经常用到Criteria,所以做一下总结

 net.sf.hibernate.Criteria这个接口代表对一个特定的持久化类的查询。Session是用来制造Criteria实例的工厂。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Criteria对SQL进行封装，让开发人员可以用物件的方式来对资料库进行操作，例如下面的查询User表格中的所有资料：  **Criteria criteria = session.createCriteria(User.class); // 查询user所有栏位 List users = criteria.list(); Iterator iterator =  users.iterator(); System.out.println("id /t name/age"); while(iterator.hasNext()) {     User user = (User) iterator.next();      System.out.println(user.getId() +                                " /t " + user.getName() +                                "/" + user.getAge());            }**  Hibernate实际上使用以下的SQL来查询资料库：  **select this\_.id as id0\_, this\_.name as name0\_0\_, this\_.age as age0\_0\_ from user this\_**  Criteria实际上只是个容器，如果想要设定查询条件，则要使用add()方法加入Restrictions的条件限制，例如查询age大于20且小于40的资料：  **Criteria criteria = session.createCriteria(User.class); criteria.add(Restrictions.gt("age", new Integer(20))); criteria.add(Restrictions.lt("age", new Integer(40))); List users = criteria.list();**  您也可以使用逻辑组合来进行查询，例如结合age等于（eq）20或（or）age为空（isNull）的条件：  **Criteria criteria = session.createCriteria(User.class);** **criteria.add(Restrictions.or(                     Restrictions.eq("age", new Integer(20)),                     Restrictions.isNull("age")                 ));  List users = criteria.list();**  也可以使用sqlRestriction()方法来提供SQL语法作限定查询，例如查询name以cater开头的资料：  **Criteria criteria = session.createCriteria(User.class); criteria.add(Restrictions.sqlRestriction("{alias}.name LIKE (?)", "cater%", Hibernate.STRING)); List users = criteria.list();**  其中alias将被替换为与User类别相关的名称，而?将被替换为cater%，也就是第二个参数所提供的值，在SQL撰写时，不必再写WHERE，如果有多个查询条件，例如BETWEEN子句的查询，则可以如下：  **Criteria criteria = session.createCriteria(User.class); Integer[] ages = {new Integer(20), new Integer(40)}; Type[] types = {Hibernate.INTEGER, Hibernate.INTEGER}; criteria.add(Restrictions.sqlRestriction("{alias}.age BETWEEN (?) AND (?)", ages, types)); List users = criteria.list();**  Restrictions的几个常用限定查询方法如下表所示：   |  |  | | --- | --- | | 方法 | 说明 | | Restrictions.eq | 等于 | | Restrictions.allEq | 使用Map，使用key/value进行多个等于的比对 | | Restrictions.gt | 大于 > | | Restrictions.ge | 大于等于 >= | | Restrictions.lt | 小于 < | | Restrictions.le | 小于等于 <= | | Restrictions.between | 对应SQL的BETWEEN子句 | | Restrictions.like | 对应SQL的LIKE子句 | | Restrictions.in | 对应SQL的in子句 | | Restrictions.and | and关系 | | Restrictions.or | or关系 | | Restrictions.sqlRestriction | SQL限定查询 |   **Criteria** **高级特性**  **限定返回的记录范围**   通过criteria. setFirstResult/setMaxResults 方法可以限制一次查询返回的记录范围:   Criteria criteria = session.createCriteria(User.**class**);   //限定查询返回检索结果中，从第一百条结果开始的20条记录   criteria.setFirstResult(100);   criteria.setMaxResults(20);  **对查询结果进行排序**   //查询所有groupId=2的记录   //并分别按照姓名(顺序)和groupId（逆序）排序   Criteria criteria = session.createCriteria(TUser.**class**);   criteria.add(Expression.eq("groupId",**new** Integer(2)));   criteria.addOrder(Order.asc("name"));   criteria.addOrder(Order.desc("groupId")); |

进行复合查询示例:

 LogicalExpression subcriteria = Expression.or(Expression.ilike("name", searchtxt, MatchMode.ANYWHERE),   
          Expression.ilike("description", searchtxt, MatchMode.ANYWHERE));  
      criteria.add(subcriteria );