Extending the Example Classes

In some cases it may be desirable to extended the generated example classes. You may want to add criterion that are specific to your database (such as Oracle ROWNUM support), or add criterion that are not automatically generated (such as a case insensitive search). In situations like these, you may extend the generated example class to add these additional criteria.

General Principles

MyBatis Generator (MBG) generates an "example" class for each table, unless instructed otherwise in the configuration. The "example" class is used to generate a dynamic where clause for use in the `xxxByExample` statements. The standard "example" class includes functionality for all standard SQL predicates. In some cases, it may be desirable to add additional predicates for the specific needs of your application. This may include adding support for non-standard predicates, or for using database specific functions in your where clauses.

The generated "example" class includes a nested inner class where the actual functionality of the predicates exists. This inner class is always named `GeneratedCriteria`. MBG also generates an inner class named `Criteria` which extends `GeneratedCriteria` and which you may use for adding your own functions to the example classes. The `Criteria` class will not be deleted by the Eclipse Java code merger, so you may add to it without fear of losing your changes upon regeneration.

For example, suppose there is a table called CUSTOMER. Typically, MBG would generate a class named `CustomerExample`. To add functionality to the `CustomerExample` class, you should add additional methods to the `CustomerExample.Criteria` class.

Extending vs. Plugging In

If you find that you are extending the example classes frequently, it might be more convenient for you to create a plugin to generate the additional functionality rather than hand coding the extended classes. The example below (under the heading "Single Parameter Predicates") can also be accomplished with a plugin as demonstrated by the class `org.mybatis.generator.plugins.CaseInsensitiveLikePlugin`.

Adding Predicates

MBG generates a dynamic SQL fragment that allows virtually unlimited where clauses to be created at run-time. To accomplish this, the generated SQL fragment supports four broad types of SQL predicates. For each type of SQL predicate, there is a corresponding method in the `GeneratedCriteria` inner class that can be used to add a predicate to the dynamic where clause.

1. Simple String Substitution

This type of predicate is used when there is no need for a property from the parameter object to be substituted into the where clause. Examples include:

```
FIRST_NAME is null
LAST_NAME is not null
```

The `GeneratedCriteria` class method for this predicate is:

```
addCriterion(String anyString)
```

Where "anyString" is the string to be substituted into the where clauses. This method can be used to add any kind of test to the generated where clause.

For example, suppose you wanted to use the SOUNDEX function to do a "sounds like" name search. In MySQL, the predicate should look like this:

```
SOUNDEX(FIRST_NAME) = SOUNDEX('frod')
```

This predicate is too complex to use one of the other methods, so it must be inserted into the where clause by simple string substitution. Add the following method to the `Criteria` inner class for this functionality:

```
public Criteria andFirstNameSoundsLike(String value) {
  StringBuffer sb = new StringBuffer("SOUNDEX(FIRST_NAME) = SOUNDEX('");
  sb.append(value);
  sb.append("')");

  addCriterion(sb.toString());

  return this;
}
```

The following code shows the use of this new functionality with the `selectByExample` method:

```
CustomerExample example = new CustomerExample();
Criteria criteria = example.createCriteria();
criteria.andFirstNameSoundsLike("frod");
List results = selectByExample(example);
```

This method can be used to add virtually any predicate to a where clause. However, it is generally better to use parameter substitution if possible because the problems of formatting different data types properly (most notably dates, times, and timestamps). Also, there is a chance of SQL injection issues with this method if you expose a too generic method. If possible, we suggest using one of the other methods listed below.

2. Single Parameter Predicates

This type of predicate is used when there is a single property from the parameter object to be substituted into the where clause. Examples include

```
FIRST_NAME = ?
LAST_NAME <> ?
```

The generated `Criteria` class method for this predicate is:

```
addCriterion(String anyString, Object anyObject, String propertyName)
```

Where:

anyString
is the string to be substituted into the where clause before a parameter substitution
anyObject
is the Object to be substituted into the where clause after the string substitution
propertyName
is a string denoting the property name related to this clause. This String is only used in potential error messages.

This method can be used to add simple tests related to a single parameter to the generated where clause.

For example, suppose you wanted to perform a case insensitive search on certain columns. In MySQL, the predicate could look like this:

```
upper(FIRST_NAME) like ?
```

This predicate fits the capabilities of a single parameter predicate - where the predicate is a string value followed by a single parameter. Add the following method to the `ExtendedCriteria` for this functionality:

```
public ExtendedCriteria andFirstNameLikeInsensitive(String value) {
  addCriterion("upper(FIRST_NAME) like",
    value.toUpperCase(), "firstName");

  return this;
}
```

The following code shows the use of this new functionality with the `selectByExample` method:

```
ExtendedExample example = new ExtendedExample();
ExtendedCriteria criteria = (ExtendedCriteria) example.createCriteria();
criteria.andFirstNameLikeInsensitive("fred%");
List results = selectByExample(example);
```

3. List Predicates

List predicates are used to add a variable sized list of values as parameters to a where clause. Examples include:

```
FIRST_NAME IN (?, ?, ?)
LAST_NAME NOT IN (?, ?, ?, ?)
```

This predicate is less flexible then the others because it is included specifically for the "in" and "not in" standard predicates. Nevertheless, if you find some use for it the corresponding method in the `Criteria` class is as follows:

```
addCriterion(String anyString, List listOfObjects, String propertyName)
```

Where:

anyString
is the string to be substituted into the where clause before a parameter substitution
listOfObjects
is the List of Objects to be substituted into the where clause after the string substitution (an open parenthesis will be appended before the list, list items will be comma delimited, and a close parenthesis will be appended after the list).
propertyName
is a string denoting the property name related to this clause. This String is only used in potential error messages.

4. Between Predicates

Between predicates are used to add a two parameters to a where clause in a specific format. Examples include:

```
FIRST_NAME BETWEEN ? AND ?
LAST_NAME NOT BETWEEN ? AND ?
```

This predicate is less flexible then the others because it is included specifically for the "between" and "not between" standard predicates. Nevertheless, if you find some use for it the corresponding method in the `Criteria` class is as follows:

```
addCriterion(String anyString, Object object1, Object object2, String propertyName)
```

Where:

anyString

is the string to be substituted into the where clause before a parameter substitution

object1

is the objects to be substituted into the where clause after the string substitution (the word "and" will be appended after this object).

object2

is the objects to be substituted into the where clause after the word "and".

propertyName

is a string denoting the property name related to this clause. This String is only used in potential error messages.