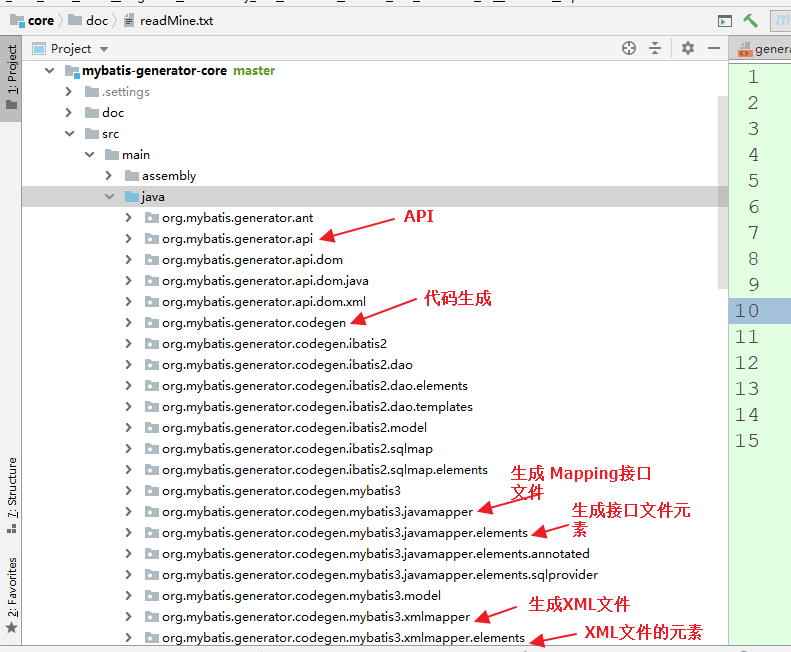
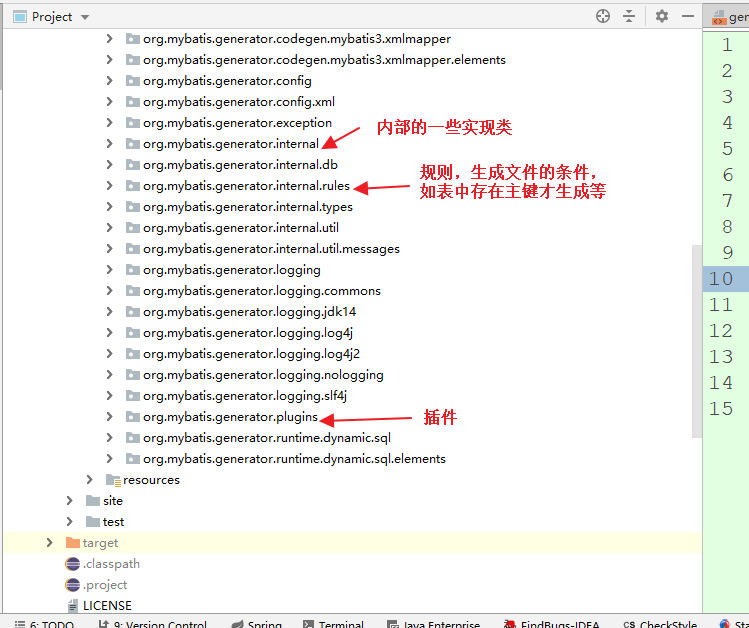
MBG1.3.7源码修改

# 项目说明

官网下载： MBG1.3.7源码

## 项目目录说明





## 实现原理

**程序入口（解读源码的入口）：**

Main-Class: org.mybatis.generator.api.ShellRunner

**主函数基本逻辑**

1、命令行参数是否合理并解析

2、读取xml配置文件并解析位内存数据以工使用

3、实例化一个shell回调函数，用于执行过程中的回调

4、实例化一个mybatis代码生成对象

5、实例化一个过程回调函数，用于是否输出过程日志等处理（或根本不需要回调）

6、调用生成方法生成代码并保存为相关文件；

# 修改文件生成的代码风格

## Mapper.xml空格改成四个空格

**修改位置：**

类 org.mybatis.generator.api.dom.OutputUtilities

方法 xmlIndent(StringBuilder sb, int indentLevel)

**修改内容：**

将

sb.append(" "); //$NON-NLS-1$

改为

// 控制生成的 xml 文件中空格的缩进，默认是两个空格 改为四个空格

sb.append(" "); //$NON-NLS-1$

## Mapper.xml每个元素之间加入空行

**修改位置：**

类 org.mybatis.generator.api.dom.xml.XmlElement

方法 public String getFormattedContent(int indentLevel)

**修改内容：**

将

if (elements.size() > 0) {

sb.append(">"); //$NON-NLS-1$

for (Element element : elements) {

OutputUtilities.newLine(sb);

sb.append(element.getFormattedContent(indentLevel + 1));

}

OutputUtilities.newLine(sb);

OutputUtilities.xmlIndent(sb, indentLevel);

sb.append("</"); //$NON-NLS-1$

sb.append(name);

sb.append('>');

} else {

sb.append(" />"); //$NON-NLS-1$

}

改为

if (elements.size() > 0) {

sb.append(">"); //$NON-NLS-1$

for (Element element : elements) {

OutputUtilities.newLine(sb);

sb.append(element.getFormattedContent(indentLevel + 1));

/\*

// 每个元素之前加入空行，如sql、insert、update等

if(1 == indentLevel){

OutputUtilities.newLine(sb);

}

\*/

}

OutputUtilities.newLine(sb);

OutputUtilities.xmlIndent(sb, indentLevel);

sb.append("</"); //$NON-NLS-1$

sb.append(name);

sb.append('>');

// 每个元素之间加入空行，如sql、insert、update等

if(1 == indentLevel){

OutputUtilities.newLine(sb);

}

} else {

sb.append(" />"); //$NON-NLS-1$

}

## 修改dao包下的名称，由原来的XXXMapper改成XXXDao （放弃修改）

**修改位置：**

类 org.mybatis.generator.api.IntrospectedTable

方法 protected void calculateJavaClientAttributes()

**修改内容：**

将

sb.append("Mapper"); //$NON-NLS-1$

改为

// 修改dao包下的名称，由原来的XXXMapper改成XXXDao

sb.append("Dao"); //$NON-NLS-1$

## 修改生成文件的注释信息

**修改后配置：**

generatorConfig.xml配置里面设置成：

<property name="suppressAllComments" value="false"/>。

修改生成文件的注释主要修改DefaultCommentGenerator类

**公共修改位置：**

类 org.mybatis.generator.internal.DefaultCommentGenerator

### 修改表或视图的备注

**说明**：解决在model或mapper的类注释中获取不到表的备注信息

**修改的类**：

org.mybatis.generator.internal.JDBCConnectionFactory

**修改的方法**：

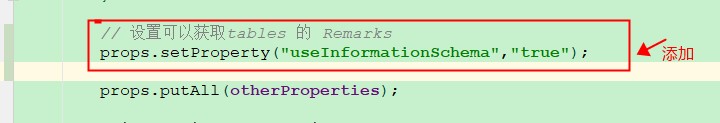
**public** Connection getConnection***()* throws** SQLException ***{***

**修改的内容**：将下面的内容 加入 设置属性 props.putAll(otherProperties); 的上一行。

代码内容：

*// 设置可以获取tables 的 Remarks*props.setProperty***(*"useInformationSchema"**,**"true"*)***;

截图所示：



### 修改mapper.xml文件里面insert和update不根据字段判断的非动态SQL（放弃修改）

**说明：**去掉mapper.xml文件里面insert和update不根据字段判断的非动态SQL，对应mapper.xml里面的

***<*insert id="insert" parameterType="xxx" *>***

***</*insert*>  
<*update id="updateByPrimaryKey"parameterType="xxx" *>***

***</*update*>***

**修改的类**：

org.mybatis.generator.codegen.mybatis3.xmlmapper.XMLMapperGenerator

**修改的方法**：

**protected** XmlElement getSqlMapElement***() {***

**修改的内容**：将addInsertElement(answer);

addUpdateByPrimaryKeyWithoutBLOBsElement(answer);

注释掉

### 注释掉Mapper.xml里面的<!-- 注释 -->

**说明：**Mapper.xml里面的<!-- 注释 -->, 实际使用未用到

**修改方法：**

**public void** addComment***(***XmlElement xmlElement***) {***

**修改内容：**全部注释或删除

### 添加Java文件注释，版权信息

**说明：**给Java文件加注释，生成版权等信息，这个注释是在文件的顶部，也就是package上面。待完善

**修改方法：**

**public void** addJavaFileComment***(***CompilationUnit compilationUnit***) {***

**修改内容：**去掉以前的内容，加入下面的新内容

**dateFormat** = **new** SimpleDateFormat***(*"yyyy-MM-dd"*)***;  
String dateStr = **dateFormat**.format***(*new** Date***())***;  
compilationUnit.addFileCommentLine***(*"/\*\* "*)***;  
compilationUnit.addFileCommentLine***(*" \* "** + compilationUnit.getType***()***.getShortName***()*** + **".java"*)***;  
compilationUnit.addFileCommentLine***(*" \* Copyright 2010-"** + dateStr.substring***(***0, 4***)*** + **" the MBG. All Rights Reserved."*)***;  
compilationUnit.addFileCommentLine***(*" \* generated by MBG"*)***;  
compilationUnit.addFileCommentLine***(*" \* generated on "** + dateStr***)***;  
compilationUnit.addFileCommentLine***(*" \*/ "*)***;

### 修改生成Model的字段注释内容

**说明：**两个addFieldComment方法均修改

**修改方法1：**

**public void** addFieldComment***(***Field field,  
 IntrospectedTable introspectedTable,  
 IntrospectedColumn introspectedColumn***) {***

**修改内容1：**去掉以前的内容，加入下面的新内容

**if *(*suppressAllComments*) {* return**;  
***}***field.addJavaDocLine***(*"/\*\*"*)***;  
StringBuilder stringBuilder = **new** StringBuilder***()***;  
stringBuilder.append***(*" \* "*)***;  
stringBuilder.append("@Fields ");

// 原始表字段

stringBuilder.append(introspectedColumn.getActualColumnName());  
stringBuilder.append***(*" "*)***;  
stringBuilder.append***(***field.getName***())***;  
stringBuilder.append***(*" "*)***;  
String remark = introspectedColumn.getRemarks***()***;  
**if *(***isNotBlank***(***remark***)) {*** stringBuilder.append***(***remark***)***;  
***}***field.addJavaDocLine***(***stringBuilder.toString***())***;  
field.addJavaDocLine***(*" \*/"*)***;

**修改方法2：**

**public void** addFieldComment***(***Field field, IntrospectedTable introspectedTable***) {***

**修改内容2：**全部注释掉或删除

### 修改Model里面Getter和Setter方法的注释

**说明：**去掉Getter和Setter方法的注释

**修改方法：**

**public void** addGetterComment***(***Method method,  
 IntrospectedTable introspectedTable,  
 IntrospectedColumn introspectedColumn***) {***

**public void** addSetterComment***(***Method method,  
 IntrospectedTable introspectedTable,  
 IntrospectedColumn introspectedColumn***) {***

**修改内容：**Getter和Setter方法 全部注释掉或删除

### 修改XXXMapper.java的接口方法的Javadoc注释

**说明：**修改XXXDao.java(XXXMapper.java)的接口普通方法的Javadoc注释

**修改方法：**

**public void** addGeneralMethodComment***(***Method method,  
 IntrospectedTable introspectedTable***) {***

**修改内容：**去掉以前的内容，加入下面的新内容

**if (suppressAllComments) {**

**return;**

**}**

**method.addJavaDocLine("/\*\*");**

**StringBuilder sb = new StringBuilder();**

**sb.append(" \* ");**

**if (method.isConstructor()) {**

**sb.append(" 构造查询条件 ");**

**}**

**String methodName = method.getName();**

**if ("toString".equals(methodName)) {**

**sb.append(" toString");**

**} else if ("selectAll".equals(methodName)) {**

**sb.append(" 查询全部");**

**} else if ("setOrderByClause".equals(methodName)) {**

**sb.append(" 设置排序字段");**

**} else if ("setDistinct".equals(methodName)) {**

**sb.append(" 设置过滤重复数据");**

**} else if ("getOredCriteria".equals(methodName)) {**

**sb.append(" 获取当前的查询条件实例");**

**} else if ("isDistinct".equals(methodName)) {**

**sb.append(" 是否过滤重复数据");**

**} else if ("getOrderByClause".equals(methodName)) {**

**sb.append(" 获取排序字段");**

**} else if ("createCriteria".equals(methodName)) {**

**sb.append(" 创建一个查询条件");**

**} else if ("createCriteriaInternal".equals(methodName)) {**

**sb.append(" 内部构建查询条件对象");**

**} else if ("clear".equals(methodName)) {**

**sb.append(" 清除查询条件");**

**} else if ("countByExample".equals(methodName)) {**

**sb.append(" 根据指定的条件获取数据库记录数");**

**} else if ("deleteByExample".equals(methodName)) {**

**sb.append(" 根据指定的条件删除数据库符合条件的记录");**

**} else if ("deleteByPrimaryKey".equals(methodName)) {**

**sb.append(" 根据主键删除数据库的记录");**

**} else if ("insert".equals(methodName)) {**

**sb.append(" 新写入数据库记录");**

**} else if ("insertSelective".equals(methodName)) {**

**sb.append(" 动态字段,写入数据库记录");**

**} else if ("selectByExample".equals(methodName)) {**

**sb.append(" 根据指定的条件查询符合条件的数据库记录");**

**} else if ("selectByPrimaryKey".equals(methodName)) {**

**sb.append(" 根据指定主键获取一条数据库记录");**

**} else if ("updateByExampleSelective".equals(methodName)) {**

**sb.append(" 动态根据指定的条件来更新符合条件的数据库记录");**

**} else if ("updateByExample".equals(methodName)) {**

**sb.append(" 根据指定的条件来更新符合条件的数据库记录");**

**} else if ("updateByPrimaryKeySelective".equals(methodName)) {**

**sb.append(" 动态字段,根据主键来更新符合条件的数据库记录");**

**} else if ("updateByPrimaryKey".equals(methodName)) {**

**sb.append(" 根据主键来更新符合条件的数据库记录");**

**}**

**final List<Parameter> parameterList = method.getParameters();**

**boolean isHaveParameter = true;**

**if (null == parameterList || parameterList.isEmpty()) {**

**isHaveParameter = false;**

**if ("or".equals(methodName)) {**

**sb.append(" 创建一个新的或者查询条件");**

**}**

**} else {**

**method.addJavaDocLine(" \*");**

**if ("or".equals(methodName)) {**

**sb.append(" 增加或者的查询条件,用于构建或者查询");**

**}**

**}**

**sb.append(" : ");**

**sb.append(introspectedTable.getFullyQualifiedTable());**

**method.addJavaDocLine(sb.toString());**

**if (isHaveParameter) {**

**for (Parameter parameter : parameterList) {**

**sb.setLength(0);**

**sb.append(" \* @param "); //$NON-NLS-1$**

**String paramterName = parameter.getName();**

**sb.append(paramterName);**

**if ("orderByClause".equals(paramterName)) {**

**sb.append(" 排序字段"); //$NON-NLS-1$**

**} else if ("distinct".equals(paramterName)) {**

**sb.append(" 是否过滤重复数据");**

**} else if ("criteria".equals(paramterName)) {**

**sb.append(" 过滤条件实例");**

**} else {**

**sb.append(" ");**

**sb.append(paramterName);**

**}**

**method.addJavaDocLine(sb.toString());**

**}**

**}**

**FullyQualifiedJavaType fullyQualifiedJavaType = method.getReturnType();**

**if(null != fullyQualifiedJavaType){**

**String returnType = fullyQualifiedJavaType.toString();**

**if(returnType.contains("List")){**

**returnType = "List";**

**} else {**

**int lastIndex = returnType.lastIndexOf(".");**

**returnType = lastIndex != -1 ? returnType.substring(lastIndex + 1) : returnType;**

**}**

**method.addJavaDocLine(" \* @return " + returnType);**

**}**

**method.addJavaDocLine(" \*/");**

### 修改生成Model的类注释内容

**说明：**为模型类添加注释。 Java代码合并应该通知不要删除整个class，万一有任何class已经做出了改变。 所以这个方法会永远使用“不要删除”注释。

**修改方法：**

**public void** addModelClassComment***(***TopLevelClass topLevelClass,  
 IntrospectedTable introspectedTable***) {***

**修改内容：**去掉以前的内容，加入下面的新内容

**if *(*suppressAllComments*) {* return**;  
***}***topLevelClass.addJavaDocLine***(*"/\*\*"*)***;  
topLevelClass.addJavaDocLine***(*" \* "*)***;  
StringBuilder stringBuilder = **new** StringBuilder***()***;  
stringBuilder.append***(*"表："*)***;  
TableConfiguration tableConfiguration = introspectedTable.getTableConfiguration***()***;  
stringBuilder.append***(***tableConfiguration.getTableName***())***;  
stringBuilder.append***(*" "*)***;  
String remark = introspectedTable.getRemarks***()***;  
**if *(***isNotBlank***(***remark***)) {*** stringBuilder.append***(***remark***)***;  
***}***stringBuilder.append***(*" 的实体类"*)***;  
topLevelClass.addJavaDocLine***(*" \* "** + stringBuilder.toString***())***;  
topLevelClass.addJavaDocLine***(*" \* "*)***;  
topLevelClass.addJavaDocLine***(*" \* "** + **"@author generated by MBG"*)***;  
**dateFormat** = **new** SimpleDateFormat***(*"yyyy-MM-dd HH:mm:ss"*)***;  
topLevelClass.addJavaDocLine***(*" \* "** + **"@date "** + **dateFormat**.format***(*new** Date***()))***;  
topLevelClass.addJavaDocLine***(*" \* @version 1.0.0"*)***;  
topLevelClass.addJavaDocLine***(*" \* "*)***;  
topLevelClass.addJavaDocLine***(*" \*/"*)***;

### 添加实体类注释

**说明：**实体类添加相应的注释，有两个方法。

**修改方法1：**

**public void** addClassComment***(***InnerClass innerClass,  
 IntrospectedTable introspectedTable***) {***

**修改内容1：**去掉以前的内容，加入下面的新内容

**if *(*suppressAllComments*) {* return**;  
***}***innerClass.addJavaDocLine***(*"/\*\*"*)***;  
innerClass.addJavaDocLine***(*" \* @Title "** + introspectedTable.getFullyQualifiedTable***()*** + **"表的实体类"*)***;  
String remark = introspectedTable.getRemarks***()***;  
**if *(***isNotBlank***(***remark***)) {*** innerClass.addJavaDocLine***(*" \* @Description "** + remark***)***;  
***}***innerClass.addJavaDocLine***(*" \* @Author generated by MBG"*)***;  
**dateFormat** = **new** SimpleDateFormat***(*"yyyy-MM-dd HH:mm:ss"*)***;  
innerClass.addJavaDocLine***(*" \* @Date "** + **dateFormat**.format***(*new** Date***()))***;  
innerClass.addJavaDocLine***(*" \* @version 1.0.0"*)***;  
innerClass.addJavaDocLine***(*" \* "*)***;  
innerClass.addJavaDocLine***(*" \*/"*)***;

**修改方法2：**

**public void** addClassComment***(***InnerClass innerClass,  
 IntrospectedTable introspectedTable, **boolean** markAsDoNotDelete***) {***

**修改内容2：**去掉以前的内容，加入下面的新内容

**public void** addClassComment***(***InnerClass innerClass,  
 IntrospectedTable introspectedTable, **boolean** markAsDoNotDelete***) {* this**.addClassComment***(***innerClass,introspectedTable***)***;  
***}***

# 修改默认插件

## 修改ToStringPlugin插件

**说明：**生成toString时，判断是否存在序列属性（serialVersionUID），存在时不生成该字段

**修改内容1：**加入静态常量

*/\*\* 序列号 \*/  
private static final* String *SVUID* = "serialVersionUID";

**修改内容2：**组装数据时，判断是否存在该属性

修改方法：

*private void* generateToString(IntrospectedTable introspectedTable,  
 TopLevelClass topLevelClass) {

修改内容：在

StringBuilder sb = *new* StringBuilder();  
*for* (Field field : topLevelClass.getFields()) {  
 String property = field.getName();

下加入

*if*(*UID*.equals(property)){  
 *continue*;  
}

## 修改ToStringPlugin插件

# 自定义插件