Student();

student.setName("ram");

student.setCity("pune");

student.setMobile("9595978");

session.save(student);

t.commit();

session.close();

System.***out***.println("Record saved successfully.");

}

}

Output-

Record saved successfully.