hibernate-day04

查询总结

1. 1.oid查询-get
2. 2.对象属性导航查询
3. 3.HQL
4. 4.Criteria
5. 5.原生SQL

查询-HQL语法

基础语法

```
String hql = " from  cn.itcast.domain.Customer ";//完整写法
String hql2 = " from  Customer ";//简单写法
String hql3 = " from java.lang.Object ";

Query query = session.createQuery(hql3);

List list = query.list();
```

排序

```
String hql1 = " from  cn.itcast.domain.Customer order by cust_id asc ";//完整写法
String hql2 = " from  cn.itcast.domain.Customer order by cust_id desc ";//完整写法

Query query = session.createQuery(hql2);

List list = query.list();
```

条件

```
String hql1 = " from  cn.itcast.domain.Customer where cust_id =? ";//完整写法
String hql2 = " from  cn.itcast.domain.Customer where cust_id = :id ";//完整写法

Query query = session.createQuery(hql1);

query.setParameter(0, 2);//
query.setParameter("id",21);

List list = query.list();
```

分页

```
String hql1 = " from  cn.itcast.domain.Customer  ";//完整写法

Query query = session.createQuery(hql1);

//limit 2,2
// (页数页数-1)*每页多少
query.setFirstResult(2);
query.setMaxResults(2);

List list = query.list();
```

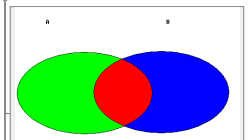
聚合

```
String hql1 = " select count(*) from  cn.itcast.domain.Customer  ";//完整写法
String hql2 = " select sum(cust_id) from  cn.itcast.domain.Customer  ";//完整写法
String hql3 = " select avg(cust_id) from  cn.itcast.domain.Customer  ";//完整写法
String hql4 = " select max(cust_id) from  cn.itcast.domain.Customer  ";//完整写法
String hql5 = " select min(cust_id) from  cn.itcast.domain.Customer  ";//完整写法

Query query = session.createQuery(hql5);

Number number = (Number) query.uniqueResult();
```

投影

```
String hql1 = " select cust_name from  cn.itcast.domain.Customer ";
String hql2 = " select cust_name,cust_id from  cn.itcast.domain.Customer  ";
String hql3 = " select new Customer(cust_id,cust_name) from  cn.itcast.domain.Customer  ";

Query query = session.createQuery(hql3);

List list = query.list();
```

多表查询



内连接： 红色部分

左外连接：绿色+红色

右外连接：红色+蓝色

SQL

```
//隐藏-得出SQL
// 出现连接-最多3对(最长)
//     select * from A,B
//     // 内连接
//       |-隐式内连接
//           select * from A,B  where b.aid = a.id
//       |-显式内连接
//           select * from A inner join B on b.aid = a.id
//   // 外连接
//       |- 左外
//           select * from A left [outer] join B on b.aid = a.id
//       |- 右外
//           select * from A right [outer] join B on b.aid = a.id
```

HQL

```
//HQL的多表查询
        //内连接(迫切)
        //外连接
//            |-左外(迫切)
//            |-右外(迫切)
```

HQL语法

内连接

```
String hql = " from Customer c inner join c.linkMens ";

Query query = session.createQuery(hql);

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

for(Object[] arr : list){
    System.out.println(Arrays.toString(arr));
}
```

```
String hql = " from Customer c inner join fetch c.linkMens ";

Query query = session.createQuery(hql);

List<Customer> list = query.list();

System.out.println(list);
```

左外链接

```
String hql = " from Customer c left join c.linkMens ";

Query query = session.createQuery(hql);

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

for(Object[] arr : list){
    System.out.println(Arrays.toString(arr));
}
```

右外连接

```
String hql = " from Customer c right join c.linkMens ";

Query query = session.createQuery(hql);

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

for(Object[] arr : list){
    System.out.println(Arrays.toString(arr));
}
```

查询-Criteria语法

语法

基本

```
Criteria c = session.createCriteria(Customer.class);

List<Customer> list = c.list();

System.out.println(list);
```

条件

```
Criteria c = session.createCriteria(Customer.class);

c.add(Restrictions.idEq(21));
c.add(Restrictions.eq("cust_id",21));

List<Customer> list = c.list();

System.out.println(list);
```

分页

```
Criteria c = session.createCriteria(Customer.class);
//limit 2,2
c.setFirstResult(0);
c.setMaxResults(2);

List<Customer> list = c.list();
```

排序

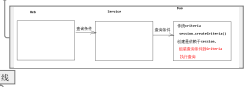
```
Criteria c = session.createCriteria(Customer.class);

c.addOrder(Order.asc("cust_id"));
//c.addOrder(Order.desc("cust_id"));

List<Customer> list = c.list();
```

统计

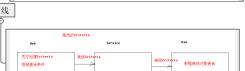
```
Criteria c = session.createCriteria(Customer.class);

//设置查询目标
c.setProjection(Projections.rowCount());

List list = c.list();
```

离线查询

非离线



离线



演示

```
@Test:
public void fun1(){
    //Service/web层
    DetachedCriteria dc  = DetachedCriteria.forClass(Customer.class);

    dc.add(Restrictions.idEq(61));//序列化中(全部与最第Criteria一致)

    //-----------------------------------------------
    Session session = HibernateUtils.openSession();
    Transaction tx = session.beginTransaction();
    //-----------------------------------------------
    Criteria c = dc.getExecutableCriteria(session);

    List list = c.list();

    System.out.println(list);
    //-----------------------------------------------
    tx.commit();
    session.close();

}
```

查询优化

类级别查询

- get方法:没有任何策略,调用即立即查询数据库加载数据.
- load方法:应用类级别的加载策略

```
<class name="Customer" table="cst_customer" lazy="false" >
```

- lazy(默认值)true,查询类时,会返回代理对象,会在使用属性时,根据关联的session查询数据库,加载数据.
- lazy:false. load方法会与get方法没有任何区别,调用时即加载数据.
- 结论:为了提高效率,建议使用延迟加载(懒加载)
- 注意:使用懒加载时要确保,调用属性加载数据时,session还是打开的,不然会抛出异常
  - org.hibernate.LazyInitializationException: could not initialize proxy - no Session

关联级别查询

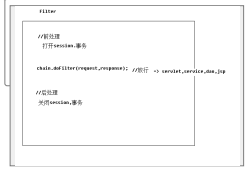
集合策略

```
lazy属性: 决定是否延迟加载
    true(默认值): 延迟加载,懒加载
    false: 立即加载
    extra: 极其懒惰
fetch属性: 决定加载策略,使用什么类型的sql语句加载集合数据
    select(默认值): 单表查询加载
    join: 使用多表查询加载集合
    subselect:使用子查询加载集合
```

关联属性策略

```
<!--
fetch 决定加载的sql语句
    select: 使用单表查询
    join : 多表查询
lazy  决定加载时机
    false: 立即加载
    proxy: 由customer的类级别加载策略决定.
  -->
```

- 结论:为了提高效率,fetch的选择上应选择select. lazy的取值应选择 true. 全部使用默认值.
- no-session问题解决:扩大session的作用范围.



批量抓取

```
<!-- batch-size: 抓取集合的数量为3.
    抓取客户的集合时,一次抓取几个客户的联系人集合.
  -->
```

练习:为客户列表增加查询条件