***Hibernate Elements:***

***Steps to design Hibernate Applications:***

***Step1: Design Database as per the application requirement.***

***Step2: Design Persistence Class.***

***Step3: Prepare Mapping File***

***Step4: Prepare Hibernate Configuration file***

***Step5: Prepare Client application***

Step1: Prepare a table which contain primary key

***SQL> create table producthbr(pid varchar2(20),pname varchar2(20),pcost number(7,2));***

Table created.

Step2: Presistance Class

Before storing the data into database we have to represent the data in the object oriented model. For this we have to take an object and it is called as persistence object.

In this context a class which we used to prepare persistence object is called as persistence class.

Here we are using POJO classes as persistence classes.

POJO class is a normal java bean class which should not extend and should not implement any predefined library.

**public** **class** Product

{

**private** String pid;

**private** String pname;

**private** **int** pcost;

**public** **void** setPid(String pid)

{

**this**.pid=pid;

}

**public** **void** setPname(String pname)

{

**this**.pname=pname;

}

**public** **void** setPcost(**int** pcost)

{

**this**.pcost=pcost;

}

**public** String getPid()

{

**return** pid;

}

**public** String getPname()

{

**return** pname;

}

**public** **int** getPcost()

{

**return** pcost;

}

}

Step3:

Hibernate Mapping File:

The main purpose of the mapping file is to provide the mapping between POJO classes and the Data Base Tables, persistence class properties and data base table columns.

In hibernate applications we have to provide an individual mapping file for each and every persistence class.

To provide mapping file in hibernate applications we have to use the fallowing naming conventions.

***Persistence class name.hbm.xml***

We have to use fallowing xml tags:

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="Product" table="producthbr">

<id name="pid" column="pid"/>

<property name="pname"/>

<property name="pcost"/>

</class>

</hibernate-mapping>

Step4: Hibernate Configuration file

The main purpose of the Hibernate Configuration file is to provide configuration details of all the JDBC parameters, to establish the connection with data base like driver class, driver url, database user name, and password, data source configuration details like DSN name, Hibernate dialect configuration to load database specific configurations and automatic updations on database table at the time of loading Hibernate Software.

Ex:

<!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"*>sun.jdbc.odbc.JdbcOdbcDriver</property>

<property name=*"hibernate.connection.url"*>jdbc:odbc:challa</property>

<property name=*"hibernate.connection.user"*>scott</property>

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

<property name=*"dialect"*>org.hibernate.dialect.Oracle9Dialect</property>

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

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

</session-factory>

</hibernate-configuration>

***Step5: Prepare Client Application***

As part of the application design client application is nothing but a normal java class, or a servlet, a jsp page and ---

Steps to prepare client application:

Step1: create configuration object

Setp2: prepare session factory

Step3: Obtain session from session factory

Step4: prepare and start transaction

Step5: perform persistence operations

Step6: commit or rollback the transactions and close the session and sessionfactory.

1. Configuration cfg=new Configuration();

cfg.configure ();

Then jvm search for hibernate.cfg.xml file when it is identified jvm will pickup that configuration file and perform loading, parsing and read the content and stored into configuration Object.

1. And 3 :

As part of hibernate applications we need to get SessionFactory object we have to perform the persistence operations. We need to get predefined support from hibernate software and it is provided in the form of session object.

Use the interface for getting session object

**org.hibernate.Session**

**Session s=sf.openSession();**

**4.**

**Prepare transaction**

The purpose of the transactions in hibernate applications is to perform the persistence operations as a single unit.

Use the fallowing interface

**org.hibernate.Transaction**

**Transaction tx=s.beginTransaction();**

**tx.begin();**

**Perform persistence operations:**

**Create:**

**save();**

**Retrive:**

**get();**

**load();**

**Update:**

**update();**

**saveOrUpdate();**

**Delete:**

**delete();**

**public void commit();**

**public void rollback();**

* To compile client application we have to set the class path environment variable to Hibernate3.jar file
* To execute hibernate applications we have to set class path environment variable to the fallowing jar files.

ant-1.6.5.jar

antlr-2.7.6.jar

asm.jar

asm-attrs.jar;

cglib-2.1.3.jar

commons-collections-2.1.1.jar

commons-logging-1.0.4.jar

dom4j-1.6.1.jar

chcache-1.2.3.jar

hibernate3.jar

jaas.jar

jta.jar

ojdbc14.jar;

hibernate-annotations.jar

hibernate-commons-annotations.jar;

ejb3-persistance.jar

**To get Hibernate Mapping file DOCTYPE definition we have to use the fallowing path**

**Extract hibernate3.jar**

**Org-> hibernate->hibernate-mapping-3.0dtd**

**ClientApp .java:**

**import org.hibernate.\*;**

**import org.hibernate.cfg.\*;**

**import java.util.\*;**

**public class ClientApp**

**{**

**public static void main(String[] args)throws Exception**

**{**

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

**cfg.configure();**

**SessionFactory sf=cfg.buildSessionFactory();**

**Session s=sf.openSession();**

**Product p=new Product();**

**p.setPid("xyz1");**

**p.setPname("ddd");**

**p.setPcost(800);**

**Transaction tx=s.beginTransaction();**

**s.save(p);**

**tx.commit();**

**System.out.println("Product Insertion Success");**

**s.close();**

**}**

**}**

**set classpath=%classpath%;D:\Softwares\hbnjars\ant-1.6.5.jar;D:\Softwares\hbnjars\antlr-2.7.6.jar;D:\Softwares\hbnjars\asm.jar;D:\Softwares\hbnjars\asm-attrs.jar;D:\Softwares\hbnjars\cglib-2.1.3.jar;D:\Softwares\hbnjars\commons-collections-2.1.1.jar;D:\Softwares\hbnjars\commons-logging-1.0.4.jar;D:\Softwares\hbnjars\dom4j-1.6.1.jar;D:\Softwares\hbnjars\ehcache-1.2.3.jar;D:\Softwares\hbnjars\hibernate3.jar;D:\Softwares\hbnjars\jaas.jar;D:\Softwares\hbnjars\jta.jar;C:\oraclexe\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar;C:\Users\CollectionKING\Desktop\hbnjars\hibernate-annotations.jar;C:\Users\CollectionKING\Desktop\hbnjars\hibernate-commons-annotations.jar;C:\Users\CollectionKING\Desktop\hbnjars\ejb3-persistence.jar;**

**Basic Curd Operations Example2:**

**SQL> create table emp2hbr(eno number,name varchar2(20),esal number(10,2));**

**Table created.**

**Employee.hbm.xml**

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name=*"com.challa.cts.Employee"* table=*"emp2hbr"* lazy=*"false"*>

<id name=*"eno"* type=*"int"*>

<column name=*"eno"*/>

<generator class=*"increment"*/>

</id>

<property name=*"name"* column=*"name"*/>

<property name=*"esal"* column=*"esal"*/>

</class>

</hibernate-mapping>

**hibernate.cfg.xml**

<!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=*"connection.driver\_class"*>

sun.jdbc.odbc.JdbcOdbcDriver

</property>

<property name=*"connection.url"*>

jdbc:odbc:challa

</property>

<property name=*"connection.user"*>

scott

</property>

<property name=*"connection.password"*>

tiger

</property>

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

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

</session-factory>

</hibernate-configuration>

**Employee.java:**

**package** com.challa.cts;

**import** java.util.\*;

**public** **class** Employee

{

**private** **int** eno;

**private** String name;

**private** **double** esal;

**public** **void** setEno(**int** eno)

{

**this**.eno=eno;

}

**public** **int** getEno()

{

**return** eno;

}

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

{

**this**.name=name;

}

**public** String getName()

{

**return** name;

}

**public** **void** setEsal(**double** esal)

{

**this**.esal=esal;

}

**public** **double** getEsal()

{

**return** esal;

}

}

HibernateBasicTestCase.java:

**import** org.hibernate.cfg.\*;

**import** org.hibernate.\*;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.Level;

**import** java.util.Date;

**import** com.challa.cts.Employee;

**class** HibernateBasicTestCase

{

**private** SessionFactory factory;

**public** **void** createEmployee(Employee emp)

{

Session session=**null**;

Transaction tx=**null**;

**try**

{

session=factory.openSession();

tx=session.beginTransaction();

session.save(emp);

tx.commit();

System.*out*.println("Employee Inserted Successfully with Employee Number:"+emp.getEno());

}

**catch** (HibernateException hibernateException)

{

tx.rollback();

System.*out*.println("Exception in creating Employee");

}

**finally**

{

session.close();

}

}

**public** **void** updateEmployee(Employee emp)

{

Session session=**null**;

Transaction tx=**null**;

**try**

{

session=factory.openSession();

tx=session.beginTransaction();

session.update(emp);

tx.commit();

System.*out*.println("Employee Updated Successfully ");

}

**catch** (HibernateException hibernateException)

{

tx.rollback();

System.*out*.println("Exception in Updating Employee");

}

**finally**

{

session.close();

}

}

**public** Employee getEmployee(**int** eno)

{

Session session=**null**;

**try**

{

session=factory.openSession();

Employee emp=(Employee)session.load(Employee.**class**,**new** Integer(eno));

**return** emp;

}

**catch** (HibernateException hibernateException)

{

System.*out*.println("Exception in Loading Employee");

}

**finally**

{

session.close();

}

**return** **null**;

}

**public** **void** removeEmployee(Employee emp)

{

Session session=**null**;

Transaction tx=**null**;

**try**

{

session=factory.openSession();

tx=session.beginTransaction();

session.delete(emp);

tx.commit();

System.*out*.println("Employee Removed");

}

**catch** (HibernateException hibernateException)

{

tx.rollback();

System.*out*.println("Exception in Removing Employee");

}

**finally**

{

session.close();

}

}

**public** **static** **void** main(String[] args) **throws** Exception

{

Logger.*getRootLogger*().setLevel(Level.*OFF*);

HibernateBasicTestCase test=**new** HibernateBasicTestCase();

// Preparing configuration object

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

cfg.configure();

//Building Session Factory Object

test.factory=cfg.buildSessionFactory();

// Creating a New Persistant State

Employee emp=**new** Employee();

emp.setName("challa vasu");

emp.setEsal(12000);

test.createEmployee(emp);

//Updating Persistant State

emp.setEsal(25000);

emp.setName("vasu");

test.updateEmployee(emp);

//Getting an Employee

emp=test.getEmployee(Integer.*parseInt*(args[0]));

**if** (emp!=**null**)

{

System.*out*.println("Employee Details Of the Employee NUmber:");

System.*out*.println(emp.getEno());

System.*out*.println(emp.getName());

System.*out*.println(emp.getEsal());

}

//Removing an Employee

**if**(emp!=**null**)

{

test.removeEmployee(emp);

}

}

}

**Example 3:**

**create table personaldetails(pid number,firstname varchar2(20),lastname varchar2(20),permanent\_street varchar2(20),permanent\_city varchar2(20),permanent\_state varchar2(20),permanent\_pin number,present\_street varchar2(20),present\_city varchar2(20),present\_state varchar2(20),present\_pin number)**

**/**

**insert into personaldetails**

**values(7369,'Srinu','Reddy','Ameerpet','Hyd**

**erabad','AP','500049','TempleRoad','Hyderab**

**ad','AP','500048')**

**/**

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name=*"com.cts.srinu.PersonalDetails"* table=*"personaldetails"* >

<id name=*"pid"* >

<column name=*"pid"*/>

<generator class=*"increment"*/>

</id>

<property name=*"firstname"* column=*"firstname"*/>

<property name=*"lastname"* column=*"lastname"*/>

<component name=*"permanentAddress"* class=*"com.cts.srinu.Address"*>

<property name=*"street"* column=*"permanent\_street "*/>

<property name=*"city"* column=*"permanent\_city "*/>

<property name=*"state"* column=*"permanent\_state "*/>

<property name=*"pin"* column=*"permanent\_pin "*/>

</component>

<component name=*"presentAddress"* class=*"com.cts.srinu.Address"*>

<property name=*"street"* column=*"present\_street "*/>

<property name=*"city"* column=*"present\_city "*/>

<property name=*"state"* column=*"present\_state "*/>

<property name=*"pin"* column=*"present\_pin "*/>

</component>

</class>

</hibernate-mapping>

<!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=*"connection.driver\_class"*>

sun.jdbc.odbc.JdbcOdbcDriver

</property>

<property name=*"connection.url"*>

jdbc:odbc:challa

</property>

<property name=*"connection.user"*>

scott

</property>

<property name=*"connection.password"*>

tiger

</property>

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

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

</session-factory>

</hibernate-configuration>

**package** com.cts.srinu;

**public** **class** PersonalDetails {

**private** **int** pid;

**public** **int** getPid() {

**return** pid;

}

**public** **void** setPid(**int** pid) {

**this**.pid = pid;

}

**public** String getFname() {

**return** fname;

}

**public** **void** setFname(String fname) {

**this**.fname = fname;

}

**public** String getLname() {

**return** lname;

}

**public** **void** setLname(String lname) {

**this**.lname = lname;

}

**public** Address getPermanentAddress() {

**return** permanentAddress;

}

**public** **void** setPermanentAddress(Address permanentAddress) {

**this**.permanentAddress = permanentAddress;

}

**public** Address getPresentAddress() {

**return** presentAddress;

}

**public** **void** setPresentAddress(Address presentAddress) {

**this**.presentAddress = presentAddress;

}

**private** String fname,lname;

**private** Address permanentAddress,presentAddress;

}

**package** com.cts.srinu;

**public** **class** Address {

**private** String street,city,state;

**public** String getStreet() {

**return** street;

}

**public** **void** setStreet(String street) {

**this**.street = street;

}

**public** String getCity() {

**return** city;

}

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

**this**.city = city;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** **int** getPin() {

**return** pin;

}

**public** **void** setPin(**int** pin) {

**this**.pin = pin;

}

**private** **int** pin;

}

**import** org.hibernate.cfg.\*;

**import** org.hibernate.\*;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.Level;

**import** java.util.Date;

**import** com.cts.srinu.PersonalDetails;

**import** com.cts.srinu.Address;

**public** **class** PersonalDetailsTestCase

{

**public** **static** **void** main(String[] args) **throws** Exception

{

Logger.*getRootLogger*().setLevel(Level.*OFF*);

// Preparing configuration object

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

cfg.configure();

//Building Session Factory Object

SessionFactory factory=cfg.buildSessionFactory();

Session session=factory.openSession();

PersonalDetails personalDetails=(PersonalDetails).session.load(PersonalDetails.**class**,Integer.*parseInt*(args[0]));

System.*out*.println("First Name:"+personalDetails.getFname());

System.*out*.println("Last Name:"+personalDetails.getLname());

System.*out*.println("");

Address permanentAddress=personalDetails.getPermanentAddress();

System.*out*.println("PermanentAddress:");

System.*out*.println("\t Street :"+permanentAddress.getStreet());

System.*out*.println("\t City :"+permanentAddress.getCity());

System.*out*.println("\t State :"+permanentAddress.getState());

System.*out*.println("\t Pin :"+permanentAddress.getPin());

Address presentAddress=personalDetails.getPresentAddress();

System.*out*.println("PresentAddress:");

System.*out*.println("\t Street :"+permanentAddress.getStreet());

System.*out*.println("\t City :"+presentAddress.getCity());

System.*out*.println("\t State :"+presentAddress.getState());

System.*out*.println("\t Pin :"+presentAddress.getPin());

}

}

***Example Inheritance one to one relation:***

**SQL> create table EMPLOYEEHBR(empno number,userName varchar2(20),deptno number);**

**Table created.**

**SQL> create table personaldetailsHBR(empno number,firstname varchar2(20),lastname varchar2(20));**

**Table created.**

**SQL> insert into EMPLOYEEHBR values(101,'Srinu',10);**

**1 row created.**

**SQL> commit;**

**Commit complete.**

**SQL> insert into personaldetailsHBR values(101,'Vasu','Challa');**

**1 row created.**

**SQL> commit;**

**Commit complete.**

***Employee.java***

**package** cts.hbn.inheritance;

**public** **class** Employee {

**private** **int** empno;

**public** **int** getEmpno() {

**return** empno;

}

**public** **void** setEmpno(**int** empno) {

**this**.empno = empno;

}

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** PersonalDetails getPersonalDetails() {

**return** personalDetails;

}

**public** **void** setPersonalDetails(PersonalDetails personalDetails) {

**this**.personalDetails = personalDetails;

}

**private** String userName;

**private** PersonalDetails personalDetails;

}

***PersonalDetails.java:***

**package** cts.hbn.inheritance;

**public** **class** PersonalDetails {

**private** **int** empno;

**public** **int** getEmpno() {

**return** empno;

}

**public** **void** setEmpno(**int** empno) {

**this**.empno = empno;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**private** String firstName,lastName;

}

***Employee.hbm.xml:***

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package=*"cts.hbn.inheritance"*>

<class name=*"Employee"* table=*"EMPLOYEEHBR"* lazy=*"false"*>

<id name=*"empno"* type=*"int"*>

<column name=*"empno"*/>

<generator class=*"increment"*/>

</id>

<property name=*"userName"* />

<one-to-one name=*"personalDetails"* class=*"PersonalDetails"*/>

</class>

</hibernate-mapping>

***PersonalDetails.hbm.xml:***

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package=*"cts.hbn.inheritance"*>

<class name=*"PersonalDetails"* table=*"personaldetailsHBR"* lazy=*"false"*>

<id name=*"empno"* type=*"int"*>

<column name=*"empno"*/>

<generator class=*"increment"*/>

</id>

<property name=*"firstName"* />

<property name=*"lastName"* />

</class>

</hibernate-mapping>

***EmploeeTestCase.java:***

**import** org.hibernate.cfg.\*;

**import** org.hibernate.\*;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.Level;

**import** cts.hbn.inheritance.Employee;

**import** cts.hbn.inheritance.PersonalDetails;

**public** **class** EmploeeTestCase {

**public** **static** **void** main(String[] args) **throws** Exception

{

Logger.*getRootLogger*().setLevel(Level.*OFF*);

//EmploeeTestCase test=new EmploeeTestCase();

// Preparing configuration object

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

cfg.configure();

cfg.addResource("Employee.hbm.xml");

cfg.addResource("PersonalDetails.hbm.xml");

//Building Session Factory Object

SessionFactory factory=cfg.buildSessionFactory();

//Open a new session

Session session=factory.openSession();

// Creating a New Persistant State

Employee employee= (Employee)session.load(Employee.**class**, Integer.*parseInt*(args[0]));

System.*out*.println("Employee Details Of the Employee Number:");

System.*out*.println(employee.getUserName());

PersonalDetails personalDetails=employee.getPersonalDetails();

System.*out*.println(personalDetails.getFirstName());

System.*out*.println(personalDetails.getLastName());

} }

**Example HQL From clause:**

**package** cts.hql;

**import** org.hibernate.cfg.\*;

**import** org.hibernate.\*;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.Level;

**import** cts.hbn.inheritance.Employee;

**import** cts.hbn.inheritance.PersonalDetails;

**import** java.util.\*;

**public** **class** BasicHQLFromTestCase {

**public** **static** **void** main(String[] args) **throws** Exception

{

Logger.*getRootLogger*().setLevel(Level.*OFF*);

// Preparing configuration object

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

cfg.configure();

cfg.addResource("Employee.hbm.xml");

cfg.addResource("PersonalDetails.hbm.xml");

//Building Session Factory Object

SessionFactory factory=cfg.buildSessionFactory();

//Open a new session

Session session=factory.openSession();

// Creating a New Persistant State

Query query=session.createQuery("from Employee");

List<Employee> employees=query.list();

System.*out*.println("Empno\tUserName");

System.*out*.println("--------------------");

**for**(Employee e:employees)

{

System.*out*.print(e.getEmpno()+"\t");

System.*out*.print(e.getUserName());

System.*out*.println("");

}

} }

**Examples for HQL:**

**Employee.java:**

**public** **class** Employee {

**private** **int** eno;

**public** **int** getEno() {

**return** eno;

}

**public** **void** setEno(**int** eno) {

**this**.eno = eno;

}

**public** String getEname() {

**return** ename;

}

**public** **void** setEname(String ename) {

**this**.ename = ename;

}

**public** **int** getEsal() {

**return** esal;

}

**public** **void** setEsal(**int** esal) {

**this**.esal = esal;

}

**private** String ename;

**private** **int** esal;

}

**Employee.hbm.xml:**

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping >

<class name=*"Employee"* table=*"Emphbr1"* >

<id name=*"eno"* column=*"eno"*>

</id>

<property name=*"ename"* column=*"ename"* />

<property name=*"esal"* column=*"esal"* />

</class>

</hibernate-mapping>

**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"*>sun.jdbc.odbc.JdbcOdbcDriver</property>

<property name=*"hibernate.connection.url"*>jdbc:odbc:mydsn1</property>

<property name=*"hibernate.connection.user"*>training</property>

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

<property name=*"dialect"*>org.hibernate.dialect.Oracle9Dialect</property>

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

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

</session-factory>

</hibernate-configuration>

**Example for From Clause:**

**import** org.hibernate.\*;

**import** org.hibernate.cfg.\*;

**import** java.util.\*;

**public** **class** ClientApplication {

**public** **static** **void** main(String args[])**throws** Exception

{

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

cfg.configure();

SessionFactory sf=cfg.buildSessionFactory();

Session s=sf.openSession();

Employee emp=**new** Employee();

Query q=s.createQuery("from Employee");

List l=q.list();

Iterator i=l.iterator();

System.*out*.println("Eno \t Ename \t Esal");

System.*out*.println("------------------------");

**while**(i.hasNext())

{

emp=(Employee)i.next();

System.*out*.println(emp.getEno()+" \t "+emp.getEname()+"\t"+emp.getEsal());

}

s.close();

}

}

***Example for Remaining Queries like select, order by , where etc….***

**import** org.hibernate.\*;

**import** org.hibernate.cfg.\*;

**import** java.util.\*;

**public** **class** ClinetApplicationSelect {

**public** **static** **void** main(String args[])**throws** Exception

{

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

cfg.configure();

SessionFactory sf=cfg.buildSessionFactory();

Session s=sf.openSession();

Employee emp=**new** Employee();

System.*out*.println("Result From Select Query");

Query q=s.createQuery("select e.eno,e.ename from Employee e");

List<Object[]> l=q.list();

System.*out*.println("Eno \t Ename ");

System.*out*.println("------------------------");

**for** (Object[] result:l)

{

System.*out*.print(result[0]+"\t");

System.*out*.print(result[1]+"\n");

}

//trying for where clause

System.*out*.println("Result From Select Query and Where Clause");

Query q1=s.createQuery("select e.eno,e.ename from Employee e where e.esal<12000");

List<Object[]> l1=q1.list();

System.*out*.println("Eno \t Ename ");

System.*out*.println("------------------------");

**for** (Object[] result:l1)

{

System.*out*.print(result[0]+"\t");

System.*out*.print(result[1]+"\n");

}

//trying for Order by Clause

System.*out*.println("Result From Select Query and Where Clause and Order by ");

Query q11=s.createQuery("select e.eno,e.ename,e.esal from Employee e where e.esal>8000 order by e.esal DESC");

List<Object[]> l11=q11.list();

System.*out*.println("Eno \t Ename ");

System.*out*.println("------------------------");

**for** (Object[] result:l11)

{

System.*out*.print(result[0]+"\t");

System.*out*.print(result[1]+"\t");

System.*out*.print(result[2]+"\n");

}

s.close();

}

}