Hibernate Tutorial 10 Criteria Queries

By Gary Mak
hibernatetutorials@metaarchit.com
September 2006

1. Using criteria queries

In the last tutorial, we have introduced how to use HQL for querying persistent objects. Hibernate is providing an alternative method called “Criteria Queries” for this task. It is a set of API for us to build up a query in a more programmatic way comparing to HQL.

Sometimes we need to build up a query dynamically in our application, e.g. for an advanced search function. The traditional method of doing this is to generate a HQL statement, or SQL statement if not using Hibernate, by string concatenation. The problem for this method is making your code hard to maintain because of the hard reading statement fragments.

```
Criteria criteria = session.createCriteria(Book.class)
criteria.add(Restrictions.eq("name", "Hibernate Quickly"));
List books = criteria.list();
```

This criteria query corresponds to the following HQL query.

```
from Book book
where book.name = 'Hibernate Quickly'
```

Most methods in the Criteria class return the instance of itself, so that we can build up our criteria in the following way.

```
Criteria criteria = session.createCriteria(Book.class)
    .add(Restrictions.eq("name", "Hibernate Quickly"));
List books = criteria.list();
```

2. Restrictions

We can add a series of restrictions to our criteria query to filter the results, just like building up the where clause in HQL. The Restrictions class provides a variety of methods for restriction building. Each restriction added will be treated as a logical conjunction.

```
Criteria criteria = session.createCriteria(Book.class)
    .add(Restrictions.like("name", "%Hibernate%"))
    .add(Restrictions.between("price", new Integer(100), new Integer(200)));
List books = criteria.list();
```

So, this criteria query corresponds to the following HQL query.

```
from Book book
where (book.name like '%Hibernate%') and (book.price between 100 and 200)
```

We can also group some of the restrictions with logical disjunction.

```
Criteria criteria = session.createCriteria(Book.class)
    .add(Restrictions.or(
            Restrictions.like("name", "%Hibernate%"),
            Restrictions.like("name", "%Java%")
        )
    )
    .add(Restrictions.between("price", new Integer(100), new Integer(200)));
List books = criteria.list();
```

The corresponding HQL statement is as follow.

```
from Book book
where (book.name like '%Hibernate%' or book.name like '%Java%') and (book.price between 100
and 200)
```

3. Associations

In HQL, we can reference an association property by its name to trigger an implicit join. But for criteria queries, will the same thing happen?

```
Criteria criteria = session.createCriteria(Book.class)
    .add(Restrictions.like("name", "%Hibernate%"))
    .add(Restrictions.eq("publisher.name", "Manning"));
List books = criteria.list();
```

If you execute the above code, an exception will be thrown for Hibernate could not resolve the property “publisher.name”. That means implicit join is not supported for criteria queries. To join an association explicitly, you need to create a new criteria object for it.

```
Criteria criteria = session.createCriteria(Book.class)
    .add(Restrictions.like("name", "%Hibernate%"))
    .createCriteria("publisher")
        .add(Restrictions.eq("name", "Manning"));
List books = criteria.list();
```

```
from Book book
where book.name like '%Hibernate%' and book.publisher.name = 'Manning'
```

When using criteria query, you can specify the fetch mode of an association dynamically.

```
Criteria criteria = session.createCriteria(Book.class)
    .add(Restrictions.like("name", "%Hibernate%"))
    .setFetchMode("publisher", FetchMode.JOIN)
    .setFetchMode("chapters", FetchMode.SELECT);
List books = criteria.list();
```

4. Projections

If you want to customize the fields of the result, you should use Projections to specify the fields to be returned.

```
Criteria criteria = session.createCriteria(Book.class)
    .setProjection(Projections.property("name"));
List books = criteria.list();

select book.name
from Book book
```

You can use the aggregate functions as well. These functions are already encapsulated in the Projections class as static methods.

```
Criteria criteria = session.createCriteria(Book.class)
    .setProjection(Projections.avg("price"));
List books = criteria.list();

select avg(book.price)
from Book book
```

5. Ordering and grouping

The results can be sorted in ascending or descending order also.

```
Criteria criteria = session.createCriteria(Book.class)
    .addOrder(Order.asc("name"))
    .addOrder(Order.desc("publishDate"));
List books = criteria.list();

from Book book
order by book.name asc, book.publishDate desc
```

Grouping is also supported for criteria queries. You can assign any properties as group properties and they will appear in the group by clause.

```
Criteria criteria = session.createCriteria(Book.class)
    .setProjection(Projections.projectionList()
        .add(Projections.groupProperty("publishDate"))
        .add(Projections.avg("price"))
    );
List books = criteria.list();

select book.publishDate, avg(book.price)
from Book book
group by book.publishDate
```

6. Querying by example

Criteria queries can be constructed through an “example” object. The object should be an instance of the persistent class you want to query. All null properties are ignored by default.

```
Book book = new Book();
book.setName("Hibernate");
book.setPrice(new Integer(100));
Example exampleBook = Example.create(book);
Criteria criteria = session.createCriteria(Book.class)
    .add(exampleBook);
List books = criteria.list();

from Book book
where book.name = 'Hibernate' and book.price = 100
```

You can specify some creation rules for the construction of example object. For example, exclude some of the properties or enable “like” for string comparison.

```
Book book = new Book();
book.setName("%Hibernate%");
book.setPrice(new Integer(100));
Example exampleBook = Example.create(book)
    .excludeProperty("price")
    .enableLike();
Criteria criteria = session.createCriteria(Book.class)
    .add(exampleBook);
List books = criteria.list();

from Book book
where book.name like '%Hibernate%'
```