**1.Steps to Create Hibernate Application:**

**a.** Define a Hibernate persistence class/Domain class

b. create Hibernate configuration file.

c. Create Hibernate mapping File.

d. Develop the Client Application.

**A.Define a Hibernate Persistence class/Domain Class:-**  It is normal Java Bean class whose objects can manage persistent data in order to do persistent operations like inserting data ,deleting data, updating data and …etc.

**Example:**

We want to store student data(sid,sname,age).

Where

persistent data is “7, suku, 40”.

Persistent operation is inserting data.

To store and manage the persistent data , we need one java bean class. That class is said to be a Hibernate persistent class.

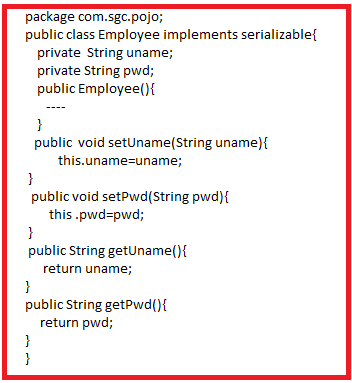
Some guide line have given to define the Hibernate persistence class. The guide lines are.

* The Persistence class must be a POJO(Plain Old java Object) class.
* The hibernate persistence class must be public,non-abstract and non-final class.

In order to bring the scope of hibernate persistence class to Hibernate software , the persistence class to be declared as public class. That’why hibernate software can create object to persistence class.

The main intension to declare persistence class as non-final is to allow the inheritance between the persistence classes.

* In Domain class, all properties must be declared as private properties.
* In Domain class, The programmer define setter (-) and getter() methods for each property.
* Define public and no-param constructor in Domain class. Because for creating Domain class object, hibernate software invokes the 0-param constructor of Domain class.
* In Domain class, the property with name ‘Id’ to be declared. The represents primary key of table.



**b. create Hibernate configuration file:-**

It is xml Document. Any name we can use for Hibernate configuration file. But Hibernate gave naming convention to configuration file.



Example:

Employee.hbm.xml

Note:- a.**In single hibernate application, we can provide multiple hibernate configuration files.**

**b**. The Domain class properties names and column names of relation need not be same.

The hibernate configuration file have following configurations.

* Basic O-R mapping Details

className ------ Table name

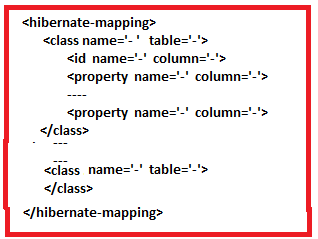
Id ---------------- Primary key of Table

Normal properties --------- normal columns of table.

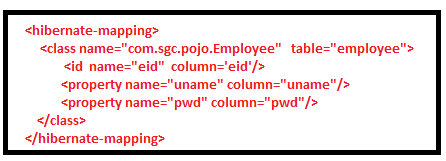
* Inheritance mapping.
* Component mapping.
* Association Mapping.
* Filter configurations.
* Procedure and Functions Configurations.
* …etc.

The following tags are used for Basic O-R mapping.

1. <class name=’=’ table=’-‘> for mapping the Domain class in java with Relation in DB.
2. <id name=’-‘ column=’-‘> for mapping the id property in Domain class with Primary key of Relation.
3. <property name=’-‘ column=’-‘ > for mapping normal properties of domain class with normal column of relation.
4. <hibernate-mapping> is root tag. <class> is child tag to <hibernate-mapping> tag . <id> and <property> tags are child tags to <class> tag.
5. To name attribute , assign the Domain class name or property name of Domain class .
6. To table and column attributes, assign the tablename and column names respectively.



Example:



**c. Create Hibernate mapping File:-**

It is also xml document. Suggested name to Hibernate mapping File is



It is not fixed name. we can use name to this document.

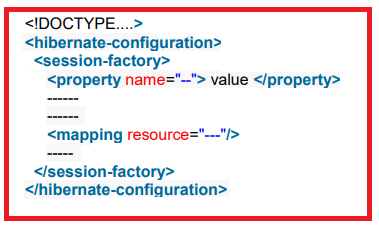
The JDBC Driver class Name

* DB url
* DB Username
* DB Password
* Mapping configuration
* Connection pooling configuration
* Transaction isolation level configuration.

…etc

The above details are informed to hibernate software in the form of xml entries of hibernate mapping file.

Syntax:



The following tags are used to define the configurations.

1. <hibernate-configuration> is root tag.
2. <session-factory> is able to hold the multiple <property > tags.
3. With <property> tag, we provide property configuration. It is paired tag.

To name attribute in <property>tag, we should assign the **property name** and this tag takes **property value** as content.

1. <Mapping> tag is able to provide mapping file configuration.

To resource attribute in <mapping> tag, we should assign the hibernate mapping file address.

Example:-

A screen shot of a computer

Description automatically generated

A close-up of a document

Description automatically generated

**D.Develop the Client Application: -** The main intension of client application is to activate the hibernate software, creating persistence class objects and performing persistence operations.

Steps as per hibernate 3.x version:

Step1: create configuration object.

Step2: create SessionFactory object.

Step3: create Session object.

Step4: create transaction object, if required.

Step5: perform the persistence operations.

Step6: close sessionFactory object and Session objects.

**Step1:- Create configuration class object:-**

It is object of org.hibernate.cfg.Configuration. The main objective of this object is to activate the hibernate s/w.

Configuration conf=new Configuration();

Initially, the configuration object will be create in heap memory. It is empty object. That means It does not have any configuration details.

If we want to store configuration details from” then one of the following method to be invoke.

1. Public Configuration configure()

This method will get configuration details from “hibernate.cfg.xml file.

1. Public Configuration configure(String cfgfilename)

This method will get configuration details from configuration file with any name.

Internally what does configure() or configure(-)method do?

From the configuration file, this method knows the mapping file name. This method loads configuration file(hibernate.cfg.xml) and also mapping file(Domainclass.hbm.xml) file into JVM. If both files are not valid and well formed documents then exception will be raised otherwise “In line memory meta data” will be created in JVM. This In line memory meta data have configuration file data , mapping file data and extra information. The configuration object(contains) represents in-line memory meta data.

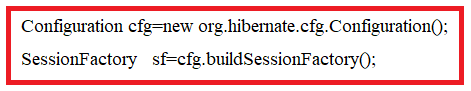
What is advantage by in-line memory meta data?

As per Hibernate application requirement , the both xml files are to be read and load 100 times into JVM and both xml file to be checked whether valid and well formed or not 100 times . This leads to reduce application performance. That’ why only one time , both xml file content will be read , load and verified .

It is placed in JVM until application closed. It increase application performance.

**Step2: create Session Factory object:-**

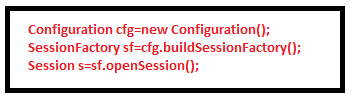
Hibernate provided one predefined interface which is org.hibernate.SessionFactory. Its implementation class object is given by buildSessionFactory() method. This method exist in org.hibernate.cfg. Configuration Class.



* The configuration object data and **connection pool** is dumped into SessionFactory object. Once dumped , the data can’t be changed because SessionFactory object is “immutable object”.
* The SessionFactory is heavy weight object because it contains the lot of information(all In-line memory meta data,jar files data, process data..etc).
* It is thread safe object.

**Step3: create Session object:-**

Hibernate provided one predefined interface which is org.hibernate.Session. Its implementation class object is given by openSession() method. This method exist in org.hibernate.SessionFactory Interface.



This object takes one connection object from connection pool which is represented by SessionFactory. through hibernate framework and JDBC driver software , it opens one session to DB software. This session allows application to do persistence operations.

**Step4: create transaction object, if required:-**

Creating the transaction object is optional. When application wants to do non-select persistent operations(update,delete,insert) , Transaction object has to be created.

Hibernate provided pre-defined interface in the form of “org.hibernate.Transaction”. Its Implementation class object is created by either following two methods:

1. Public Transaction beginTransaction()

This method create Transaction object and also starts it.

1. Public Transaction getTransaction()

This method only creates Transaction object but it does not start. To start transaction, Application has to invoke begin() method.

Public void begin();

After completing the persistence operation, we must perform either commit or rollback in order to complete the transaction. For this we have following two methods.

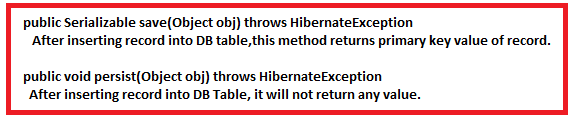
1. Public void commit();
2. Public void rollback();

**Step5: perform the persistence operations:-**

The session interface provided following methods to perform the persistence operations.

1. Save(--)
2. Persist(--)
3. Update(--)
4. saveOrUpdate(--)
5. get(--)
6. Load(--)
7. Delete(-)

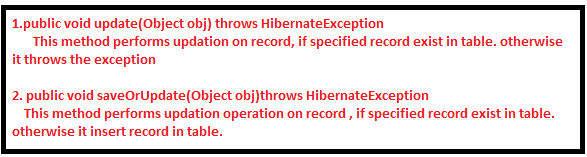
* To insert record/object into DB Table , we use either save(-) method or persist(--) method.



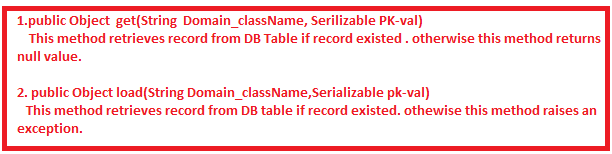
* To delete record from Db table, delete(--) method is used.



* In order to update the record in Table, we should use either update(--) method or saveOrUpdate(--) method.



* In order to get records from Table, we have to use either get(-) or load(-) method:



All above methods are used to perform single record manipulation only. If we want to retrieve multiple records , delete multiple records and update records then we should use

1. HQL.
2. Native SQL.
3. Criterion API.