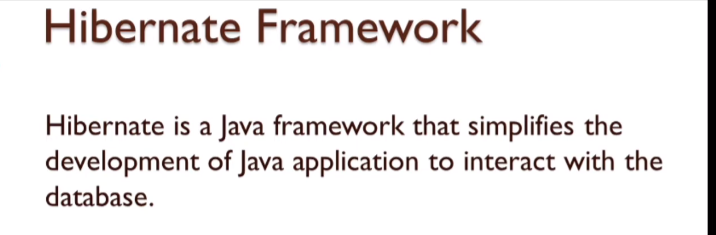
Hibernate



Diffrence between jdbc api and hibernate :

* Hibernate is database independent because it use ORM(Object Relational mapping) Tool to data maping
* Hibernate internally used jdbc API
* Jdbc is database independent

🡪Project Setup

i)create a marven project

ii)go to the mvnrepositand add dependencies a)hibernate b)mysql

iii)config hibernate

A)by xml

i)creating a new xml file with the name – hibernate.cfg.xml

ii)config xml :

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

<!**DOCTYPE** hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

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

<**hibernate-configuration**>

<**session-factory**>

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

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

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

<**property** name=*"connection.password"*>Ajaj13994010#</**property**>

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

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

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

</**session-factory**>

</**hibernate-configuration**>

Creating a table : using java config

Student table : using Entity directive

package com.becoder;

import javax.persistence.Entity;

import javax.persistence.Id;

*@Entity*

public class Student {

// to create a primary key

*@Id*

private int id;

private String name;

private String email;

private String address;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public Student() {

super();

// **TODO** Auto-generated constructor stub

}

public Student(int id, String name, String email, String address) {

super();

this.id = id;

this.name = name;

this.email = email;

this.address = address;

}

}

HibernateUtil.java config file

package com.becoder;

import java.util.Properties;

import org.hibernate.SessionFactory;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

import org.hibernate.cfg.Environment;

import org.hibernate.service.ServiceRegistry;

public class HibernateUtil {

        private static *SessionFactory* sessionFactory;

        public static *SessionFactory* getSessionFactory() {

            if(sessionFactory==null) {

*Configuration* cfg=new Configuration();

*Properties* properties=new Properties();

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

                properties.put(Environment.URL,"jdbc:mysql://localhost:3306/hibernatedatabase");

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

                properties.put(Environment.PASS, "Ajaj13994010#");

//              properties.put(Environment.DIALECT,"org.hibernate.dialect.MYSQL5Dialect");

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

                properties.put(Environment.FORMAT\_SQL, true);

                properties.put(Environment.SHOW\_SQL, true);

                cfg.setProperties(properties);

                // creating a table

                cfg.addAnnotatedClass(Student.class);

*ServiceRegistry* sessionRegistry= new StandardServiceRegistryBuilder().applySettings(cfg.getProperties()).build();

                sessionFactory= cfg.buildSessionFactory(sessionRegistry);

            }

            return sessionFactory;

        }

}

🡪insert data into table

package com.becoder;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

/\*\*

 \* Hello world!

 \*

 \*/

public class App

{

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

    {

        System.out.println( "Hello World!" );

//        Code for hibernate xml configration

//        Configuration cfg=new Configuration();

//        cfg.configure();

//

//      SessionFactory factory= cfg.buildSessionFactory();

//

//      System.out.println(factory);

*Student* st=new Student();

        st.setId(1);

        st.setName("Ajaj Alam");

        st.setEmail("ajajalam3811@gmail.com");

        st.setAddress("Bettiah,West Champaran, Bihar");

*SessionFactory* factory= HibernateUtil.getSessionFactory();

*Session* session= factory.openSession();

*Transaction* transaction= session.beginTransaction();

        session.save(st);

        transaction.commit();

        session.close();

    }

}

🡪Commonly used hibernate anotation :

Import javax.presistence.\*

[1.@Entity](mailto:1.@Entity) - it is used to create a table , it is perform at class level

[2.@Table](mailto:2.@Table) -it is used at table level to perform operation on table like give defautl table name : @Table(name=”student”)

[3.@Id](mailto:3.@Id) – it is used to create aprimary key

[4.@GnetratedValue](mailto:4.@GnetratedValue) – it is used for auto increament

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

[5.@Column](mailto:5.@Column) – it is used to perform opertion at column level eg

*@Column*(nullable = false,unique = true,length = 15)

[6.@Transient](mailto:6.@Transient) -- it is ued to notify hibernate not to create a column with that function

[7.@Temporal](mailto:7.@Temporal) – it is used for formating data

*@Temporal*(*TemporalType*.***TIMESTAMP***)

[8.@Lob](mailto:8.@Lob) – it is used for large object data

[9.@OneToOne](mailto:9.@OneToOne)

[10.@OneToMany](mailto:10.@OneToMany)

[11.@ManyToOne](mailto:11.@ManyToOne)

[12.@ManyToMany](mailto:12.@ManyToMany)

[13.@JoinColumn](mailto:13.@JoinColumn)

🡪Fetching data from database using Get Or Load Method L

1. By sessoin.get method
2. SessionFactory factory= HibernateUtil.*getSessionFactory*();
4. Session session= factory.openSession();
6. Student data= session.get(Student.class, 1);
8. session.close();
10. System.***out***.println(data);

2.By sesson.laod metod

SessionFactory factory= HibernateUtil.*getSessionFactory*();

Session session= factory.openSession();

Student data= session.load(Student.class, 1);

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

session.close();

* Diffrence between get and load – if data not found , then if we are using get mentod then it will not going to give use exception it will going to give us a null object , but if we are using load metod it will going to give excption objectNotFound and load mtod will going to fire only when we calll the object because it return a proxy object.

🡪Reading all the data

List<Student> data= session.createQuery("from Student",Student.class).list();

🡪Updating data – first get data then update using session.saveorUpdata

SessionFactory factory= HibernateUtil.*getSessionFactory*();

Session session= factory.openSession();

Student st=session.get(Student.class, 1);

st.setName("Ajaj");

Transaction trx= session.beginTransaction();

session.saveOrUpdate(st);

trx.commit();

session.close();

factory.close();

🡪Delete data

SessionFactory factory= HibernateUtil.*getSessionFactory*();

Session session= factory.openSession();

Student st=session.get(Student.class, 1);

Transaction trx= session.beginTransaction();

session.delete(st);

trx.commit();

session.close();

factory.close();

One to one hibernate mapping relationship :

i)we make do maping in tow direction :

A)unidirectional

i)Employe class

package com.relationship;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.OneToOne;

*@Entity*

public class Employe {

*@Id*

private int id;

private String name;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Adress getAdress() {

return adress;

}

public void setAdress(Adress adress) {

this.adress = adress;

}

*@OneToOne*

private Adress adress;

}

ii)Adress class

package com.relationship;

import javax.persistence.Entity;

import javax.persistence.Id;

*@Entity*

public class Adress {

*@Id*

private int id;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

private String address;

}

iii)hibernateutil class

iv)AppMain class

package com.relationship;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

public class AppMain {

public static void main(String[] args) {

//insert data into address table

// Adress ad=new Adress();

// ad.setId(101);

// ad.setAddress("odisha");

//

// // inserting data into employee table

// Employe em=new Employe();

// em.setId(201);

// em.setName("Ajaj");

// em.setAdress(ad);

//

SessionFactory factory= HibernateUtil.*getSessionFactory*();

Session session= factory.openSession();

Transaction tnx= session.beginTransaction();

// session.save(ad);

// session.save(em);

Employe emp= session.get(Employe.class, 201);

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

System.***out***.println(emp.getAdress().getAddress());

tnx.commit();

session.close();

factory.close();

}

}

B)Bidirectional

i)adress table

package com.relationship;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.OneToOne;

*@Entity*

public class Adress {

*@Id*

private int id;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public Employe getEmp() {

return emp;

}

public void setEmp(Employe emp) {

this.emp = emp;

}

private String address;

*@OneToOne*(mappedBy = "address")

private Employe emp;

🡪Many to one mapping :

1.Employe table class

package com.onetomany;

import java.util.List;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.OneToMany;

*@Entity*

public class Employe {

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public List<Adress> getAdress() {

return adress;

}

public void setAdress(List<Adress> adress) {

this.adress = adress;

}

*@Id*

private int id;

private String name;

*@OneToMany*(mappedBy = "emp")

private List<Adress> adress;

}

2.adress table class

package com.onetomany;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.ManyToOne;

*@Entity*

public class Adress {

*@Id*

private int id;

private String Adress;

private String AdressType;

*@ManyToOne*

private Employe emp;

public Employe getEmp() {

return emp;

}

public void setEmp(Employe emp) {

this.emp = emp;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getAdress() {

return Adress;

}

public void setAdress(String adress) {

Adress = adress;

}

public String getAdressType() {

return AdressType;

}

public void setAdressType(String adressType) {

AdressType = adressType;

}

public Adress() {

super();

// **TODO** Auto-generated constructor stub

}

public Adress(int id, String adress, String adressType) {

super();

this.id = id;

Adress = adress;

AdressType = adressType;

}

}

3. AppMain.java

package com.onetomany;

import java.util.ArrayList;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

public class AppMain {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Adress ad1=new Adress(101, "Delhi", "Permanent");

Adress ad2=new Adress(102, "Mumbai", "Office");

List<Adress> ls=new ArrayList<>();

ls.add(ad1);

ls.add(ad2);

Employe em=new Employe();

em.setId(201);

em.setAdress(ls);

em.setName("Ajaj");

SessionFactory factory= HibernateUtil.*getSessionFactory*();

Session session= factory.openSession();

Transaction trnx=session.beginTransaction();

session.save(ad1);

session.save(ad2);

session.save(em);

trnx.commit();

session.close();

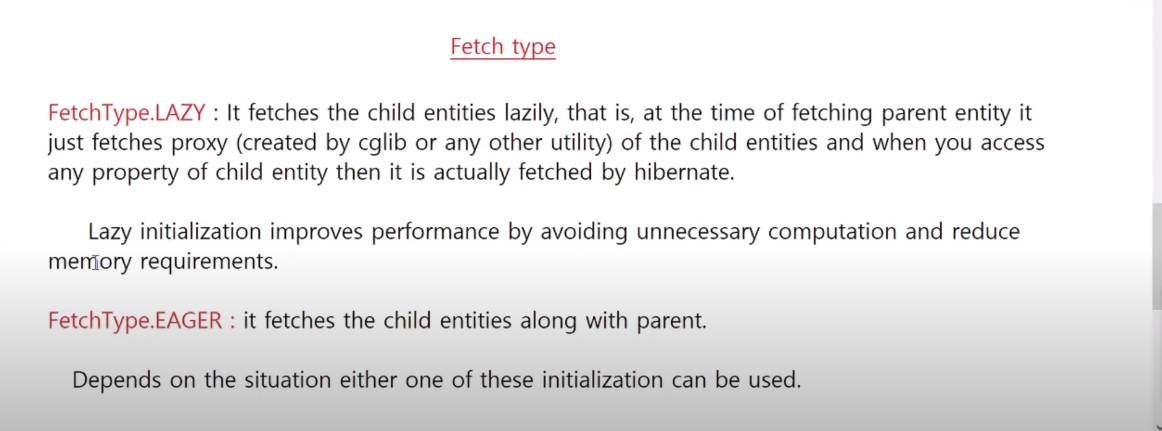
factory.close();

}

}

🡺Many to many hibernate RelationShip

🡪Fetching in Hibernate :



By default we have fetch type : Lazy



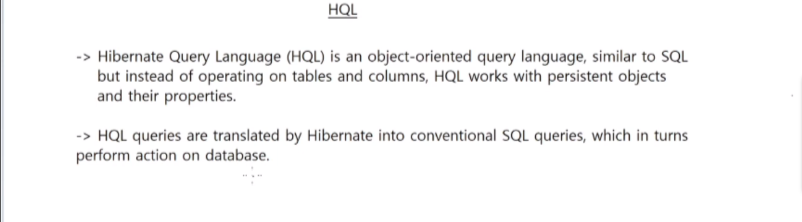
Cascade in Hibernate :

With the help of cascade what we ensure is that when ever we perform any opertaion in parent entity the all the operation related to the child enetity should excute automatically.

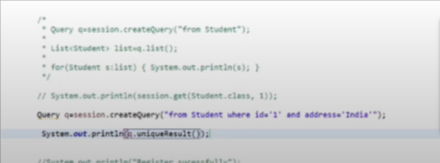
Way to defnie cascadeType :



HQL🡪



Geeting data :



Passing dynamic parameter :



Using alias



Deleting data :



Updateing data:

