**ORM-Object Relationship Mapping.**

* It Reduces the developers efforts, time and cost-(as data is stored in tables i.e attribute format it is difficult to write query for all attributes in large database application).
* It Overcome the mismatch between object oriented programming language and database.

**Hibernate**

* Hibernate create SQL query at runtime according to database.
* It provide automatic table creation feature.
* It provide primary key auto-increment feature.
* It provide relationship mapping as,

1. Has-A relationship

**->**one-to-one.

->one-to-many

->many-to-one

->many-to-many

2. Is-A relationship

->Default Inheritance.

-> Single Table.

->Join

->Table for class.

* Hibernate convert all checked exception to unchecked exception.
* It provide cache mechanism 1st level cache and 2nd level cache.
* Hibernate provide HQL query and CriteriaBuilder Api.
* It provide validation feature.

|  |  |  |
| --- | --- | --- |
| XML | Annotation | Java Based |
| hibernate.cfg.xml | hibernate.cfg.xml |  |
| Student-hbm.xml |  |  |
| Student.java | Student.java | Student.java |
| Test | Test | Test |

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">

<!-- Generated by MyEclipse Hibernate Tools. -->

<hibernate-configuration>

<session-factory>

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

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/test1</property>

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

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

<property name=*"hbm2ddl.auto"*>update</property> <!-- create table at runtime -->

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQL5Dialect</property> <!-- it will configure database according to mysql -->

<property name=*"show\_sql"*>true</property><!-- false will not show sql query in console -->

<mapping class=*"com.cjc.Model.Employee"*></mapping><!-- if this class is changed the it

give classnotfoundexception -->

</session-factory>

</hibernate-configuration>

Student.java

**package** com.cjc.Model;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

@Entity

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

@Id

**private** **int** rollno;

**private** String name;

**private** String address;

**setters and getters**}

Test.java

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** com.cjc.Model.Student;

**public** **class** Test {

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

StandardServiceRegistry registry=**new** StandardServiceRegistryBuilder().configure().build();

MetadataSources mds=**new** MetadataSources(registry);

Metadata md=mds.getMetadataBuilder().build();

SessionFactory sf=md.getSessionFactoryBuilder().build();

Student stu=**new** Student();

stu.setRollno(2);

stu.setName("abc");

stu.setAddress("pune");

Session session=sf.openSession();

session.save(stu);

session.beginTransaction().commit();

}

}

Exceptions possible occur in hibernate execution of program:

1. **@Entity**: If @Entity is not written in above entity class (i.e pojo class) then it will give compile time exception as org.hibernate.MappingException-unknown exception.
2. **@Id:** if @Id is not written in above id then it will give Compile time Exception as org.hibernate.AnnotationException.
3. **Hibernate.cfg.xml:** if hibernate.cfg.xml file is not present in src directory it will give an error as could not locate cfg.xml resources.

**And if hibernate.cfg.xml file is named with different name then it will give an exception as classNotFoundExcaption.**

Properties of hdm2ddl.auto:

Create

Console output:

1. If table is not exist in database.

* Drop table if exists <tablename>
* Create table <tablename>
* Insert into <tablename>

1. If table is exist in database.

* Drop table if exists <tablename>
* Create table <tablename>
* Insert into <tablename>

Update

Console output:

1. If table is not exist in database.

* Create table <tablename>
* Insert into <tablename>

1. If table is exist in database.

* Insert into <tablename>

Create-drop

Console output:

1. If table is exist or not in database in case of sessionfactory.close().

* Drop table if exists <tablename>
* Create table <tablename>
* Insert into <tablename>
* Drop table <tablename>

1. If table is exist or not in database.

* Drop table if exists <tablename>
* Create table <tablename>
* Insert into <tablename>

Validate

Console output:

1. If table is not exist or attribute is present in entity class but not present in table or name is different in entity class and table attribute is different then it will give schema definition not found.

**Factory Design Pattern:**

Hibernate program without xml file:

HibenrateUtil:

**public** **class** HibernateUtil {

**private** **static** StandardServiceRegistry *registry*;

**private** **static** SessionFactory *sessionFactory*;

**public** **static** SessionFactory getSessionFactory() {

**if**(*sessionFactory*==**null**) {

*registry*=**new**StandardServiceRegistryBuilder().applySettings(settings).build();

MetadataSources mds=**new** MetadataSources(*registry*);

mds.addAnnotatedClass(Student.**class**);

Metadata md=mds.getMetadataBuilder().build();

*sessionFactory*=md.getSessionFactoryBuilder().build();

}

**return** *sessionFactory*;

}

}

Student.java:

**package** com.cjc.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="studentdetails")//to change the table name

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

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)//Auto Generation of Primary key

@Column(name="s\_id")//to change column name

**private** **int** sid;

@Column(name="s\_name")

**private** String sname;

@Column(name="addr")

**private** String address;

**public** **int** getSid() {

**return** sid;

}

**public** **void** setSid(**int** sid) {

**this**.sid = sid;

}

**public** String getSname() {

**return** sname;

}

**public** **void** setSname(String sname) {

**this**.sname = sname;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

}

Test.java

**import** java.util.List;

**import** org.hibernate.Session;

**import** com.cjc.entity.Employee;

**import** com.cjc.entity.HibernateUtil;

**public** **class** Demo {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Student stu=session.get(Student.**class**, 1);

//Student Stu=session.Load(Student.class, 1);//

System.***out***.println(stu.getSid());

/\*List<Student> slist=session.createQuery("from Student").getResultList();

for(Student s:slist) {

System.out.println(s.getSid()+" "+S.getSname());

}\*/

session.close();

}

}

|  |  |
| --- | --- |
| Lazy Loading | Eager Loading |
| Load() method work as a lazy loading as default | Get() method works as eager loading by default |
| Load() return hibernate exception(objectNotFoundException) if record is not present | Get() return null value(nullpointerException) if record is not present |

Java Based Hibernate Application:

HibernateUtil.java: changes only add hashmap to add jdbc properties.

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.cfg.Environment;

**import** com.cjc.entity.Student;

**public** **class** HibernateUtil {

**private** **static** StandardServiceRegistry *registry*;

**private** **static** SessionFactory *sessionFactory*;

**public** **static** SessionFactory getSessionFactory() {

**if**(*sessionFactory*==**null**) {

Map<String,Object> settings=**new** HashMap<>();

settings.put(Environment.***DRIVER***, "com.mysql.jdbc.Driver");

settings.put(Environment.***URL***,"jdbc:mysql://localhost:3306/cjc");

settings.put(Environment.***USER***, "root");

settings.put(Environment.***PASS***, "root");

settings.put(Environment.***DIALECT***,"org.hibernate.dialect.MySQL5Dialect");

settings.put(Environment.***HBM2DDL\_AUTO***, "update");//create

settings.put(Environment.***SHOW\_SQL***, "true");

*registry*=**new**StandardServiceRegistryBuilder().applySettings(settings).build();

MetadataSources mds=**new** MetadataSources(*registry*);

mds.addAnnotatedClass(Student.**class**);

Metadata md=mds.getMetadataBuilder().build();

*sessionFactory*=md.getSessionFactoryBuilder().build();

}

**return** *sessionFactory*;

}

}

Remaining Entity class and test class are same mentioned in factory design pattern

Relationship mapping:

1. One-to-one(unidirectional):

Account.java

**package** com.cjc.entity;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.OneToOne;

@Entity

**public** **class** Account {

@Id

**private** **int** acno;

**private** String name;

**private** String address;

@OneToOne(cascade=CascadeType.***ALL***)

@JoinColumn(name="acd")

**private** AccountDetails acdd;

**public** **int** getAcno() {

**return** acno;

}

**public** **void** setAcno(**int** acno) {

**this**.acno = acno;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** AccountDetails getAcdd() {

**return** acdd;

}

**public** **void** setAcdd(AccountDetails acdd) {

**this**.acdd = acdd;

}

}

AccountDetails.java:

**package** com.cjc.entity;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

@Entity

**public** **class** AccountDetails {

@Id

**private** **int** acd;

**private** **int** amt;

**public** **int** getAcd() {

**return** acd;

}

**public** **void** setAcd(**int** acd) {

**this**.acd = acd;

}

**public** **int** getAmt() {

**return** amt;

}

**public** **void** setAmt(**int** amt) {

**this**.amt = amt;

}

}

Hibernateutil.java:

**package** com.cjc.utiltiy;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.cfg.Environment;

**import** com.cjc.entity.Account;

**import** com.cjc.entity.AccountDetails;

**public** **class** HibernateUtil {

**private** **static** StandardServiceRegistry *registry*;

**private** **static** SessionFactory *sessionFactory*;

**public** **static** SessionFactory getSessionFactory() {

**if**(*sessionFactory*==**null**) {

Map<String,Object> settings=**new** HashMap<>();

settings.put(Environment.***DRIVER***, "com.mysql.jdbc.Driver");

settings.put(Environment.***URL***, "jdbc:mysql://localhost:3306/hibernate-cjc-onetoone");

settings.put(Environment.***USER***, "root");

settings.put(Environment.***PASS***, "root");

settings.put(Environment.***DIALECT***, "org.hibernate.dialect.MySQL5Dialect");

settings.put(Environment.***HBM2DDL\_AUTO***, "update");

settings.put(Environment.***SHOW\_SQL***, "true");

*registry*=**new** StandardServiceRegistryBuilder().applySettings(settings).build();

MetadataSources mds=**new** MetadataSources(*registry*);

mds.addAnnotatedClass(Account.**class**);

mds.addAnnotatedClass(AccountDetails.**class**);

Metadata md=mds.getMetadataBuilder().build();

*sessionFactory*=md.getSessionFactoryBuilder().build();

}

**return** *sessionFactory*;

}

}

Testunidemo.java:

**package** com.cjc.utiltiy;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.cfg.Environment;

**import** com.cjc.entity.Account;

**import** com.cjc.entity.AccountDetails;

**public** **class** HibernateUtil {

**private** **static** StandardServiceRegistry *registry*;

**private** **static** SessionFactory *sessionFactory*;

**public** **static** SessionFactory getSessionFactory() {

**if**(*sessionFactory*==**null**) {

Map<String,Object> settings=**new** HashMap<>();

settings.put(Environment.***DRIVER***, "com.mysql.jdbc.Driver");

settings.put(Environment.***URL***, "jdbc:mysql://localhost:3306/hibernate-cjc-onetoone");

settings.put(Environment.***USER***, "root");

settings.put(Environment.***PASS***, "root");

settings.put(Environment.***DIALECT***, "org.hibernate.dialect.MySQL5Dialect");

settings.put(Environment.***HBM2DDL\_AUTO***, "update");

settings.put(Environment.***SHOW\_SQL***, "true");

*registry*=**new** StandardServiceRegistryBuilder().applySettings(settings).build();

MetadataSources mds=**new** MetadataSources(*registry*);

mds.addAnnotatedClass(Account.**class**);

mds.addAnnotatedClass(AccountDetails.**class**);

Metadata md=mds.getMetadataBuilder().build();

*sessionFactory*=md.getSessionFactoryBuilder().build();

}

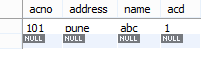
**return** *sessionFactory*;

}

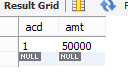
}

Output:

Account table:



Account detail table:



GetRecords from db:

**import** java.util.List;

**import** org.hibernate.Session;

**import** com.cjc.entity.Account;

**import** com.cjc.utiltiy.HibernateUtil;

**public** **class** TestUniDemoGet {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Account ac=session.load(Account.**class**, 101);

System.***out***.println(ac.getAcno());

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

System.***out***.println(ac.getAddress());

System.***out***.println(ac.getAcdd().getAcd());

System.***out***.println(ac.getAcdd().getAmt());

//to get all records from db

List<Account> list=session.createQuery("from Account").getResultList();

**for**(Account a:list) {

System.***out***.println(a.getAcno());

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

System.***out***.println(a.getAddress());

System.***out***.println(a.getAcdd().getAcd());

System.***out***.println(a.getAcdd().getAmt());

}

}

}

Output:

101

Hibernate: select account0\_.acno as acno1\_0\_0\_, account0\_.acd as acd4\_0\_0\_, account0\_.address as address2\_0\_0\_, account0\_.name as name3\_0\_0\_, accountdet1\_.acd as acd1\_1\_1\_, accountdet1\_.amt as amt2\_1\_1\_ from Account account0\_ left outer join AccountDetails accountdet1\_ on account0\_.acd=accountdet1\_.acd where account0\_.acno=?

abc

pune

1

50000

Hibernate: select account0\_.acno as acno1\_0\_, account0\_.acd as acd4\_0\_, account0\_.address as address2\_0\_, account0\_.name as name3\_0\_ from Account account0\_

101

abc

pune

1

50000

**One-to-one bidirection:**

Hibernateutil.java is same as above program

Account.java:

**package** com.cjc.entity;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.OneToOne;

@Entity

**public** **class** Account {

@Id

**private** **int** acno;

**private** String name;

**private** String address;

@OneToOne(cascade=CascadeType.***ALL***)

@JoinColumn(name="acd\_no")

**private** AccountDetails acdd;

**public** **int** getAcno() {

**return** acno;

}

**public** **void** setAcno(**int** acno) {

**this**.acno = acno;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** AccountDetails getAcdd() {

**return** acdd;

}

**public** **void** setAcdd(AccountDetails acdd) {

**this**.acdd = acdd;

}

}

AccountDetails.java

**package** com.cjc.entity;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.OneToOne;

@Entity

**public** **class** AccountDetails {

@Id

**private** **int** acd;

**private** **int** amt;

@OneToOne(cascade=CascadeType.***ALL***,mappedBy="acdd")

**private** Account account;

**public** Account getAccount() {

**return** account;

}

**public** **void** setAccount(Account account) {

**this**.account = account;

}

**public** **int** getAcd() {

**return** acd;

}

**public** **void** setAcd(**int** acd) {

**this**.acd = acd;

}

**public** **int** getAmt() {

**return** amt;

}

**public** **void** setAmt(**int** amt) {

**this**.amt = amt;

}

}

**TestBidirectionDemo.java**

**import** org.hibernate.Session;

**import** com.cjc.entity.Account;

**import** com.cjc.entity.AccountDetails;

**import** com.cjc.utiltiy.HibernateUtil;

**public** **class** TestBidirectionDemo {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Account ac=**new** Account();

ac.setAcno(101);

ac.setName("abc");

ac.setAddress("pune");

AccountDetails acd=**new** AccountDetails();

acd.setAcd(1);

acd.setAmt(50000);

acd.setAccount(ac);

ac.setAcdd(acd);

session.save(ac);

session.beginTransaction().commit();

}

}

Output:

Account table(In this table extra column is added as accountdetails is mapped to account)



Accountdetails table:



FetchRecords:

**import** org.hibernate.Session;

**import** com.cjc.entity.Account;

**import** com.cjc.entity.AccountDetails;

**import** com.cjc.utiltiy.HibernateUtil;

**public** **class** TestBidirectionDemoGet2 {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Account ac=session.get(Account.**class**, 101);

System.***out***.println(ac.getAcno());

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

System.***out***.println(ac.getAddress());

System.***out***.println(ac.getAcdd().getAcd());

System.***out***.println(ac.getAcdd().getAmt());

AccountDetails acd=session.get(AccountDetails.**class**, 1);

System.***out***.println(acd.getAcd());

System.***out***.println(acd.getAmt());

System.***out***.println(acd.getAccount().getAcno());

System.***out***.println(acd.getAccount().getName());

}

}

Output:

Hibernate: select account0\_.acno as acno1\_0\_0\_, account0\_.acd\_no as acd\_no4\_0\_0\_, account0\_.address as address2\_0\_0\_, account0\_.name as name3\_0\_0\_, accountdet1\_.acd as acd1\_1\_1\_, accountdet1\_.amt as amt2\_1\_1\_ from Account account0\_ left outer join AccountDetails accountdet1\_ on account0\_.acd\_no=accountdet1\_.acd where account0\_.acno=?

101

abc

pune

1

50000

1

50000

101

abc

2.One-to-many/may-to-one(note: bydefult its bidirectional):

Hibernateutil.java:

**package** com.cjc.utility;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.cfg.Environment;

**import** com.cjc.entity.Department;

**import** com.cjc.entity.Employee;

**public** **class** HibernateUtil {

**private** **static** StandardServiceRegistry *registry*;

**private** **static** SessionFactory *sessionFactory*;

**public** **static** SessionFactory getSessionFactory() {

**if**(*sessionFactory*==**null**) {

Map<String,Object> settings=**new** HashMap<>();

settings.put(Environment.***DRIVER***, "com.mysql.jdbc.Driver");

settings.put(Environment.***URL***, "jdbc:mysql://localhost:3306/hibernate-cjc-one-to-many");

settings.put(Environment.***USER***, "root");

settings.put(Environment.***PASS***, "root");

settings.put(Environment.***DIALECT***, "org.hibernate.dialect.MySQL5Dialect");

settings.put(Environment.***HBM2DDL\_AUTO***, "update");

settings.put(Environment.***SHOW\_SQL***, "true");

*registry*=**new** StandardServiceRegistryBuilder().applySettings(settings).build();

MetadataSources mds=**new** MetadataSources(*registry*);

mds.addAnnotatedClass(Employee.**class**);

mds.addAnnotatedClass(Department.**class**);

Metadata md=mds.getMetadataBuilder().build();

*sessionFactory*=md.getSessionFactoryBuilder().build();

}

**return** *sessionFactory*;

}

}

Employee.java:

**package** com.cjc.entity;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

@Entity

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

@Id

**private** **int** eid;

**private** String name;

**private** String designation;

@ManyToOne(cascade = CascadeType.***ALL***)

@JoinColumn(name = "d\_id")

**private** Department department;

**public** **int** getEid() {

**return** eid;

}

**public** **void** setEid(**int** eid) {

**this**.eid = eid;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getDesignation() {

**return** designation;

}

**public** **void** setDesignation(String designation) {

**this**.designation = designation;

}

**public** Department getDepartment() {

**return** department;

}

**public** **void** setDepartment(Department department) {

**this**.department = department;

}

}

Department.java:

**package** com.cjc.entity;

**import** java.util.HashSet;

**import** java.util.Set;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

@Entity

**public** **class** Department {

@Id

**private** **int** did;

**private** String dname;

@OneToMany(cascade=CascadeType.***ALL***,mappedBy="department")

**private** Set<Employee> semp=**new** HashSet<>();

**public** **int** getDid() {

**return** did;

}

**public** **void** setDid(**int** did) {

**this**.did = did;

}

**public** String getDname() {

**return** dname;

}

**public** **void** setDname(String dname) {

**this**.dname = dname;

}

**public** Set<Employee> getSemp() {

**return** semp;

}

**public** **void** setSemp(Set<Employee> semp) {

**this**.semp = semp;

}

}

Storedata:

**import** org.hibernate.Session;

**import** com.cjc.entity.Department;

**import** com.cjc.entity.Employee;

**import** com.cjc.utility.HibernateUtil;

**public** **class** StoreRecordEmployeeOneToMany {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Department dp=**new** Department();

dp.setDid(1);

dp.setDname("IT");

Department dp1=**new** Department();

dp1.setDid(2);

dp1.setDname("HR");

Employee e1=**new** Employee();

e1.setEid(101);

e1.setName("abc");

e1.setDesignation("manager");

e1.setDepartment(dp);

Employee e2=**new** Employee();

e2.setEid(102);

e2.setName("xyz");

e2.setDesignation("developer");

e2.setDepartment(dp);

dp.getSemp().add(e1);

dp.getSemp().add(e2);

Employee e3=**new** Employee();

e3.setEid(103);

e3.setName("pqr");

e3.setDesignation("QA");

e3.setDepartment(dp1);

dp1.getSemp().add(e3);

session.save(dp);

session.save(dp1);

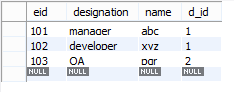
session.beginTransaction().commit();

}

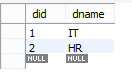
}

Output:

Employee:



Department:



Fetch records:

**import** java.util.Set;

**import** org.hibernate.Session;

**import** com.cjc.entity.Department;

**import** com.cjc.entity.Employee;

**import** com.cjc.utility.HibernateUtil;

**public** **class** RetriveRecordsOnetoMany {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Department dp=session.get(Department.**class**, 1);

System.***out***.println(dp.getDid()+" "+dp.getDname());

Set<Employee> eset=dp.getSemp();

**for**(Employee e:eset) {

System.***out***.println(e.getEid()+" "+e.getName()+" "+e.getDesignation());

}

Employee emp=session.get(Employee.**class**, 101);

System.***out***.println(emp.getEid()+" "+emp.getName()+" "+emp.getDesignation()+" "+emp.getDepartment().getDid()+" "+emp.getDepartment().getDname());

}

}

Note : as its bidirectional by default so we can fetch record from db by using any class object and change hdm2ddl-update

Output:

Hibernate: select department0\_.did as did1\_0\_0\_, department0\_.dname as dname2\_0\_0\_ from Department department0\_ where department0\_.did=?

1 IT

Hibernate: select semp0\_.d\_id as d\_id4\_1\_0\_, semp0\_.eid as eid1\_1\_0\_, semp0\_.eid as eid1\_1\_1\_, semp0\_.d\_id as d\_id4\_1\_1\_, semp0\_.designation as designat2\_1\_1\_, semp0\_.name as name3\_1\_1\_ from Employee semp0\_ where semp0\_.d\_id=?

102 xyz developer

101 abc manager

101 abc manager 1 IT

**3: Many-to-many:**

**package** com.utility;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.cfg.Environment;

**import** com.entity.Course;

**import** com.entity.Student;

**public** **class** HibernateUtil {

**private** **static** StandardServiceRegistry *registry*;

**private** **static** SessionFactory *sessionFactory*;

**public** **static** SessionFactory getSessionFactory() {

**if**(*sessionFactory*==**null**) {

Map<String,Object> settings=**new** HashMap<>();

settings.put(Environment.***DRIVER***, "com.mysql.jdbc.Driver");

settings.put(Environment.***URL***, "jdbc:mysql://localhost:3306/hibernate-cjc-many-to-many");

settings.put(Environment.***USER***, "root");

settings.put(Environment.***PASS***, "root");

settings.put(Environment.***DIALECT***, "org.hibernate.dialect.MySQL5Dialect");

settings.put(Environment.***HBM2DDL\_AUTO***, "create");//change to update for fetching

settings.put(Environment.***SHOW\_SQL***, "true");

*registry*=**new** StandardServiceRegistryBuilder().applySettings(settings).build();

MetadataSources mds=**new** MetadataSources(*registry*);

mds.addAnnotatedClass(Student.**class**);

mds.addAnnotatedClass(Course.**class**);

Metadata md=mds.getMetadataBuilder().build();

*sessionFactory*=md.getSessionFactoryBuilder().build();

}

**return** *sessionFactory*;

}

}

**package** com.entity;

**import** java.util.HashSet;

**import** java.util.Set;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.JoinTable;

**import** javax.persistence.ManyToMany;

@Entity

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

@Id

**private** **int** rollno;

**private** String name;

@ManyToMany(cascade=CascadeType.***ALL***)

@JoinTable(name="student\_course",joinColumns= {@JoinColumn(name="rno")},inverseJoinColumns= {@JoinColumn(name="cid")})

**private** Set<Course> setcourses=**new** HashSet<>();

**public** **int** getRollno() {

**return** rollno;

}

**public** **void** setRollno(**int** rollno) {

**this**.rollno = rollno;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** Set<Course> getSetcourses() {

**return** setcourses;

}

**public** **void** setSetcourses(Set<Course> setcourses) {

**this**.setcourses = setcourses;

}

}

**package** com.entity;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

@Entity

**public** **class** Course {

@Id

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

**private** String coursename;

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

**return** id;

}

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

**this**.id = id;

}

**public** String getCoursename() {

**return** coursename;

}

**public** **void** setCoursename(String coursename) {

**this**.coursename = coursename;

}

}

**package** com.entity;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

@Entity

**public** **class** Course {

@Id

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

**private** String coursename;

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

**return** id;

}

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

**this**.id = id;

}

**public** String getCoursename() {

**return** coursename;

}

**public** **void** setCoursename(String coursename) {

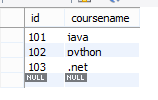
**this**.coursename = coursename;

}

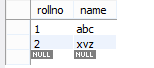
}

Output:

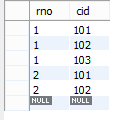
Course table:



Student table:



Student\_course table:



**Cache Mechanism:**

First level cache:

* First level cache is session cache.
* It is mandatory cache through which all element must pass.
* First level cache in hibernate is enabled by default.
* We cannot disable first level cache forcefully.
* First level cache is associated with session object.
* Session object is created from session factory and once the session is closed then object which are in session also removed.
* When we query an entity first time it is remove from db and stored in first level cache associated with hibernate session.
* The loaded entity can be removed from session using evict() method
* The whole session cache can be removed using clear() method, it will remove all stored in cache .

**package** com.cjc.app;

**import** org.hibernate.Session;

**import** com.cjc.entity.Employee;

**import** com.cjc.utility.HibernateUtil;

**public** **class** Test2 {

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

Session session = HibernateUtil.*getSessionFactory*().openSession();

Employee e1 = session.get(Employee.**class**, 1);

System.***out***.println(e1.getEid() + " " + e1.getEname() + " " + e1.getDname());

Session session1 = HibernateUtil.*getSessionFactory*().openSession();

Employee e2 = session1.get(Employee.**class**, 1);

System.***out***.println(e2.getEid() + " " + e2.getEname() + " " + e2.getDname());

/\*Employee e3 = session1.get(Employee.class, 2);

System.out.println(e3.getEid() + " " + e3.getEname() + " " + e3.getDname());

\*/

}

}

Hibernate: select employee0\_.eid as eid1\_0\_0\_, employee0\_.dname as dname2\_0\_0\_, employee0\_.ename as ename3\_0\_0\_ from Employee employee0\_ where employee0\_.eid=?

1 qwerty sdfg

Hibernate: select employee0\_.eid as eid1\_0\_0\_, employee0\_.dname as dname2\_0\_0\_, employee0\_.ename as ename3\_0\_0\_ from Employee employee0\_ where employee0\_.eid=?

1 qwerty sdfg

In First level cache if there are one or more session and try to fetch records then it will fire hql select query that number of times i.e it hits db for every time.

**Second level cache:**

* Second level cache is created in session factory scope.
* Second level cache is available for all session created using that session factory.
* Once session factory is closed all cache associated with will die.
* Whenever hibernate session try to load an entity it first looks for cached copy of entity In first level cache.
* If there is no cache entity in first level cache, then Second level cache is looked up for cache entity.
* If the second level cache has cached entity, it returns the cached entity and stores the same entity in first level cache too.
* When the request from same entity comes it retrieves from first level cache instead of hitting second level cache.

package com.cjc.utility;

import java.util.HashMap;

import java.util.Map;

import org.hibernate.SessionFactory;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Environment;

import com.cjc.entity.Employee;

public class HibernateUtil {

private static StandardServiceRegistry registry;

private static SessionFactory sessionFactory;

public static SessionFactory getSessionFactory() {

if(sessionFactory==null) {

Map<String,Object> settings=new HashMap<>();

settings.put(Environment.DRIVER, "com.mysql.jdbc.Driver");

settings.put(Environment.URL, "jdbc:mysql://localhost:3306/cjc");

settings.put(Environment.USER, "root");

settings.put(Environment.PASS, "root");

settings.put(Environment.DIALECT, "org.hibernate.dialect.MySQL5Dialect");

settings.put(Environment.HBM2DDL\_AUTO, "update");

settings.put(Environment.SHOW\_SQL, "true");

settings.put(Environment.USE\_SECOND\_LEVEL\_CACHE, "true");

settings.put(Environment.CACHE\_REGION\_FACTORY, "org.hibernate.cache.ehcache.internal.EhcacheRegionFactory");

registry=new StandardServiceRegistryBuilder().applySettings(settings).build();

MetadataSources mds=new MetadataSources(registry);

mds.addAnnotatedClass(Employee.class);

Metadata md=mds.getMetadataBuilder().build();

sessionFactory=md.getSessionFactoryBuilder().build();

}

return sessionFactory;

}

}

**package** com.cjc.entity;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** org.hibernate.annotations.Cache;

**import** org.hibernate.annotations.CacheConcurrencyStrategy;

@Entity

@Cache(usage=CacheConcurrencyStrategy.***READ\_WRITE***)

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

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

**private** **int** eid;

**private** String ename;

**private** String dname;

**public** **int** getEid() {

**return** eid;

}

**public** **void** setEid(**int** eid) {

**this**.eid = eid;

}

**public** String getEname() {

**return** ename;

}

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

**this**.ename = ename;

}

**public** String getDname() {

**return** dname;

}

**public** **void** setDname(String dname) {

**this**.dname = dname;

}

}

**package** com.cjc.app;

**import** org.hibernate.Session;

**import** com.cjc.entity.Employee;

**import** com.cjc.utility.HibernateUtil;

**public** **class** Test2 {

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

Session session = HibernateUtil.*getSessionFactory*().openSession();

Employee e1 = session.get(Employee.**class**, 1);

System.***out***.println(e1.getEid() + " " + e1.getEname() + " " + e1.getDname());

Session session1 = HibernateUtil.*getSessionFactory*().openSession();

Employee e2 = session1.get(Employee.**class**, 1);

System.***out***.println(e2.getEid() + " " + e2.getEname() + " " + e2.getDname());

/\*Employee e3 = session1.get(Employee.class, 2);

System.out.println(e3.getEid() + " " + e3.getEname() + " " + e3.getDname());

\*/

}

}

Output:

Hibernate: select employee0\_.eid as eid1\_0\_0\_, employee0\_.dname as dname2\_0\_0\_, employee0\_.ename as ename3\_0\_0\_ from Employee employee0\_ where employee0\_.eid=?

1 qwerty sdfg

1 qwerty sdfg

**Stored Procedure:**

CREATE DEFINER=`root`@`localhost` PROCEDURE `mobile\_count`(in mid integer,Out mcount integer)

BEGIN

select count(\*) into mcount from mobile where m\_id=mid;

END

Testclass:

**package** com.cjc.client;

**import** javax.persistence.ParameterMode;

**import** javax.persistence.StoredProcedureQuery;

**import** org.hibernate.Session;

**import** com.cjc.utility.HibernateUtil;

**public** **class** TestProcudure {

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

//stored procedure using hibernate

Session session=HibernateUtil.*getSessionFactory*().openSession();

StoredProcedureQuery spq=session.createStoredProcedureQuery("mobile\_count");

spq.registerStoredProcedureParameter("mid", Integer.**class**, ParameterMode.***IN***);

spq.registerStoredProcedureParameter("mcount",Integer.**class**, ParameterMode.***OUT***);

spq.setParameter("mid", 1);

spq.execute();

System.***out***.println(spq.getOutputParameterValue("mcount"));

}

}

Output:

Hibernate: {call mobile\_count(?,?)}

2

**Inheritance Relationship (IS-A):**

**1. Default Inheritance:**

Cjc.java:

**package** com.entity.Default;

**import** javax.persistence.Id;

**import** javax.persistence.MappedSuperclass;

@MappedSuperclass

**public** **class** Cjc {

@Id

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

**private** String cname;

**private** String sname;

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

**return** id;

}

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

**this**.id = id;

}

**public** String getCname() {

**return** cname;

}

**public** **void** setCname(String cname) {

**this**.cname = cname;

}

**public** String getSname() {

**return** sname;

}

**public** **void** setSname(String sname) {

**this**.sname = sname;

}

}

Akurdi.java:

**package** com.entity.Default;

**import** javax.persistence.Entity;

@Entity

**public** **class** Akurdi **extends** Cjc{

**private** String weekendbatch;

**public** String getWeekendbatch() {

**return** weekendbatch;

}

**public** **void** setWeekendbatch(String weekendbatch) {

**this**.weekendbatch = weekendbatch;

}

}

Karvenager.java:

**package** com.entity.Default;

**import** javax.persistence.Entity;

@Entity

**public** **class** Karvenagar **extends** Cjc{

**private** String regularBatch;

**public** String getRegularBatch() {

**return** regularBatch;

}

**public** **void** setRegularBatch(String regularBatch) {

**this**.regularBatch = regularBatch;

}

}

In HibernateUtil.java add following lines:

mds.addAnnotatedClass(Cjc.**class**);

mds.addAnnotatedClass(Akurdi.**class**);

mds.addAnnotatedClass(Karvenagar.**class**);

Test.java:

**package** com.client;

**import** org.hibernate.Session;

**import** com.entity.Default.Akurdi;

**import** com.entity.Default.HibernateUtil;

**import** com.entity.Default.Karvenagar;

//default inheritance

**public** **class** TestISA {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Akurdi ak=**new** Akurdi();

ak.setId(1);

ak.setSname("abc");

ak.setCname("java");

ak.setWeekendbatch("B113");

Karvenagar ka=**new** Karvenagar();

ka.setId(101);

ka.setSname("xyz");

ka.setCname("spring");

ka.setRegularBatch("b115");

session.save(ka);

session.save(ak);

session.beginTransaction().commit();

}

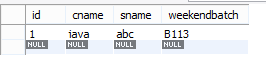
}

Output:

Karvenager Table:



Akurdi table:



2. SingleTable:

Cjc.java:

**package** com.entity.SingleTable;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Inheritance;

**import** javax.persistence.InheritanceType;

@Inheritance(strategy=InheritanceType.***SINGLE\_TABLE***)

@Entity

**public** **class** Cjc {

@Id

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

**private** String cname;

**private** String sname;

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

**return** id;

}

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

**this**.id = id;

}

**public** String getCname() {

**return** cname;

}

**public** **void** setCname(String cname) {

**this**.cname = cname;

}

**public** String getSname() {

**return** sname;

}

**public** **void** setSname(String sname) {

**this**.sname = sname;

}

}

Akurdi.java

**package** com.entity.SingleTable;

**import** javax.persistence.Entity;

@Entity

**public** **class** Akurdi **extends** Cjc{

**private** String weekendbatch;

**public** String getWeekendbatch() {

**return** weekendbatch;

}

**public** **void** setWeekendbatch(String weekendbatch) {

**this**.weekendbatch = weekendbatch;

}

}

Karvenagar.java:

**package** com.entity.SingleTable;

**import** javax.persistence.Entity;

@Entity

**public** **class** Karvenagar **extends** Cjc{

**private** String regularBatch;

**public** String getRegularBatch() {

**return** regularBatch;

}

**public** **void** setRegularBatch(String regularBatch) {

**this**.regularBatch = regularBatch;

}

}

In HibernateUtil.java add following lines:

mds.addAnnotatedClass(Cjc.**class**);

mds.addAnnotatedClass(Akurdi.**class**);

mds.addAnnotatedClass(Karvenagar.**class**);

Test.java

**package** com.client;

**import** org.hibernate.Session;

**import** com.entity.SingleTable.Akurdi;

**import** com.entity.SingleTable.HibernateUtil;

**import** com.entity.SingleTable.Karvenagar;

//singletable

**public** **class** TestISASingleTable {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Akurdi ak=**new** Akurdi();

ak.setId(1);

ak.setSname("abc");

ak.setCname("java");

ak.setWeekendbatch("B113");

Karvenagar ka=**new** Karvenagar();

ka.setId(101);

ka.setSname("xyz");

ka.setCname("spring");

ka.setRegularBatch("b115");

session.save(ka);

session.save(ak);

session.beginTransaction().commit();

}

}

Output:

Cjc table:



3. Joined table:

**package** com.entity.Joined;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Inheritance;

**import** javax.persistence.InheritanceType;

@Inheritance(strategy=InheritanceType.***JOINED***)

@Entity

**public** **class** Cjc {

@Id

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

**private** String cname;

**private** String sname;

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

**return** id;

}

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

**this**.id = id;

}

**public** String getCname() {

**return** cname;

}

**public** **void** setCname(String cname) {

**this**.cname = cname;

}

**public** String getSname() {

**return** sname;

}

**public** **void** setSname(String sname) {

**this**.sname = sname;

}

}

Akurdi.java:

**package** com.entity.Joined;

**import** javax.persistence.Entity;

@Entity

**public** **class** Akurdi **extends** Cjc{

**private** String weekendbatch;

**public** String getWeekendbatch() {

**return** weekendbatch;

}

**public** **void** setWeekendbatch(String weekendbatch) {

**this**.weekendbatch = weekendbatch;

}

}

Karvenagar.java:

**package** com.entity.Joined;

**import** javax.persistence.Entity;

@Entity

**public** **class** Karvenagar **extends** Cjc{

**private** String regularBatch;

**public** String getRegularBatch() {

**return** regularBatch;

}

**public** **void** setRegularBatch(String regularBatch) {

**this**.regularBatch = regularBatch;

}

}

Test.java:

**package** com.client;

**import** org.hibernate.Session;

**import** com.entity.Joined.Akurdi;

**import** com.entity.Joined.HibernateUtil;

**import** com.entity.Joined.Karvenagar;

**public** **class** TestJoined {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Akurdi ak=**new** Akurdi();

ak.setId(1);

ak.setSname("abc");

ak.setCname("java");

ak.setWeekendbatch("B113");

Karvenagar ka=**new** Karvenagar();

ka.setId(101);

ka.setSname("xyz");

ka.setCname("spring");

ka.setRegularBatch("b115");

session.save(ka);

session.save(ak);

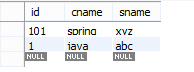
session.beginTransaction().commit();

}

}

Output:

Cjc table:



Karvenager.java



Akurdi.java



4.Table per class:

Cjc.java:

package com.entity.TablePerClass;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Inheritance;

import javax.persistence.InheritanceType;

@Inheritance(strategy=InheritanceType.TABLE\_PER\_CLASS)

@Entity

public class Cjc {

@Id

private int id;

private String cname;

private String sname;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getCname() {

return cname;

}

public void setCname(String cname) {

this.cname = cname;

}

public String getSname() {

return sname;

}

public void setSname(String sname) {

this.sname = sname;

}

}

Akurdi.java

**package** com.entity.TablePerClass;

**import** javax.persistence.Entity;

@Entity

**public** **class** Akurdi **extends** Cjc{

**private** String weekendbatch;

**public** String getWeekendbatch() {

**return** weekendbatch;

}

**public** **void** setWeekendbatch(String weekendbatch) {

**this**.weekendbatch = weekendbatch;

}

}

Karvenagar.java:

**package** com.entity.TablePerClass;

**import** javax.persistence.Entity;

@Entity

**public** **class** Karvenagar **extends** Cjc{

**private** String regularBatch;

**public** String getRegularBatch() {

**return** regularBatch;

}

**public** **void** setRegularBatch(String regularBatch) {

**this**.regularBatch = regularBatch;

}

}

Test.java:

**package** com.client;

**import** org.hibernate.Session;

**import** com.entity.TablePerClass.Akurdi;

**import** com.entity.TablePerClass.HibernateUtil;

**import** com.entity.TablePerClass.Karvenagar;

//Tableperclass

**public** **class** TestTablePerClass {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

Akurdi ak=**new** Akurdi();

ak.setId(1);

ak.setSname("abc");

ak.setCname("java");

ak.setWeekendbatch("B113");

Karvenagar ka=**new** Karvenagar();

ka.setId(2);

ka.setSname("xyz");

ka.setCname("spring");

ka.setRegularBatch("b115");

session.save(ka);

session.save(ak);

session.beginTransaction().commit();

}

}

Output:

Cjc table



Akurdi.java



Karvenagar.java



|  |  |
| --- | --- |
| SQL (native query) | HQL(oop query) |
| It Is non-object oriented query | It is a object oriented query |
| Native query are depend on table name and column name | HQL query depend on entity class name and variables. |
| Native query are fast | HQL Query are slow because it internally convert in native query. |
| Native query is difficult to construct and maintenance | HQL Query is easy to construct and maintenance |

|  |  |
| --- | --- |
| HQL | Criteria Builder |
| It is a query language we need to write some special query | Criteria builder have only interface, class, methods |
| Through HQL query we can perform select as well as non-select query(insert,delete,update) | Through criteria builder we can perform only select query |

HQL Query:

For insert records:

**package** com.Hql;

**import** org.hibernate.Session;

**import** com.cjc.utiltiy.HibernateUtil;

**import** com.entity.Employee;

**public** **class** TestInsert {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

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

emp.setName("xyz");

emp.setAddress("pune");

emp.setSalary(100000.0);

session.save(emp);

session.beginTransaction().commit();

}

}

Hibernate: insert into Employee (address, name, salary) values (?, ?, ?)

**For fetch records:**

import java.util.List;

import org.hibernate.Session;

import com.cjc.utiltiy.HibernateUtil;

import com.entity.Employee;

public class FetchData {

public static void main(String[] args) {

Session session=HibernateUtil.getSessionFactory().openSession();

@SuppressWarnings("unchecked")

List<Employee> elist=session.createQuery("from Employee").getResultList();

for(Employee e:elist) {

System.out.println(e.getEmpid()+" "+e.getName()+" "+e.getAddress()+" "+e.getSalary());

}

//Query query=session.createQuery("from Employee where")

}

}

Hibernate: select employee0\_.empid as empid1\_0\_, employee0\_.address as address2\_0\_, employee0\_.name as name3\_0\_, employee0\_.salary as salary4\_0\_ from Employee employee0\_

1 xyz pune 100000.0

2 bvfd mumbai 7845184.0

4 xyz pune 100000.0

Update records:

package com.Hql;

import javax.persistence.Query;

import org.hibernate.Session;

import org.hibernate.Transaction;

import com.cjc.utiltiy.HibernateUtil;

public class UpdateRecord {

public static void main(String[] args) {

Session session = HibernateUtil.getSessionFactory().openSession();

session.getTransaction().begin();

//Transaction tx=session.beginTransaction();

Query query = session.createQuery("update Employee set name=:ename" +

" where empid=:empid");

query.setParameter("ename", "rrr");

query.setParameter("empid", 4);

int rs = query.executeUpdate();

//tx.commit();

session.getTransaction().commit();

}

}

Hibernate: update Employee set name=? where empid=?

Delete records:

package com.Hql;

import javax.persistence.Query;

import org.hibernate.Session;

import com.cjc.utiltiy.HibernateUtil;

public class DeleteRecord {

public static void main(String[] args) {

Session session = HibernateUtil.getSessionFactory().openSession();

session.getTransaction().begin();

Query query=session.createQuery("delete from Employee where empid=:id");

query.setParameter("id", 4);

int rs=query.executeUpdate();

session.getTransaction().commit();

}

}

Hibernate: delete from Employee where empid=?

Named Query:

* Instead of writing multiple places native query or hql query , we can in single place which is entity class.
* Because of Named query sql and hql query will not be scattered.
* It is easy to use and maintain.
* There are 4 annotation

* @NamedQuery
* @NamedQueries
* @NamedNativeQuery
* @NamedNativeQueries

Employee.java

**package** com.entity;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.NamedQuery;

@NamedQuery(name="all\_emp",query="from Employee")

@Entity

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

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

**private** **int** empid;

**private** String name;

**private** String address;

**private** **double** salary;

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** **int** getEmpid() {

**return** empid;

}

}

Test.java

package com.client;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.query.Query;

import com.cjc.utiltiy.HibernateUtil;

import com.entity.Employee;

public class TestNamedQuery {

public static void main(String[] args) {

Session session=HibernateUtil.getSessionFactory().openSession();

Query q=session.createNamedQuery("all\_emp");

List<Employee> list=q.getResultList();

for(Employee e:list) {

System.out.println(e.getEmpid()+" "+e.getName()+" "+e.getAddress()+" "+e.getSalary());

}

}

}

Output:

Hibernate: select employee0\_.empid as empid1\_0\_, employee0\_.address as address2\_0\_, employee0\_.name as name3\_0\_, employee0\_.salary as salary4\_0\_ from Employee employee0\_

1 xyz pune 100000.0

2 bvfd mumbai 7845184.0

Creating Native Query:

Employee.java

**package** com.entity;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.NamedNativeQuery;

@Entity

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

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

**private** **int** empid;

**private** String name;

**private** String address;

**private** **int** salary;

**public** **int** getSalary() {

**return** salary;

}

**public** **void** setSalary(**int** salary) {

**this**.salary = salary;

}

**public** **int** getEmpid() {

**return** empid;

}

**public** **void** setEmpid(**int** empid) {

**this**.empid = empid;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

}

package com.app;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.query.Query;

import com.entity.Employee;

import com.utility.HibernateUtil;

public class CreatingNativeQuery {

public static void main(String[] args) {

Session session=HibernateUtil.getSessionFactory().openSession();

Query<Employee> query=session.createNativeQuery("select \* from employee",Employee.class);

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

for(Employee e:list) {

System.out.println(e.getEmpid()+" "+e.getName()+ " "+e.getAddress()+" "+e.getSalary());

}

}

}

Output:

Hibernate: select \* from employee

1 abc pune 5000

2 xyz pune 2000

Creating NamedNativeQuery:

Employee.java

**package** com.entity;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.NamedNativeQuery;

@NamedNativeQuery(name="e\_all",query="select \* from Employee",resultClass=Employee.**class**)

@Entity

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

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

**private** **int** empid;

**private** String name;

**private** String address;

**private** **int** salary;

**public** **int** getSalary() {

**return** salary;

}

**public** **void** setSalary(**int** salary) {

**this**.salary = salary;

}

**public** **int** getEmpid() {

**return** empid;

}

**public** **void** setEmpid(**int** empid) {

**this**.empid = empid;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

}

Test.java

package com.app;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.query.Query;

import com.entity.Employee;

import com.utility.HibernateUtil;

public class CreatingNamedQuery {

public static void main(String[] args) {

Session session=HibernateUtil.getSessionFactory().openSession();

Query<Employee> query=session.createNamedQuery("e\_all",Employee.class);

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

for(Employee e:list) {

System.out.println(e.getEmpid()+" "+e.getName()+ " "+e.getAddress()+" "+e.getSalary());

}

}

}

Hibernate: select \* from Employee

1 abc pune 5000

2 xyz pune 2000

Criteria Builder:

1. select \* from employee:

**package** com.app;

**import** java.util.List;

**import** javax.persistence.criteria.CriteriaBuilder;

**import** javax.persistence.criteria.CriteriaQuery;

**import** javax.persistence.criteria.Root;

**import** org.hibernate.Session;

**import** org.hibernate.query.Query;

**import** com.entity.Employee;

**import** com.utility.HibernateUtil;

**public** **class** CriteriaBuilderAppDemo {

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

Session session=HibernateUtil.*getSessionFactory*().openSession();

CriteriaBuilder builder=session.getCriteriaBuilder();

CriteriaQuery<Employee> cQuery=builder.createQuery(Employee.**class**);

Root<Employee> root=cQuery.from(Employee.**class**);

cQuery.select(root);

//cQuery.orderBy(builder.asc(root.get("salary")));

Query query=session.createQuery(cQuery);

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

**for**(Employee e:list) {

System.***out***.println(e.getEmpid()+" "+e.getName()+" "+e.getAddress()+" "+e.getSalary());

}

}

}

Hibernate: select employee0\_.empid as empid1\_0\_, employee0\_.address as address2\_0\_, employee0\_.name as name3\_0\_, employee0\_.salary as salary4\_0\_ from Employee employee0\_

1 abc pune 5000

2 xyz pune 2000

1. select from employee where id=1;

package com.app;

import java.util.List;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import org.hibernate.Session;

import org.hibernate.query.Query;

import com.entity.Employee;

import com.utility.HibernateUtil;

public class CriteriaBuilderAppDemo2 {

public static void main(String[] args) {

Session session = HibernateUtil.getSessionFactory().openSession();

CriteriaBuilder builder = session.getCriteriaBuilder();

CriteriaQuery<Employee> cQuery = builder.createQuery(Employee.class);

Root<Employee> root = cQuery.from(Employee.class);

cQuery.select(root).where(builder.equal(root.get("empid"), 1));

Query query = session.createQuery(cQuery);

Employee e = (Employee) query.getSingleResult();

System.out.println(e.getEmpid() + " " + e.getName() + " " + e.getAddress());

}

}

Hibernate: select employee0\_.empid as empid1\_0\_, employee0\_.address as address2\_0\_, employee0\_.name as name3\_0\_, employee0\_.salary as salary4\_0\_ from Employee employee0\_ where employee0\_.empid=1

1 abc pune

1. select name from employee:

package com.app;

import java.util.List;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import org.hibernate.Session;

import org.hibernate.query.Query;

import com.entity.Employee;

import com.utility.HibernateUtil;

public class CriteriaBuilderAppDemo3 {

public static void main(String[] args) {

Session session = HibernateUtil.getSessionFactory().openSession();

CriteriaBuilder builder = session.getCriteriaBuilder();

CriteriaQuery<String> cQuery = builder.createQuery(String.class);

Root<Employee> root = cQuery.from(Employee.class);

/\*cQuery.select(root).where(builder.equal(root.get("name"), "abc"));

\*/

cQuery.select(root.get("name"));

Query query = session.createQuery(cQuery);

List<String> list = query.getResultList();

for(String nm:list) {

System.out.println(nm);

}

}

}

Hibernate: select employee0\_.name as col\_0\_0\_ from Employee employee0\_

abc

xyz

1. select name. address from employee

import java.util.List;

import javax.persistence.criteria.CriteriaBuilder;

import javax.persistence.criteria.CriteriaQuery;

import javax.persistence.criteria.Root;

import org.hibernate.Session;

import org.hibernate.query.Query;

import com.entity.Employee;

import com.utility.HibernateUtil;

public class CriteriaBuilderAppDemo4 {

public static void main(String[] args) {

Session session = HibernateUtil.getSessionFactory().openSession();

CriteriaBuilder builder = session.getCriteriaBuilder();

CriteriaQuery<Object[]> cQuery = builder.createQuery(Object[].class);

Root<Employee> root = cQuery.from(Employee.class);

cQuery.multiselect(root.get("name"),root.get("address"));

List<Object[]> list = session.createQuery(cQuery).getResultList();

for(Object[] obj:list) {

System.out.println(obj[0]+" "+obj[1]);

}

}

}

Hibernate: select employee0\_.name as col\_0\_0\_, employee0\_.address as col\_1\_0\_ from Employee employee0\_

abc pune

xyz pune