**Hibernate Criteria Queries**

The simplest example of a criteria query is one with no optional parameters or restrictions—the criteria query will simply return every object that corresponds to the class.

**Criteria crit = session.createCriteria(Product.class);**

**List<Product> results = crit.list();**

Using Restrictions with Criteria

**i) Restrictions.eq() Example**

To retrieve objects that have a property value that “**equals**” your restriction, use the eq() method on Restrictions, as follows:

**Criteria crit = session.createCriteria(Product.class);**

**crit.add(Restrictions.eq("description","Mouse"));**

**List<Product> results = crit.list()**

Above query will search all products having description as “Mouse”.

**i) Restrictions.ne() Example**

To retrieve objects that have a property value “not equal to” your restriction, use the ne() method on Restrictions, as follows:

**Criteria crit = session.createCriteria(Product.class);**

**crit.add(Restrictions.ne("description","Mouse"));**

**List<Product> results = crit.list()**

Above query will search all products having description anything but not “Mouse”.

NOTE :

**You cannot use the not-equal restriction to retrieve records with a NULL value in the database for that property (in SQL, and therefore in Hibernate, NULL represents the absence of data, and so cannot be compared with data)**. If you need to retrieve objects with NULL properties, you will have to use the **isNull()**restriction.

**iii) Restrictions.like() and Restrictions.ilike() Example**

Instead of searching for exact matches, we can retrieve all objects that have a property matching part of a given pattern. To do this, we need to create an SQL LIKE clause, with either the like() or the ilike()method. **The ilike() method is case-insensitive.**

**Criteria crit = session.createCriteria(Product.class);**

**crit.add(Restrictions.like("name","Mou%",MatchMode.ANYWHERE));**

**List<Product> results = crit.list();**

The MatchMode object (a type-safe enumeration) has four different matches:

**ANYWHERE: Anyplace in the string  
END: The end of the string  
EXACT: An exact match  
START: The beginning of the string**

**iv) Restrictions.isNull() and Restrictions.isNotNull() Example**

The isNull() and isNotNull() restrictions allow you to do a search for objects that have (or do not have) null property values.

**Criteria crit = session.createCriteria(Product.class);**

**crit.add(Restrictions.isNull("name"));**

**List<Product> results = crit.list();**

**v) Restrictions.gt(), Restrictions.ge(), Restrictions.lt() and Restrictions.le() Examples**

Several of the restrictions are useful for doing math comparisons. The greater-than comparison is gt(), the greater-than-or-equal-to comparison is ge(), the less-than comparison is lt(), and the less-than-or-equal-to comparison is le(). We can do a quick retrieval of all products with prices over $25 like this, relying on Java’s type promotions to handle the conversion to Double:

**Criteria crit = session.createCriteria(Product.class);**

**crit.add(Restrictions.gt("price", 25.0));**

**List<Product> results = crit.list();**

**vi) Combining Two or More Criteria Examples**

Moving on, we can start to do more complicated queries with the Criteria API. For example, we can combine AND and OR restrictions in logical expressions. When we add more than one constraint to a criteria query, it is interpreted as an AND, like so:

**Criteria crit = session.createCriteria(Product.class);**

**crit.add(Restrictions.lt("price",10.0));**

**crit.add(Restrictions.ilike("description","mouse", MatchMode.ANYWHERE));**

**List<Product> results = crit.list();**

If we want to have two restrictions that return objects that satisfy either or both of the restrictions, we need to use the or() method on the Restrictions class, as follows:

**Criteria crit = session.createCriteria(Product.class);**

**Criterion priceLessThan = Restrictions.lt("price", 10.0);**

**Criterion mouse = Restrictions.ilike("description", "mouse", MatchMode.ANYWHERE);**

**LogicalExpression orExp = Restrictions.or(priceLessThan, mouse);**

**crit.add(orExp);**

**List results=crit.list();**

The orExp logical expression that we have created here will be treated like any other criterion. We can therefore add another restriction to the criteria:

**Criteria crit = session.createCriteria(Product.class);**

**Criterion price = Restrictions.gt("price",new Double(25.0));**

**Criterion name = Restrictions.like("name","Mou%");**

**LogicalExpression orExp = Restrictions.or(price,name);**

**crit.add(orExp);**

**crit.add(Restrictions.ilike("description","blocks%"));**

**List results = crit.list();**

**vii) Using Disjunction Objects with Criteria**

If we wanted to create an OR expression with more than two different criteria (for example, “price > 25.0 OR name like Mou% OR description not like blocks%”), we would use an org.hibernate.criterion.Disjunction object to represent a disjunction.

You can obtain this object from the disjunction() factory method on the Restrictions class. The disjunction is more convenient than building a tree of OR expressions in code. To represent an AND expression with more than two criteria, you can use the conjunction() method, although you can easily just add those to the Criteria object. The conjunction can be more convenient than building a tree of AND expressions in code. Here is an example that uses the disjunction:

**Criteria crit = session.createCriteria(Product.class);**

**Criterion priceLessThan = Restrictions.lt("price", 10.0);**

**Criterion mouse = Restrictions.ilike("description", "mouse", MatchMode.ANYWHERE);**

**Criterion browser = Restrictions.ilike("description", "browser", MatchMode.ANYWHERE);**

**Disjunction disjunction = Restrictions.disjunction();**

**disjunction.add(priceLessThan);**

**disjunction.add(mouse);**

**disjunction.add(browser);**

**crit.add(disjunction);**

**List results = crit.list();**

**viii) Restrictions.sqlRestriction() Example**

sqlRestriction() restriction allows you to directly specify SQL in the Criteria API. It’s useful if you need to use SQL clauses that Hibernate does not support through the Criteria API.

Your application’s code does not need to know the name of the table your class uses. Use {alias} to signify the class’s table, as follows:

**Criteria crit = session.createCriteria(Product.class);**

**crit.add(Restrictions.sqlRestriction("{alias}.description like 'Mou%'"));**

**List<Product> results = crit.list();**