作者 [小码哥Java学院](http://www.jianshu.com/users/231b43e2c05f)2015.09.03 00:13\*

<http://www.jianshu.com/p/1b826d43dbaf>

# **理解MyBatis Generator Plugin**

字数1995 阅读2224 评论0 喜欢4



*Paste\_Image.png*

Plugin能够用来在MyBatis Generator生成Java和XML文件过程中修改或者添加内容；Plugin必须实现org.mybatis.generator.api.Plugin接口，在这个接口中提供了非常多的方法，所以，很自然，MBG提供了一个适配器org.mybatis.generator.api.PluginAdapter，一般情况下只需要继承这个适配器即可；  
MBG已经提供了一些内置的Plugin（这些Plugin是我们学习插件的非常好的示例），这些插件都在org.mybatis.generator.plugins包中；要能够写自己的插件，最重要的是了解插件的执行过程（生命周期）。

## **插件的生命周期**

插件拥有一个生命周期，在MBG初始化和生成的过程中，会有序的调用Plugin上不同的方法。插件的生命周期如下：

* 使用默认构造器创建；
* setContext方法调用，注入生成器上下文；
* setProperties方法调用，传入在配置文件中插件的参数；
* validate方法调用，该方法一般用于验证传给参数的正确性，如果该方法返回false，则该插件结束执行；
* 针对context中配置的每一个table：
  + initialized方法被调用，用于初始化操作，传入IntrospectedTable；
  + Java Client Methods被调用（这个地方需要注意一下，这里的Java Client Method调用和下面的Model Method，SQL Map Method的调用的前提是针对该table配置是分别需要生成client，model和SQL的，如果一个table不需要生成java client，那么这个阶段就忽略，下面两个阶段同理）:
    - clientXXXMethodGenerated(Method, TopLevelClass, IntrospectedTable)方法调用（比如clientCountByExampleMethodGenerated方法），这些方法其实就是对应Java DAO中生成对应方法时调用（那个TopLevelClass其实就是对Java类的DOM封装）【注意】，这些方法主要针对ibatis；
    - clientXXXMethodGenerated(Method, Interface, IntrospectedTable)方法调用（比如clientCountByExampleMethodGenerated方法），这些方法其实就是对应Java中Mapper生成对应方法时调用；通过返回true和false来代表该方法是否需要生成；  
      -clientGenerated(Interface, TopLevelClass, IntrospectedTable)方法调用；
  + Model Methods被调用：
    - 对每一个字段依次调用modelFieldGenerated  
      , modelGetterMethodGenerated  
      , modelSetterMethodGenerated方法（就不一个一个详细解释了，看名字就看得出来在干嘛）
    - modelExampleClassGenerated(TopLevelClass, IntrospectedTable)：用于创建XXXExample类；TopLevelClass参数同理，也是就是生成XXXExample类的DOM；
    - modelPrimaryKeyClassGenerated(TopLevelClass, IntrospectedTable)：用于创建那个主键（KeyClass）类；
    - modelBaseRecordClassGenerated(TopLevelClass, IntrospectedTable)：用于创建那个Record class（主Class）类；
    - modelRecordWithBLOBsClassGenerated(TopLevelClass, IntrospectedTable)：用于创建包含所有BLOB列的类；  
      -如果要修改这些类的生成结果，就是去修改TopLevelClass这个DOM的结构而已；
  + SQL Map Methods：这些方法主要是在生成SQL 那个mapper.xml文件时调用；
    - sqlMapXXXElementGenerated(XmlElement, IntrospectedTable)，比如sqlMapDeleteByExampleElementGenerated，其实就是在XML文件中生成对应SQL元素的时候调用该方法，我们要修改生成的SQL或者元素内容，其实就是修改那个XmlElement，XmlElement是MBG对XML文件的DOM封装；
    - sqlMapDocumentGenerated(Document, IntrospectedTable)
    - sqlMapDocument(GeneratedXmlFile, IntrospectedTable)，这两个方法都是最后生成XML的时候调用；
  + contextGenerateAdditionalJavaFiles(IntrospectedTable)方法调用（生成额外的Java文件，MBG自己是没有实现这个方法的，提供给插件一个扩展机会）；
  + contextGenerateAdditionalXmlFiles(IntrospectedTable)方法调用（同理，生成额外的XML文件，MBG自己是没有实现这个方法的，提供给插件一个扩展机会）
* contextGenerateAdditionalJavaFiles()方法调用，同contextGenerateAdditionalJavaFiles(IntrospectedTable)方法，只是没有参数而已；  
  -contextGenerateAdditionalXmlFiles()方法调用，同contextGenerateAdditionalXmlFiles(IntrospectedTable)方法，只是没有提供参数；

## **开发插件**

开发插件最好的方法就是继承org.mybatis.generator.api.PluginAdapter，然后只扩展自己需要扩展的方法；  
实现自己的方法，可以用来修改默认的MBG生成好的代码，或者添加自己额外需要生成的代码，一般可以在插件中完成：

* 可以使用自己的注解来辅助生成代码；
* 可以在类上面添加一些方法来辅助生成代码；
* 可以添加一些XML中的元素的属性配置；
* 可以添加一些额外的XML文件或者Java文件（比如一个例子就是生成MyBatisConfig.xml文件）；

再次提醒，contextXXX方法总是会被调用，而Java Client Method，Model Method和SQL Map Method是根据配置的MBG参数来选择性的执行；比如如果配置的是flat生成样式，那么modelPrimaryKeyClassGenerated(TopLevelClass, IntrospectedTable)方法就不会被调用；  
特别注意，如果一个方法返回的是boolean类型的，那么，如果该方法返回false，这个方法对应生成的代码片段（JAVA或者XML）就不会被生成了，并且，如果一个plugin返回了false，就会阻止其他的plugin的相同方法的继续执行，换句话说，配置在generatorConfig.xml中的plugin元素是有序的，这点需要特别注意。

## **MBG提供的内置的插件**

在MBG中，提供了一些插件，是我们很好的学习的素材，简单介绍下这些内置的插件：

### **org.mybatis.generator.plugins.CachePlugin**

这个插件是一个挺有用的插件，用来生成在XML中的<cache>元素（这个插件只针对MyBatis3/MyBatis3Simple有效哈）；  
很显然，这个插件需要一些配置，支持的配置属性有：cache\_eviction，cache\_flushInterval，cache\_readOnly，cache\_size，cache\_type，具体就不解释了，和cache元素的属性一一对应；  
很好的一点，在<table>元素中，可以通过定义property元素，来覆盖<plugin>元素中提供的默认值；

### **org.mybatis.generator.plugins.CaseInsensitiveLikePlugin**

这个插件用来在XXXExample类中生成大小写敏感的LIKE方法（插件本身用处不大，但是我们可以通过这个插件学习给XXXExample类添加额外的方法）

### **org.mybatis.generator.plugins.EqualsHashCodePlugin**

很明显，这个插件用来给Java模型生成equals和hashcode方法；注意下，如果Domain类有rootClass，需要重新处理下生成的代码（因为MBG是不会考虑rootClass相关内容的，甚至都不会加载rootClass，可以参考配置文件中rootClass相关说明）；

### **org.mybatis.generator.plugins.MapperConfigPlugin**

比较有用的一个插件，可以用来帮助生成一个默认的MapperConfig.xml文件骨架，在这个骨架文件中完成了本次生成的mapper.xml文件的配置；  
该插件支持的配置属性有：

* fileName：配置文件名称，默认为MapperConfig.xml；
* targetPackage：配置文件所在的包，同MBG配置文件中的所有targetPackage配置；
* targetProject：配置文件所在目录，同MBG配置文件中的所有targetProject配置；

### **org.mybatis.generator.plugins.RenameExampleClassPlugin**

这个插件可以使用正则表达式的方式来重命名生成的XXXExample类，通过配置 searchString和replaceString属性来完成（这个实现原理请参考MBG配置文件中的columnRenamingRule元素），  
一个例子，比如要xxxExample改成xxxCriteria，只需要配置：

<property name="searchString" value="Example$" /><property name="replaceString" value="Criteria" />

### **org.mybatis.generator.plugins.RowBoundsPlugin**

这个插件可以生成一个新的selectByExample方法，这个方法可以接受一个RowBounds参数，主要用来实现分页（当然，我们后面会生成我们自己的分页查询函数），这个插件只针对MyBatis3/MyBatis3Simple有效哈；

### **org.mybatis.generator.plugins.SerializablePlugin**

这个插件主要用来为生成的Java模型类添加序列化接口，并生成serialVersionUID字段；  
有两个配置项：

* addGWTInterface：这个是针对GWT的，我们忽略；
* suppressJavaInterface：这个参数是必须要填的，我们设置为false就可以了；

### **org.mybatis.generator.plugins.ToStringPlugin**

这个插件顾名思义，为生成的Java模型创建一个toString方法，（PS：一个挺有用的插件，而且这个插件生成的toString方法性能还挺高哦~~）

好了，这篇文章先到这里吧，下面再来说说怎么自己写一个插件吧；