Example Class Usage Notes

如何构建一个动态的 where 子句

The example class specifies how to build a dynamic where clause. Each non-BLOB column in the table can optionally be included in the where clause. Examples are the best way to demonstrate the usage of this class.

The example class can be used to generate a virtually unlimited where clauses.

The example classes contain an inner static class called `Criteria` that holds a list of conditions that will be `anded` together in the where clause. The example class holds a list of `Criteria` objects and all the clauses from the inner classes will be `ored` together. Using different sets of `Criteria` classes allows you to generate virtually unlimited types of where clauses.

`Criteria` objects can be created with the either the `createCriteria` method or the `or` method in the example class. When the first `Criteria` object is created with the `createCriteria` method it is automatically added to the list of `Criteria` objects - this makes it easy to write a simple where clause if you don't need to `or` several other clauses together. When using the `or` method, the `Criteria` class is added to the list in all instances.

推荐做法：仅使用 or 方法来创建 Criteria 类实例

Important We recommend that you only use the `or` method for creating `Criteria` classes. We believe that this method makes for more readable code.

代码具有更好的可读性

Simple Queries 简单查询

本示例展示“如何使用生成的示例类来生成一个简单的 WHERE 子句”

This example shows how to generate a simple WHERE clause using the generated example class:

```
TestTableExample example = new TestTableExample();

example.createCriteria().andField1EqualTo(5);
```

Alternatively, the following syntax also works:

```
TestTableExample example = new TestTableExample();

example.or().andField1EqualTo(5);
```

In either above example, the dynamically generated where clause will effectively be:

```
where field1 = 5
```

Complex Queries 复杂查询

The next example shows how to generate a complex WHERE clause using the generated example class (using JSE 5.0 parameterized types):

```
TestTableExample example = new TestTableExample();

example.or()
  .andField1EqualTo(5)
  .andField2IsNull();

example.or()
  .andField3NotEqualTo(9)
```

```
    .andField4IsNotNull();

List<Integer> field5Values = new ArrayList<Integer>();
field5Values.add(8);
field5Values.add(11);
field5Values.add(14);
field5Values.add(22);

example.or()
  .andField5In(field5Values);  ID 列表筛选

example.or()
  .andField6Between(3, 7);   时间范围查询
```

In the above example, the dynamically generated where clause will effectively be:

```
where (field1 = 5 and field2 is null)
   or (field3 <> 9 and field4 is not null)
   or (field5 in (8, 11, 14, 22))
   or (field6 between 3 and 7)
```

Returned records will meet these criteria.

Distinct Queries

You can force queries to be DISTINCT by calling the `setDistinct(true)` method on any example class.

Criteria Classes 标准类型

The `Criteria` inner class includes `andXXX` methods for each field, and each standard SQL predicate including:

包括每个字段的 andXXX 方法和每个标准的 SQL 谓词

- IS NULL - meaning the related column must be NULL
- IS NOT NULL - meaning the related column must not be NULL
- = (equal) - meaning the related column must be equal to the value passed in on the method call
- <> (not equal) - meaning the related column must not be equal to the value passed in on the method call
- > (greater than) - meaning the related column must be greater than the value passed in on the method call
- >= (greater than or equal) - meaning the related column must be greater than or equal to the value passed in on the method call
- < (less than) - meaning the related column must be less than the value passed in on the method call
- <= (less than or equal) - meaning the related column must be less than or equal to the value passed in on the method call
- LIKE - meaning the related column must be "like" the value passed in on the method call. The code does not add the required '%', you must set that value yourself in the value you pass in on the method call.
- NOT LIKE - meaning the related column must be "not like" the value passed in on the method call. The code does not add the required '%', you must set that value yourself in the value you pass in on the method call.
- BETWEEN - meaning the related column must be "between" the two values passed in on the method call.
- NOT BETWEEN - meaning the related column must be "not between" the two values passed in on the method call.
- IN - meaning the related column must be one of the list of values passed in on the method call.
- NOT IN - meaning the related column must not be one of the list of values passed in on the method call.