1. Hibernate查询方式

1 对象导航查询

（1）根据id查询某个客户，再查询这个客户里面所有的联系人

2 OID查询

（1）根据id查询某一条记录，返回对象

3 HQL查询

（1）Query对象，写hql语句实现查询

4 QBC查询

（1）Criteria对象

5 本地sql查询

（1）SQLQuery对象，使用普通sql实现查询

2. 对象导航查询

1 查询某个客户里面所有联系人过程，使用对象导航实现

2 代码

```
//根据cid=1客户，再查询这个客户里面所有联系人
Customer customer = session.get(Customer.class, 1);
//再查询这个客户里面所有联系人
//直接得到客户里面联系人的set集合
Set<LinkMan> linkman = customer.getSetLinkMan();

System.out.println(linkman.size());
```

3. OID查询

1 根据id查询记录

（1）调用session里面的get方法实现

```
//根据cid=1客户，再查询这个客户里面所有联系人
Customer customer = session.get(Customer.class, 1);
```

4. HQL查询

1 hql：hibernate query language，hibernate提供一种查询语言，hql语言和普通sql很相似，区别：普通sql操作数据库表和字段，hql操作实体类和属性

2 常用的hql语句

（1）查询所有： from 实体类名称

（2）条件查询： from 实体类名称 where 属性名称=?

（3）排序查询： from 实体类名称 order by 实体类属性名称 asc/desc

3 使用hql查询操作时候，使用Query对象

（1）创建Query对象，写hql语句

（2）调用query对象里面的方法得到结果

4-1. 查询所有

1 查询所有客户记录

（1）创建Query对象，写hql语句

（2）调用query对象里面的方法得到结果

2 查询所有： from 实体类名称

```
//1 创建query对象
Query query = session.createQuery("from Customer");
//2 调用方法得到结果
List<Customer> list = query.list();
```

4-2. 条件查询

1 hql条件查询语句写法：

（1）from 实体类名称 where 实体类属性名称=? and实体类属性名称=?

from 实体类名称 where 实体类属性名称 like ?

2 代码

```
//1 创建query对象
//SELECT * FROM t_customer WHERE cid=? AND custName=?
Query query = session.createQuery("from Customer c where c.cid=? and c.custName=?");

//2 设置条件值
// 问号里面设置值
// setParameter方法两个参数
// 第一个参数：int类型是问号位置，问号位置从0开始
// 第二个参数：具体参数值
//设置第一个问号值
query.setParameter(0, 1);
//设置第二个问号值
query.setParameter(1, "百度");

//3 调用方法得到结果
List<Customer> list = query.list();
```

```
setParameter(int arg0, Object arg1) : Query - Query
```

模糊查询

```
//1 创建query对象
Query query = session.createQuery("from Customer c where c.custName like ?");

//2 设置？的值
// %浪%
query.setParameter(0, "%浪%");
```

4-3. 排序查询

1 hql排序语句写法

（1）from 实体类名称 order by 实体类属性名称 asc/desc

```
//1 创建query对象
Query query = session.createQuery("from Customer order by cid desc");

//2 调用方法得到结果
List<Customer> list = query.list();
```

4-4. 分页查询

1 mysql实现分页

（1）使用关键字 limit实现

```
SELECT * FROM t_customer LIMIT 0,3
```

2 在hql中实现分页

（1）在hql操作中，在语句里面不能写limit，hibernate的Query对象封装两个方法实现分页操作

```
//1 创建query对象
//写查询所有的语句
Query query = session.createQuery("from Customer");

//2 设置分页数据
//2.1 设置开始位置
query.setFirstResult(0);
//2.2 设置每页记录数
query.setMaxResults(3);

//3 调用方法得到结果
List<Customer> list = query.list();
```

4-5. 投影查询

1 投影查询：查询不是所有字段值，而是部分字段的值

2 投影查询hql语句写法：

（1）select 实体类属性名称1, 实体类属性名称2 from 实体类名称

（2）select 后面不能写 * ，不支持的

3 具体实现

```
//1 创建query对象
Query query = session.createQuery("select custName from Customer");

//2 调用方法得到结果
List<Object> list = query.list();

for (Object object : list) {
    System.out.println(object);
}
```

4-6. 聚集函数使用

1 常用的聚集函数

（1）count、sum、avg、max、min

2 hql聚集函数语句写法

（1）查询表记录数

- select count(*) from 实体类名称

```
//1 创建query对象
Query query = session.createQuery("select count(*) from Customer");

//2 调用方法得到结果
//query对象里面有方法，直接返回对象形式
Object obj = query.uniqueResult();

//返回int类型
int count = (int) obj;

//首先把object变成long类型，再变成int类型
Long lobj = (Long) obj;
int count = lobj.intValue();
```

```
g.ClassCastException: java.lang.Long cannot be cast to java.lang.Integer
```

5. QBC查询

1 使用hql查询需要写hql语句实现，但是使用qbc时候，不需要写语句了，使用方法实现

2 使用qbc时候，操作实体类和属性

3 使用qbc，使用Criteria对象实现

5-1. 查询所有

1 创建Criteria对象

2 调用方法得到结果

```
//1 创建对象
Criteria criteria = session.createCriteria(Customer.class);
//2 调用方法得到结果
List<Customer> list = criteria.list();
```

5-2.条件查询

1 没有语句，使用封装的方法实现

```
//1 创建对象
Criteria criteria = session.createCriteria(Customer.class);

//2 使用Criteria对象里面的方法设置条件值
//  首先使用add方法，表示设置条件值
//  在add方法里面使用类的方法实现条件设置
//  类似于 cid=?
criteria.add(Restrictions.eq("cid", 1));
criteria.add(Restrictions.eq("custName", "百度"));

//3 调用方法得到结果
List<Customer> list = criteria.list();
```

```
criteria.add(Restrictions.like("custName", "%百%"));
```

5-3. 排序查询

```
asc(String propertyName) : Order - Order
desc(String propertyName) : Order - Order
```

```
//1 创建对象
Criteria criteria = session.createCriteria(Customer.class);

//2 设置对哪个属性进行排序，设置排序规则
criteria.addOrder(Order.desc("cid"));
```

5-4. 分页查询

```
//1 创建对象
Criteria criteria = session.createCriteria(Customer.class);

//2 设置分页数据
//2.1 设置开始位置
criteria.setFirstResult(0);
//2.2 每页显示记录数
criteria.setMaxResults(3);
```

开始位置计算公式： （当前页-1）*每页记录数

5-5. 统计查询

```
//1 创建对象
Criteria criteria = session.createCriteria(Customer.class);

//2 设置操作
criteria.setProjection(Projections.rowCount());

//3 调用方法得到结果
Object obj = criteria.uniqueResult();

Long lobj = (Long) obj;
int count = lobj.intValue();
```

5-6. 离线查询

1 servlet调用service，service调用dao

（1）在dao里面对数据库crud操作

（2）在dao里面使用hibernate框架，使用hibernate框架时候，调用session里面的方法实现功能

```
//1 创建对象
Criteria criteria = session.createCriteria(Customer.class);
DetachedCriteria detachedCriteria = DetachedCriteria.forClass(Customer.class);

//2 最终执行时候才需要到session
Criteria criteria = detachedCriteria.getExecutableCriteria(session);

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

（3）在后面ssh练习中具体应用

6. HQL多表查询

6-1. 回顾Mysql里面多表查询

1 内连接

```sql
/*内连接查询*/
SELECT * FROM t_customer c,t_linkman l WHERE c.cid=l.clid

SELECT * FROM t_customer c INNER JOIN t_linkman l ON c.cid=l.clid
```

2 左外连接

```sql
/*左外连接*/
SELECT * FROM t_customer c LEFT OUTER JOIN t_linkman l ON c.cid=l.clid
```

3 右外连接

```sql
/*右外连接*/
SELECT * FROM t_customer c RIGHT OUTER JOIN t_linkman l ON c.cid=l.clid
```

6-2. HQL实现多表查询

Hql多表查询

（1）内连接

（2）左外连接

（3）右外连接

（4）迫切内连接

（5）迫切左外连接

6-2-1. HQL内连接

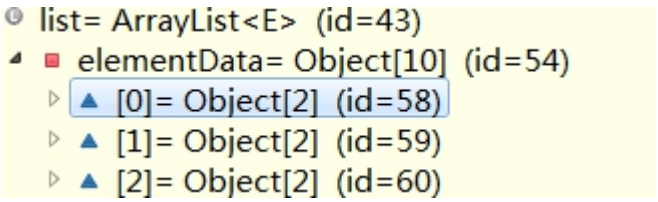
1 内连接查询hql语句写法：以客户和联系人为例

（1）from Customer c inner join c.setLinkMan

```java
//1 创建query对象
Query query = session.createQuery("from Customer c inner join c.setLinkMan");

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

返回list，list里面每部分是数组形式



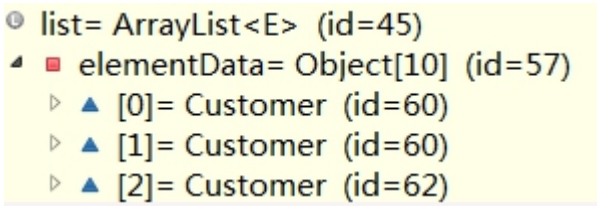
2 演示迫切内连接

（1）迫切内连接和内连接底层实现一样的

（2）区别：使用内连接返回list中每部分是数组，迫切内连接返回list每部分是对象

（3）hql语句写法

- from Customer c inner join fetch c.setLinkMan



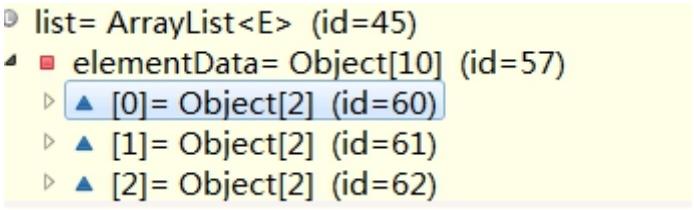
6-2-2. HQL左外连接

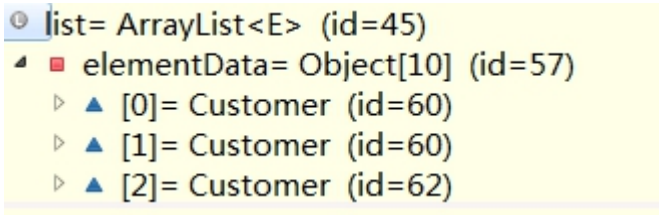
1 左外连接hql语句：

（1）from Customer c left outer join c.setLinkMan

（2）迫切左外连接from Customer c left outer join fetch c.setLinkMan

2 左外连接返回list中每部分是数组，迫切左外连接返回list每部分是对象





1 右外连接hql语句：

（1）from Customer c right outer join c.setLinkMan

7. Hibernate检索策略

1. 检索策略的概念

1 hibernate检索策略分为两类：

（1）立即查询：根据id查询，调用get方法，一调用get方法马上发送语句查询数据库

```
//根据cid=1客户
//执行get方法之后，是否发送sql语句
//调用get方法马上发送sql语句查询数据库
Customer customer = session.get(Customer.class, 1);
```

（2）延迟查询：根据id查询，还有load方法，调用load方法不会马上发送语句查询数据，只有得到对象里面的值时候才会发送语句查询数据库

```
/*
 * 1 调用load方法之后，不会马上发送sql语句
 * （1）返回对象里面只有 id值
 *
 * 2 得到对象里面不是id的其他值时候才会发送语句
 * */
Customer customer = session.load(Customer.class, 2);

System.out.println(customer.getCid());

System.out.println(customer.getCustName());
```

2 延迟查询分成两类：

（1）类级别延迟：根据id查询返回实体类对象，调用load方法不会马上发送语句

（2）关联级别延迟：

- 查询某个客户，再查询这个客户的所有联系人，查询客户的所有联系人的过程是否需要延迟，这个过程称为关联级别延迟

```
//根据cid=1客户，再查询这个客户里面所有联系人
Customer customer = session.get(Customer.class, 1);
//再查询这个客户里面所有联系人
//直接得到客户里面联系人的set集合

//得到set集合，没有发送语句
Set<LinkMan> linkman = customer.getSetLinkMan();

//  发送语句
System.out.println(linkman.size());
```

2. 关联级别延迟操作

1 在映射文件中进行配置实现

（1）根据客户得到所有的联系人，在客户映射文件中配置

2 在set标签上使用属性

（1）fetch：值select（默认）

（2）lazy：值

- true：延迟（默认）
- false：不延迟
- extra：极其延迟

```
fetch="select" lazy="true">
```

```
fetch="select" lazy="false">
```

（1）调用get之后，发送两条sql语句

```
        customer0_.custName as custName2_0_0_,
        customer0_.custLevel as custLeve3_0_0_,
        customer0_.custSource as custSour4_0_0_,
        customer0_.custPhone as custPhon5_0_0_,
        customer0_.custMobile as custMobi6_0_0_
    from
        t_customer customer0_
    where
        customer0_.cid=?
Hibernate:
    select
        setlinkman0_.clid as clid5_1_0_,
        setlinkman0_.lkm_id as lkm_id1_1_0_,
        setlinkman0_.lkm_id as lkm_id1_1_1_,
        setlinkman0_.lkm_name as lkm_name2_1_1_,
        setlinkman0_.lkm_gender as lkm_gend3_1_1_,
        setlinkman0_.lkm_phone as lkm_phon4_1_1_,
        setlinkman0_.clid as clid5_1_1_
    from
        t_linkman setlinkman0_
    where
        setlinkman0 .clid=?
```

```
fetch="select" lazy="extra">
```

（1）极其懒惰，要什么值给什么值

```
Hibernate:
    select
        count(lkm_id)
    from
        t_linkman
    where
        clid =?
2
```

8. 批量抓取

1 查询所有的客户，返回list集合，遍历list集合，得到每个客户，得到每个客户的所有联系人

（1）上面操作代码，发送多条sql语句

```
//查询所有客户
Criteria criteria = session.createCriteria(Customer.class);
List<Customer> list = criteria.list();
//得到每个客户里面所有的联系人
for (Customer customer : list) {
    System.out.println(customer.getCid()+"::"+customer.getCustName());
    //每个客户里面所有的联系人
    Set<LinkMan> setLinkMan = customer.getSetLinkMan();
    for (LinkMan linkMan : setLinkMan) {
        System.out.println(linkMan.getLkm_id()+"::"+linkMan.getLkm_name());
    }
}
```

2 在客户的映射文件中，set标签配置

（1）batch-size值，值越大发送语句越少

```
<set name="setLinkMan" batch-size="10">
```