hibernate：简化对数据库的操作

生成方式：1.手动

导jar包；写hibernate.config.xml配置文件；

写xx.hbm.xml（将实体类与数据库表建立映射关系）

主键生成策略：native或者increacement或者sequence

使用：7步（除了查询，其它都要提交事务）

1.加载总配置文件

Configuration conf=**new** Configuration().configure("hibernate.config.xml");

//2.获取SessionFactory

SessionFactory factory=conf.buildSessionFactory();

//3.获取session

Session session=factory.openSession();

//4.开启事务

Transaction ts=session.beginTransaction();

//5.执行session方法

//保存对象

session.save(dept);

//查询对象

dept2=(Dept)session.get(Dept.**class**, 10);

//修改对象session.update(dept);

//删除对象session.delete(dept);

//6.提交事务ts.commit();

//7.关闭sessions

session.close();

hibernate对象状态（新生，持久，游离）

new

delete

close()

get/load

save/update

save/update

持久

游离

新生

对象

hql语句：用于按条件查询，是面向对象的查询

查所有：from 类名 （where +条件）

Query query=session. createQuery(hql);

list=quert.list();

之后可以遍历list；

hql语句赋值：

方式1：String hql="from Dept where loc=？

query.setString(0, "BOSTON");

方式2（推荐）：

String hql="from Dept where loc=:loc

对应

query.setString("loc", "BOSTON");

方式3：将值封装到对象中：

String hql="from Dept where loc=:loc

对象.set(数值);

query. setProperties(对象)；

分页查询：

query.setMaxResults(rows);

query.setFirstResult((currPage-1)\*rows);

投影查询：

1.查一列：

String hql="select dname from Dept";

Query q=session.createQuery(hql);

List<String> dnames=q.list();

2.查多列：（类要提供对应的构造方法）

String hql="select new Dept(dname,loc) from Dept";

Query q=session.createQuery(hql);

List<Dept> list=q.list();

关联关系：

1.一对多双向关联

分清谁是多，谁是一

一的一方配置：

一的一方可以设置反转为true,提高操作效率

一的一方设置级联方式：save-update:同步保存更改下面的员工，delete：删除部门的同时删除该部门下所有员工，all:包括以上两种方式

<!-- 多的一方对应数据库哪张表 -->

<set name=*"emps"* table=*"EMP"* cascade=*"save-update"* inverse=*"true"*>

<!-- 这张表的外键是哪个列 -->

<key column=*"DEPTNO"*></key>

<!-- 这张表对应的实体类 -->

<one-to-many class=*"Emp"* />

</set>

多的一方配置：

<many-to-one name=*"dept"* class=*"Dept"* column=*"DEPTNO"*></many-to-one>

延迟加载：当用到数据的时候才会写语句查询数据值，默认lazy=”true”

反转：inverse=”true”，一般加在一的一方，将维护关系的权利交给多方

关联跟级联：

关联：能根据自己查对方的数据（单向），或根据对方查自己的数据（双向）

级联：除了包括关联的特点外，还有：对自己的增删改操作会影响到对方在数据库里的信息，一般在一的那一方加级联（cascade）

save-update:保存或修改自己会同步保存或修改对方

delete：删除自己的同时会删除对方（慎用）

all:包括以上两者

多对多：

建中间表：

中间表放两张表的主键，这两列联合做中间表主键

并且作为外键分别联系另外两张表

实体类：分别放对方的set集合

**project的hbm:**

**<set name=*"emps"* table=*"PROEMP"* cascade=*"save-update"*>**

**<key column=*"proid"*></key>**

**<many-to-many class=*"T\_Emp"* column=*"empno"*></many-to-many>**

**</set>**

**emp的hbm:**

**<set name=*"pros"* table=*"PROEMP"* inverse=*"true"*>**

**<key column=*"empno"*></key>**

<many-to-many class=*"T\_Project"* column=*"proid"*></many-to-many>

</set>

一对一：a.外键关联，b.主键关联

a.外键关联：（退化的多对一）

档案.hbm.xml

<many-to-one name=*"user1"* column=*"USERID"*

cascade=*"all"*

unique=*"true"*

></many-to-one>

用户.hbm.xml

<one-to-one name="dangan" class="DangAn" ></one-to-one>

b.主键关联：一方的主键作为外键参考另一方的主键

主动方：

<one-to-one name=*"user1"* class=*"User1"*

cascade=*"all"*></one-to-one>

被动方：

主键生成策略：foreign，param属性：另一方对象

<one-to-one name=*"dangan"* class=*"DangAn"* constrained=*”true”*></one-to-one>

组件查询：将一个表拆成2个实体类

组件属性要配component

<component name=*"home"* class=*"Home"*>

<parent name=*"user"*/>(home对象中的支配者)

<property name=*"city"* type=*"java.lang.String"* column=*"CITY"*></property>

<property name=*"address"* type=*"java.lang.String"* column=*"ADDRESS"*></property>

</component>

hql语句举例：

"select job,avg(sal) from Emp group by job";每个工作的平均工资

"select e.dept.dname,avg(e.sal) from Emp e group by e.dept.dname";//每个部门的平均工资

"select distinct e.dept.dname from Emp e where e.sal<3000";工资小于3000的员工所在部门名

"select d.dname,d.emps.size from Dept d where d.emps.size>4";部门人数大于4的部门名和部门人数

大数据存储：

数据库表中建CLOB、BLOB类型的列

实体类中建String、byte[]属性

hbm中匹配对应关系

<property name=*"bigclob"* type=*"java.lang.String"* column=*"BIGCLOB"*></property>

<property name=*"bigblob"* type=*"byte[]"* column=*"BIGBLOB"*></property>

Criteria查询：

Criteria criteria=session.createCriteria(Emp.class);//查谁放谁的.class

1.查符合某一条件的，add可以继续拼接

List<Dept> list=criteria

.add(Restrictions.eq("dname", "aaa"))

查集合属性是否为空用isEmpty(“”),查单个属性是否为空用isNull(“”)

2.按顺序查询

List<Dept> list=criteria

.addOrder(Order.asc("dname"))

3.分页查询

List<Dept> list=criteria

.setMaxResults(3)

.setFirstResult(0)

.list();

4.联合查询

List<Emp> list=criteria

.createAlias("dept", "d")

.add(Restrictions.eq("d.dname", "ACCOUNTING"))

.list();

5.投影查询

查一列：

List<String> list=criteria

.setProjection(Property.forName("dname"))

.list();

查多列：

List<Object[]> list=criteria

.setProjection( Projections.projectionList()

.add(Property.forName("dname")) .add(Property.forName("deptno"))

)

.list()

6.动态查询

DetachedCriteria创建对象不需要session

建对象

DetachedCriteria

detachedCriteria=DetachedCriteria.forClass(Dept.class);

给对象加限制条件

detachedCriteria.add(Restrictions.lt("deptno", 50));

执行查询

List<Dept> list=detachedCriteria.getExecutableCriteria(session).list();

使用Hibernate注解

1.导包：ejb3-persistence.jar；hibernate-annotations.jar；hibernate-commons-annotations.jar；

2.建实体类

@Entity

@Table(name="DEPT")//跟哪个表匹配

**public** **class** Dept1{……}

属性：

@Id //指明主键及主键生成策略

@GeneratedValue(strategy=GenerationType.SEQUENCE,generator="seq\_dept")

@SequenceGenerator(name="seq\_dept",sequenceName="seq\_dept\_deptno",initialValue=1)

**private** Integer deptno;

@Column //跟列名相同不用加name=””

**private** String dname;

@OneToMany(mappedBy="dept",cascade=CascadeType.*ALL*) //一对多

**private** Set<Emp1> emps=**new** HashSet<Emp1>();

//emp中多对一

@ManyToOne

@JoinColumn(name="deptno") //外键列

**private** Dept1 dept;

3.在总配置文件中添加实体类路径

<mapping class="entity.Dept1"/>

使用AnnotationConfiguration创建SessionFactory，再创建session