最近在项目中使用 Spring 和 Hibernate 进行开发，有感于 Criteria 比较好用，在查询方法设计上可以灵活的根据 Criteria 的特点来方便地进行查询条件的组装。现在对 Hibernate的Criteria 的用法进行总结：

　　Hibernate 设计了 CriteriaSpecification 作为 Criteria 的父接口，下面提供了 Criteria和DetachedCriteria 。

　 　Criteria 和 DetachedCriteria 的主要区别在于创建的形式不一样， Criteria 是在线的，所以它是由 Hibernate Session 进行创建的；而 DetachedCriteria 是离线的，创建时无需Session，DetachedCriteria 提供了 2 个静态方法 forClass(Class) 或 forEntityName(Name) 进行DetachedCriteria 实例的创建。 Spring 的框架提供了getHibernateTemplate().findByCriteria(detachedCriteria) 方法可以很方便地根据DetachedCriteria 来返回查询结果。

　　Criteria 和 DetachedCriteria 均可使用 Criterion 和 Projection 设置查询条件。可以设置 FetchMode( 联合查询抓取的模式 ) ，设置排序方式。对于 Criteria 还可以设置 FlushModel（冲刷 Session 的方式）和 LockMode （数据库锁模式）。

　　下面对 Criterion 和 Projection 进行详细说明。

　　Criterion 是 Criteria 的查询条件。Criteria 提供了 add(Criterion criterion) 方法来添加查询条件。

　 　Criterion 接口的主要实现包括： Example 、 Junction 和 SimpleExpression 。而 Junction 的实际使用是它的两个子类 conjunction 和 disjunction ，分别是使用 AND 和 OR 操作符进行来联结查询条件集合。

　 　Criterion 的实例可以通过 Restrictions 工具类来创建，Restrictions 提供了大量的静态方法，如 eq （等于）、 ge （大于等于）、 between 等来方法的创建 Criterion 查询条件 （SimpleExpression 实例）。除此之外， Restrictions 还提供了方法来创建 conjunction 和 disjunction 实例，通过往该实例的 add(Criteria) 方法来增加查询条件形成一个查询条件集合。

　　至于 Example 的创建有所不同， Example 本身提供了一个静态方法 create(Object entity) ，即根据一个对象（实际使用中一般是映射到数据库的对象）来创建。然后可以设置一些过滤条件：

　　Example exampleUser =**Example.create**(u)

**.ignoreCase**() // 忽略大小写

**.enableLike**(**MatchMode.**ANYWHERE);

　 　// 对 String 类型的属性，无论在那里值在那里都匹配。相当于 %value%

Project 主要是让 Criteria 能够进行报表查询，并可以实现分组。 Project 主要有 SimpleProjection 、ProjectionList 和 Property 三个实现。其中SimpleProjection 和 ProjectionList 的实例化是由内建的 Projections 来完成，如提供的 avg 、 count 、 max 、 min 、 sum 可以让开发者很容易对某个字段进行统计查询。

　　 Property 是对某个字段进行查询条件的设置，如通过Porperty.forName(“color”).in(new String[]{“black”,”red”,”write”}); 则可以创建一个 Project 实例。通过 criteria 的 add(Project) 方法加入到查询条件中去。

　　使用 Criteria 进行查询，主要要清晰的是 Hibernate 提供了那些类和方法来满足开发中查询条件的创建和组装，下面介绍几种用法：

　　1. 创建一个Criteria 实例

　　org.hibernate.Criteria接口表示特定持久类的一个查询。Session是 Criteria实例的工厂。

Criteria crit = sess.createCriteria(Cat.class);  
crit.setMaxResults(50);  
List cats = crit.list();

　　2. 限制结果集内容

　　一个单独的查询条件是org.hibernate.criterion.Criterion 接口的一个实例。

　　org.hibernate.criterion.Restrictions类 定义了获得某些内置Criterion类型的工厂方法。List cats = sess.createCriteria(Cat.class)  
　　.add( Restrictions.like("name", "Fritz%") )  
　　.add( Restrictions.between("weight", minWeight, maxWeight) )  
　　.list();

　　约束可以按逻辑分组。

List cats = sess.createCriteria(Cat.class)  
　　.add( Restrictions.like("name", "Fritz%") )  
　　.add( Restrictions.or(  
　　　　Restrictions.eq( "age", new Integer(0) ),  
　　　　Restrictions.isNull("age")  
　　) )  
　　.list();  
List cats = sess.createCriteria(Cat.class)  
　　.add( Restrictions.in( "name", new String[] { "Fritz", "Izi", "Pk" } ) )  
　　.add( Restrictions.disjunction()  
　　　　.add( Restrictions.isNull("age") )  
　　　　.add( Restrictions.eq("age", new Integer(0) ) )  
　　　　.add( Restrictions.eq("age", new Integer(1) ) )  
　　　　.add( Restrictions.eq("age", new Integer(2) ) )  
　　) )  
　　.list();

　　Hibernate提供了相当多的内置criterion类型(Restrictions 子类), 但是尤其有用的是可以允许你直接使用SQL。

List cats = sess.createCriteria(Cat.class)  
　　.add( Restrictions.sql("lower({alias}.name) like lower(?)", "Fritz%",  
Hibernate.STRING) )  
　　.list();

　　{alias}占位符应当被替换为被查询实体的列别名。

　　Property实例是获得一个条件的另外一种途径。你可以通过调用Property.forName() 创建一个Property。

Property age = Property.forName("age");  
List cats = sess.createCriteria(Cat.class)  
　　.add( Restrictions.disjunction()  
　　　　.add( age.isNull() )  
　　　　.add( age.eq( new Integer(0) ) )  
　　　　.add( age.eq( new Integer(1) ) )  
　　　　.add( age.eq( new Integer(2) ) )  
　　) )  
　　.add( Property.forName("name").in( new String[] { "Fritz", "Izi", "Pk" } ) )  
　　.list();

　　3. 结果集排序

　　你可以使用org.hibernate.criterion.Order来为查询结果排序。

List cats = sess.createCriteria(Cat.class)  
　　.add( Restrictions.like("name", "F%")  
　　.addOrder( Order.asc("name") )  
　　.addOrder( Order.desc("age") )  
　　.setMaxResults(50)  
　　.list();  
List cats = sess.createCriteria(Cat.class)  
　　.add( Property.forName("name").like("F%") )  
　　.addOrder( Property.forName("name").asc() )  
　　.addOrder( Property.forName("age").desc() )  
　　.setMaxResults(50)  
　　.list();

　　4. 关联

　　你可以使用createCriteria()非常容易的在互相关联的实体间建立 约束。

List cats = sess.createCriteria(Cat.class)  
　　.add( Restrictions.like("name", "F%")  
　　.createCriteria("kittens")  
　　　　.add( Restrictions.like("name", "F%")  
　　.list();

　　注意第二个 createCriteria()返回一个新的 Criteria实例，该实例引用kittens 集合中的元素。 接下来，替换形态在某些情况下也是很有用的。

List cats = sess.createCriteria(Cat.class)  
　　.createAlias("kittens", "kt")  
　　.createAlias("mate", "mt")  
　　.add( Restrictions.eqProperty("kt.name", "mt.name") )  
　　.list();

　　(createAlias()并不创建一个新的 Criteria实例。)

　　Cat实例所保存的之前两次查询所返回的kittens集合是 没有被条件预过滤的。如果你希望只获得符合条件的kittens， 你必须使用returnMaps()。

List cats = sess.createCriteria(Cat.class)  
　　.createCriteria("kittens", "kt")  
　　.add( Restrictions.eq("name", "F%") )  
　　.returnMaps()  
　　.list();  
Iterator iter = cats.iterator();  
while ( iter.hasNext() ) {  
　　Map map = (Map) iter.next();  
　　Cat cat = (Cat) map.get(Criteria.ROOT\_ALIAS);  
　　Cat kitten = (Cat) map.get("kt");  
}

　　5. 动态关联抓取

　　你可以使用setFetchMode()在运行时定义动态关联抓取的语义。

List cats = sess.createCriteria(Cat.class)  
　　.add( Restrictions.like("name", "Fritz%") )  
　　.setFetchMode("mate", FetchMode.EAGER)  
　　.setFetchMode("kittens", FetchMode.EAGER)  
　　.list();

　　这个查询可以通过外连接抓取mate和kittens。

　　6. 查询示例

　　org.hibernate.criterion.Example类允许你通过一个给定实例 构建一个条件查询。

Cat cat = new Cat();  
cat.setSex('F');  
cat.setColor(Color.BLACK);  
List results = session.createCriteria(Cat.class)  
　　.add( **Example.create**(cat) )  
　　.list();

　　版本属性、标识符和关联被忽略。默认情况下值为null的属性将被排除。

　　可以自行调整Example使之更实用。

Example example = **Example.create**(cat)  
　　.excludeZeroes()　　　　　 //exclude zero valued properties  
　　.excludeProperty("color") //exclude the property named "color"  
　　**.ignoreCase**()　　　　　　　//perform case insensitive string comparisons  
　　**.enableLike**();　　　　　　 //use like for string comparisons  
List results = session.createCriteria(Cat.class)  
　　.add(example)  
　　.list();

　　甚至可以使用examples在关联对象上放置条件。

List results = session.createCriteria(Cat.class)  
　　.add( **Example.create**(cat) )  
　　.createCriteria("mate")  
　　　　.add( **Example.create**( cat.getMate() ) )  
　　.list();

　　7. 投影(Projections)、聚合（aggregation）和分组（grouping）

　　org.hibernate.criterion.Projections是 Projection 的实例工厂。我们通过调用 setProjection()应用投影到一个查询。

List results = session.createCriteria(Cat.class)  
　　.setProjection( Projections.rowCount() )  
　　.add( Restrictions.eq("color", Color.BLACK) )  
　　.list();  
List results = session.createCriteria(Cat.class)  
　　.setProjection( Projections.projectionList()  
　　　　.add( Projections.rowCount() )  
　　　　.add( Projections.avg("weight") )  
　　　　.add( Projections.max("weight") )  
　　　　.add( Projections.groupProperty("color") )  
　　)  
　　.list();

　　在一个条件查询中没有必要显式的使用 "group by" 。某些投影类型就是被定义为 分组投影，他们也出现在SQL的group by子句中。

　　可以选择把一个别名指派给一个投影，这样可以使投影值被约束或排序所引用。下面是两种不同的实现方式：

List results = session.createCriteria(Cat.class)  
　　.setProjection( Projections.alias( Projections.groupProperty("color"), "colr" ) )  
　　.addOrder( Order.asc("colr") )  
　　.list();  
　　  
List results = session.createCriteria(Cat.class)  
　　.setProjection( Projections.groupProperty("color").as("colr") )  
　　.addOrder( Order.asc("colr") )  
　　.list();

　　alias()和as()方法简便的将一个投影实例包装到另外一个 别名的Projection实例中。简而言之，当你添加一个投影到一个投影列表中时 你可以为它指定一个别名：

List results = session.createCriteria(Cat.class)  
　　.setProjection( Projections.projectionList()  
　　　　.add( Projections.rowCount(), "catCountByColor" )  
　　　　.add( Projections.avg("weight"), "avgWeight" )  
　　　　.add( Projections.max("weight"), "maxWeight" )  
　　　　.add( Projections.groupProperty("color"), "color" )  
　　)  
　　.addOrder( Order.desc("catCountByColor") )  
　　.addOrder( Order.desc("avgWeight") )  
　　.list();  
List results = session.createCriteria(Domestic.class, "cat")  
　　.createAlias("kittens", "kit")  
　　.setProjection( Projections.projectionList()  
　　　　.add( Projections.property("cat.name"), "catName" )  
　　　　.add( Projections.property("kit.name"), "kitName" )  
　　)  
　　.addOrder( Order.asc("catName") )  
　　.addOrder( Order.asc("kitName") )  
　　.list();

　　也可以使用Property.forName()来表示投影：

List results = session.createCriteria(Cat.class)  
　　.setProjection( Property.forName("name") )  
　　.add( Property.forName("color").eq(Color.BLACK) )  
　　.list();  
List results = session.createCriteria(Cat.class)  
　　.setProjection( Projections.projectionList()  
　　　　.add( Projections.rowCount().as("catCountByColor") )  
　　　　.add( Property.forName("weight").avg().as("avgWeight") )  
　　　　.add( Property.forName("weight").max().as("maxWeight") )  
　　　　.add( Property.forName("color").group().as("color" )  
　　)  
　　.addOrder( Order.desc("catCountByColor") )  
　　.addOrder( Order.desc("avgWeight") )  
　　.list();

　　8. 离线(detached)查询和子查询

　　DetachedCriteria类使你在一个session范围之外创建一个查询，并且可以使用任意的 Session来执行它。

DetachedCriteria query = DetachedCriteria.forClass(Cat.class)  
　　.add( Property.forName("sex").eq('F') );  
//创建一个Session  
Session session = .;  
Transaction txn = session.beginTransaction();  
List results = query.getExecutableCriteria(session).setMaxResults(100).list();  
txn.commit();  
session.close();

　　DetachedCriteria也可以用以表示子查询。条件实例包含子查询可以通过 Subqueries或者Property获得。

DetachedCriteria avgWeight = DetachedCriteria.forClass(Cat.class)  
　　.setProjection( Property.forName("weight").avg() );  
session.createCriteria(Cat.class)  
　　.add( Property.forName("weight).gt(avgWeight) )  
　　.list();  
DetachedCriteria weights = DetachedCriteria.forClass(Cat.class)  
　　.setProjection( Property.forName("weight") );  
session.createCriteria(Cat.class)  
　　.add( Subqueries.geAll("weight", weights) )  
　　.list();

　　相互关联的子查询也是有可能的：

DetachedCriteria avgWeightForSex = DetachedCriteria.forClass(Cat.class, "cat2")  
　　.setProjection( Property.forName("weight").avg() )  
　　.add( Property.forName("cat2.sex").eqProperty("cat.sex") );  
session.createCriteria(Cat.class, "cat")  
　　.add( Property.forName("weight).gt(avgWeightForSex) )  
　　.list();