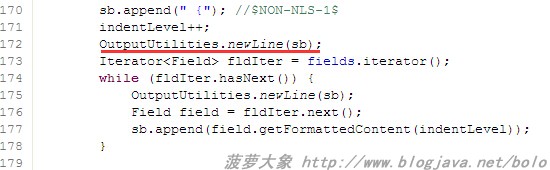
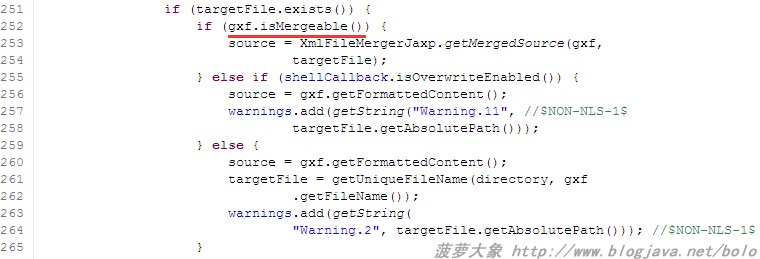
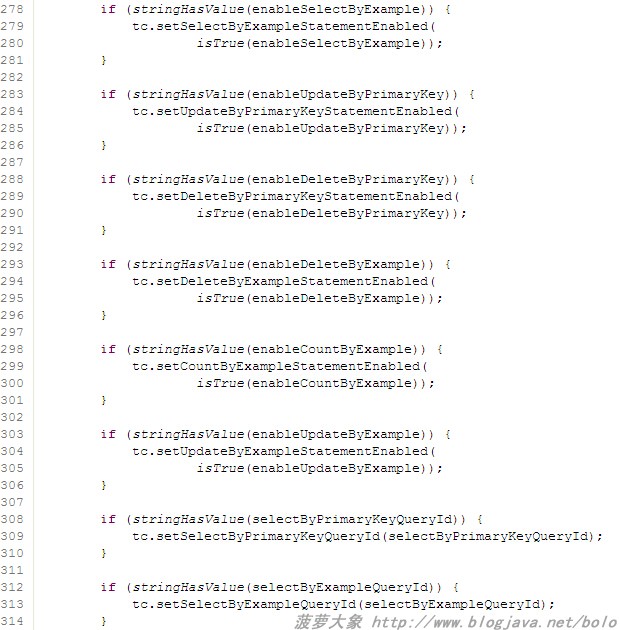
**修改一**

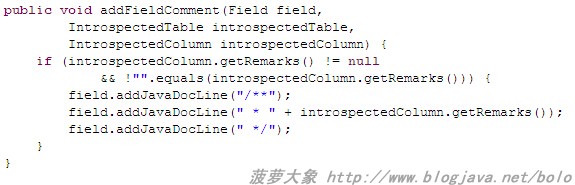
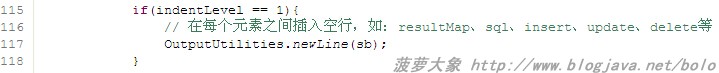
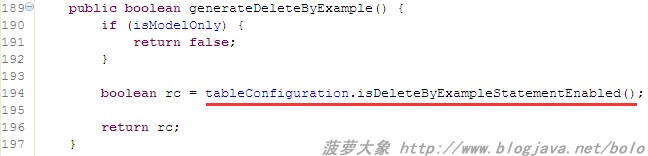
    相信很多人都用过mybatis-generator-core生成pojo和xml文件，但是这工具最后一个版本是1.3.2版，2012年7月后就没再更新了，同时因为这玩意是老外写的，可能与咱们国人的有些习惯不一致，所以在使用上总会感觉不怎么爽，大象也有这个体会，所以对源码作过一些修改，现将这些总结分享出来给需要的朋友。  
    首先大象得说一下，我对mybatis-generator-core也没有太深入的了解，仅仅是把它当成一个工具，只是感觉用的不顺手，就稍微了解了一下，顺便改了改。所以本篇没涉及到的地方，如果你知道，还请多多分享，大象感激不尽。另外，我对源码的修改主要是针对MySQL数据库，在第二篇中会讲到。  
    这个项目本身是放在googlecode上面的，但是有一道“墙”，你懂的。所以得换个方式来获得源码。  
    首先新建一个maven工程，添加下面的依赖，使用maven的Download Sources，获得mybatis-generator-core的源码。

<dependency>  
    <groupId>org.mybatis.generator</groupId>  
        <artifactId>mybatis-generator-core</artifactId>  
        <version>1.3.2</version>  
</dependency>

将源码解压放到工程里面，然后就可以开始修改了。记得把之前的依赖去掉哦！我主要说明改哪些源码，以及作用。  
    org.mybatis.generator.api.dom.OutputUtilities  
    mybatis-generator里面我觉得首先最应该改的就是OutputUtilities这个类，它里面有个xmlIndent方法是用来控制生成的xml文件中空格的缩进，默认是两个空格，但四个空格对于我们来说已经深入骨髓了，所以必须改。在sb.append("  ")里面增加两个空格就可以了。  
      
  
    org.mybatis.generator.api.dom.java.InnerClass  
    这个类就是用来控制生成pojo类以及格式化，我主要是修改属性之间以及方法之间的换行。  
    源文件：  
      
  
    第178-180行就是在每个属性后面增加一个换行，大象喜欢紧凑风格的代码，于是就把红框内的代码去掉了。再看下图，注意红线的172行，在这里增加换行是让生成的第一个属性与类的声明之间有一个空行。  
      
  
    如果你觉得getter和setter方法也不想有空行，可以把源文件的206-208行这段代码去掉，大家可以根据自己的习惯来，大象在这里把它去掉。  
    org.mybatis.generator.codegen.mybatis3.IntrospectedTableMyBatis3Impl  
    这个类的第210行有个getGeneratedXmlFiles()方法，注意下图中红线的**true**这个属性，它是用来控制是否合并生成xml文件，这里显然默认就是合并，对于我们来说，修改数据库表结构是经常发生的事情，不能每次重新生成之后，都手动去删除吧？所以这里要坚决的把它改成**false**  
      
  
    现在我们将true改成了false，问题解决了，但是它究竟是在哪里起作用的呢？请看org.mybatis.generator.api.MyBatisGenerator这个类的252行，它包含在generate方法里。  
      
  
    第一个if就会判断是否合并，上图中已经显示了isMergeable为true，所以当然是合并了，但我们将它改成false后，一定要在执行的命令行里加上overwrite参数，这样才能够让shellCallback.isOverwriteEnabled()生效，从而实现我们的xml文件覆盖生成。  
    mybatis-generator除了默认会合并xml文件外，还会默认生成一个带Example后缀的pojo，它有点类似Hibernate的Criteria对象。另外它还会在xml里面默认会生成一堆id="xxxByExample"的东西，这些都要用到那个Example类，不光如此，为了配合起来用，还有一堆与之相关的<sql>标记，所有这些在大象看来都不需要，我们使用MyBatis不就是为了它的简洁与方便吗？所以我不想看到它，需要怎么改呢？可以在配置文件里面修改。  
      
  
    加上这些属性设置，就会过滤掉我之前说的东西，而且Example类也不会再出现，呵呵，世界终于清净一些了。  
    这些属性是设置到org.mybatis.generator.config.TableConfiguration类里面起作用的，它里面有这些属性与之对应的set方法，那么配置文件中的boolean值是在哪里设置的呢？它们是在初始化解析配置文件的时候做的，完成它的类是：  
    org.mybatis.generator.config.xml.MyBatisGeneratorConfigurationParser  
      
  
    这段代码在parseTable里面，tc就是TableConfiguration，现在大家明白了吧？  
    mybatis-generator还有一个让人不爽的地方就是会在pojo和xml中生成一堆英文注释，这东西完全没什么用，我想把它去掉，像下面这样在配置文件中加上这一段。  
    

看到这里有童鞋就有意见了，泥妹的，这不是一刀切了么。我只是不想在xml里生成注释，另外想在pojo类里添加自定义注释，这个是可以做到的，下一篇大象会讲这方面的内容。

**修改二**

前面对mybatis-generator-core源码的基础修改作了说明，也可以说是最想改的地方，这一篇将在此基础上再做进一步的修改。  
    首先说说上一篇最后提到的关于自定义注释的问题，想实现这个功能就需要修改org.mybatis.generator.internal.DefaultCommentGenerator这个类。我将里面一些主要的方法讲下。  
    addJavaFileComment(CompilationUnit compilationUnit)  
    给Java文件加注释，这个注释是在文件的顶部，也就是package上面。  
      
  
    addComment(XmlElement xmlElement)  
    给生成的XML文件加注释。大象将这个方法清空了，不生成注释。  
  
    addClassComment(InnerClass innerClass,IntrospectedTable introspectedTable)  
    Java类的类注释。  
      
  
    请注意红线的getRemarks()方法，这个remarks属性在原来的FullyQualifiedTable里面是没有的，这是大象自己加上去的，就是为了保存表的注释信息。那是在哪里加进去的呢？请看org.mybatis.generator.internal.db.DatabaseIntrospector这个类，大概浏览下就会发现，数据库表以及列的信息读取，类型设置都是由它来完成的，定位到608行，正好这里是个空行，插入几行代码。如下：  
      
    这样我们就取到了表的注释信息，看到这里应该就会明白了吧？  
    另外我需要说明的是，想通过databaseMetaData.getTables()来获得表注释的童鞋，这个做法是没用的，大象已经试过了，这个结果集里面的rs.getString("REMARKS")得到的是一个空字符串，什么都木有。而且大象还想吐槽下，网上一大堆说返回的这个结果集是10列，呵呵，是的么？我debug了好久发现它还是只有5列，不知道这10列是从哪来的，请打印出10列的童鞋告之是怎么做的，我用的mysql驱动是5.1.29  
  
    addFieldComment(Field field, IntrospectedTable introspectedTable,IntrospectedColumn introspectedColumn)  
    Java属性注释。注释为空就不给属性添加。  
      
  
    addGetterComment(Method method,IntrospectedTable introspectedTable,IntrospectedColumn introspectedColumn)  
    给getter方法加注释。这里添加注释的方法和Field一样，大象把这个方法以及addSetterComment方法都清空了，因为我比较习惯把注释加到属性上面，如果你习惯在getter方法上面加注释可以自行改一下。  
    上面这些修改做完后，记得要在org.mybatis.generator.codegen.mybatis3.model.BaseRecordGenerator的61行增加下面这行代码：  
    commentGenerator.addClassComment(topLevelClass, introspectedTable);  
    因为mybatis-generator源码中本来就是不加类注释的。  
    大家都在不同的公司，要求肯定都不一样，大象没法满足所有人，只对该类做了一定程度的修改，姑且把它当做一个示例模板吧，主要是弄明白怎么改就成了。  
  
    mybatis-generator在1.3.2版里对生成xml的namespace作了修改，不再用表名当为namespace的值，而是用包结构+类名+Mapper后缀的形式设置，大象不喜欢这么复杂做法，所以这里需要改一改，只保留类名+Mapper的命名形式。去掉org.mybatis.generator.api.IntrospectedTable类的907行与908行代码。  
  
    接下来再来说说对大对象类型的处理，mybatis-generator默认会把所有jdbcType为：BINARY、BLOB、CLOB、LONGVARBINARY、LONGVARCHAR、VARBINARY这些类型都作为大对象，反应出来的效果就是生成的pojo类会多一个类名+WithBLOBs.java的文件(含有的大对象个数大于1时)，而在XML里面也会增加一个id为ResultMapWithBLOBs的resultMap，它继承BaseResultMap，大象一向喜欢简单的风格，这看得太不爽了。如果你觉得无所谓，不需要修改，请跳过这段往下看。  
    表中的text或blob等类型，通过databaseMetaData.getColumns()取出来后，它的rs.getInt("DATA\_TYPE")值是与java.sql.Types对应的。  
    text类型的值：Types.LONGVARCHAR  
    blob类型是值：Types.LONGVARBINARY  
    它用org.mybatis.generator.internal.types.JavaTypeResolverDefaultImpl解析java类型与jdbc类型，然后设置到org.mybatis.generator.api.IntrospectedColumn的jdbcTypeName属性里，最关键的地方到了，IntrospectedColumn类第156行有个isBLOBColumn()方法，它就是用来判断是否生成xxxWithBLOBs.java和ResultMapWithBLOBs的条件。当然mybatis-generator不会这么直接用，它在里面定义了一个Rules接口，由它来统一进行调用。所以我们只需要把isBLOBColumn()里面的代码都去掉，直接返回false就可以了，再结合配置文件中的columnOverride属性就能达到目的。  
      
  
    jdbcType指定的值就是生成xml后result里面对应的类型，javaType与生成的pojo里面属性类型一致，这里其实可以不定义javaType，但是当数据库的表字段有的定义为tinyint时，如果不给它指定类型，那么默认生成的java类型就是java.lang.Byte，假如你想转换类型，而且保证不会超出字段类型的大小限制，那么你可以将它的javaType设为java.lang.Integer  
    经过这么一番修改之后，输出结果已经比较理想了，但是xml文件看上去还是有点问题，元素与元素之间没空行，看起来好别扭。恩，这里要给它加上空行，所以请找到org.mybatis.generator.api.dom.xml.XmlElement类的getFormattedContent方法，这个方法有个参数：indentLevel，通过名字我们可以很直观的理解它的作用就是来控制缩进等级，它的初始值为0，这表示不缩进，当变成1时就是缩进一级，也即缩进四个空格，后面以此类推，我们应该还注意到，方法内部用到了递归，所以它是从最底元素开始，再一步步返回，所以我们就应该在indentLevel为1的元素后面加上换行。  
      
  
    在生成xml的文件里面，我们看到有一个update元素，id为updateByPrimaryKey，这个基本上用不到，但是你还不能在table里加上enableUpdateByPrimaryKey="false"，如果你这么做，虽然updateByPrimaryKey没有了，但updateByPrimaryKeySelective也会消失。所以这时请看看org.mybatis.generator.codegen.mybatis3.xmlmapper.XMLMapperGenerator这个类，getSqlMapElement()里面，有大量的addXXX方法，这些方法里面每个都用到了Rules规则来处理是否执行，比如我们在table里面加了enableDeleteByExample="false"，它对应的是addDeleteByExampleElement()，而它又调用了introspectedTable.getRules().generateDeleteByExample()方法，再进入到generateDeleteByExample()里面，我们可以看到红线部分，上一篇大象讲过，在加载配置文件的时候，TableConfiguration会将table中的这些属性设置到对应的属性里，所以说其实最后还是回到了判断enableDeleteByExample的布尔值上面。  
      
  
    上面啰嗦了一大堆，现在再来解决之前提到的问题，如何去掉id为updateByPrimaryKey的update元素，注释掉addUpdateByPrimaryKeyWithoutBLOBsElement()或直接去掉这行代码，跟踪代码你会发现，它和addUpdateByPrimaryKeySelectiveElement()的rules验证里面都用到了enableUpdateByPrimaryKey的值。或者你不想采取我说的这个办法，而是改BaseRules的generateUpdateByPrimaryKeyWithoutBLOBs()，让它直接返回false  
    最后大象再唠叨一句，这些addXXX方法的顺序决定了生成xml文件中的元素顺序，所以有代码洁癖的人可以作下调整。  
    上一篇加这一篇都是讲怎么少量的改动源码以实现自定义文件输出，写的比较凌乱，不是很系统，完全是从实用的角度出发，关键还是大象水平有限，有什么错误还请大家帮我指出来，谢谢！  
    改了这么多，总要看下效果吧，下一篇大象将写个测试看看生成的结果，然后用maven assembly将它打包生成一个zip，它将包含：  
    

**修改三**

前面对mybatis-generator-core源码进行了修改，在一定程度上做到了方便使用。这一篇先将写个测试程序看看生成的结果，然后用maven assembly将它打包生成一个zip文件。  
    新建一个类StartUp，找到org.mybatis.generator.api.ShellRunner这个类，它就是生成文件的入口类，拷贝106-117行代码，再稍微作下修改。  
      
      
    generatorConfig.xml在resources目录下面，它里面要定义输出文件的目录，如果不指定具体的路径，默认是在当前文件夹下面生成。  
      
  
    但是呢，这里有个比较蛋疼的地方，mybatis-generator-core默认是不给你生成文件夹的，不信？请看org.mybatis.generator.internal.DefaultShellCallback的第50行，当指定的目录不存在时，就抛出ShellException异常，所以输出文件也就不可能有了。这里我们注释掉51行代码，添加如下代码：  
      
      
    现在再来执行StartUp看看，发现在与src、target同级目录下生成了mybatis3目录，里面有相应的pojo与xml，打开之后看看效果，与我们改动代码后预想的结果一致。那么接下来就看看，如何将它打包输出成下面这样的效果。  
      
  
    首先我们新建一个src/main/scripts/run.bat文件，内容如下：  
    java -jar mybatis-generator-1.3.2.jar -configfile generatorConfig.xml –overwrite  
    pause  
  
    其实就是一个java命令行，里面加的参数都是ShellRunner需要的，加上pause是方便查看输出信息。  
    然后修改pom文件，既然最终输出有源文件JAR和编译后的class JAR，所以我们需要增加maven-source-plugin和maven-jar-plugin这两个插件，另外还用到了maven-assembly-plugin插件。这些plugin具体的用法，我就不讲了，我只说下要注意的地方。  
      
  
    注意红色框的内容，表示我将打包的源代码只会是原来的，增加的com.bolo.StartUp是不会包含进去的，而且这也没必要添加进去。  
      
  
    编译之后的打包同样只包含原来的代码，并且一定要在这里指定入口类。classpathPrefix是类目录，为空就是生成在当前目录下。如果写成<classpathPrefix>lib/</classpathPrefix>这样，则表示生成的mybatis-generator-1.3.2.jar会在lib目录下面。  
      
  
    上图descriptor里面的内容表明了assembly.xml的位置，它可以做很多自定义设置。  
      
  
    这里定义的id默认是加到生成的文件名后面的，但我们在上图中设置了<appendAssemblyId>false</appendAssemblyId>，所以，这个all-in-one就不会加到生成的文件名后面了，如果想要这个名字，将false改成true即可。<format>zip</format>表示将生成一个以zip为后缀的压缩包。  
      
  
    我们最终是想把几个文件都放在一个目录下面，所以outputDirectory就不需要再定义其它的名字。通过前面的输出知道，最后只会包含generatorConfig.xml和run.bat这两个文件，其它的配置文件并不是我们需要的，因此这里需要排除它们。  
    另外还有两个fileSet是指定源代码的JAR和编译class的JAR，这两个设置和前面类似。  
      
    有点补充说明：此例都是以MySQL数据库为例，所以运行是需要mysql的数据库驱动的，但我又不想在输出里面包含这个jar，因此就在generatorConfig.xml文件里面进行了配置，不清楚的可以看看。

**问题集锦**

oracle无法生成字段注释  
  
org.mybatis.generator.internal.db.ConnectionFactory  
  
Connection org.mybatis.generator.internal.db.ConnectionFactory.getConnection(JDBCConnectionConfiguration config) throws SQLException  
  
加上props.put("remarksReporting","true");  
  
就可以生成注释了

try {  
java.sql.Statement statement = databaseMetaData.getConnection().createStatement();  
java.sql.ResultSet rs = statement.executeQuery("SELECT \* FROM all\_tab\_comments WHERE TABLE\_name = '"+atn.getTableName()+"'");  
String remark = "";  
while (rs.next()) {  
remark = rs.getString("COMMENTS");  
}  
table.setRemarks(remark);  
closeResultSet(rs);  
statement.close();  
} catch (SQLException e) {  
e.printStackTrace();  
}  
  
OK获取表注释