**Mybatis学习笔记九：自定义Generator Plugin**

原创 2016年11月16日 19:36:41

* 标签：
* [mybatis](http://so.csdn.net/so/search/s.do?q=mybatis&t=blog)
* 6183

上一章中知道了如何使用Mybatis Generator Plugin自动生成Mapper、Model和相关配置文件。但是在实际使用情况中，可能会遇到不同的个性化需求，那么就需要Generator的自定义插件来实现。

1、创建基类

在项目中，我们经常要把一些具有通用方法的实体类建立一个基类，提高代码复用程度。   
假设每一个表中都有字段id和username，且每一个表都需要类似的CRUD方法（如根据主键查XX等）。

public class BaseModel {

private Integer id;

private String username;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21

public interface BaseMapper<T extends BaseModel> {

List<T> selectAll();

int deleteByPrimaryKey(Integer id);

int insert(T record);

int insertSelective(T record);

T selectByPrimaryKey(Integer id);

int updateByPrimaryKey(T record);

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14

2、修改generatorConfig.xml

加入插件标签

<!-- 自动生成toString方法 -->

<plugin type="org.mybatis.generator.plugins.ToStringPlugin"/>

<!-- 自动生成equals方法和hashcode方法 -->

<plugin type="org.mybatis.generator.plugins.EqualsHashCodePlugin"/>

<!-- 自定义方法 -->

<plugin type="com.dfz.plugin.MapperPlugin">

<property name="targetProject" value="../Mybatis-Chapter9-GeneratorPlugin/src/main/java"/>

<property name="targetPackage" value="com.dfz.mybatis.dao"/>

</plugin>

<!-- 自定义方法 -->

<plugin type="com.dfz.plugin.SerializablePlugin"/>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

并给model和dao添加基类属性

<javaModelGenerator targetPackage="com.dfz.mybatis.model" targetProject="src/main/java">

<property name="enableSubPackages" value="true" />

<property name="trimStrings" value="true" />

<property name="rootClass" value="com.dfz.base.BaseModel"/>

</javaModelGenerator>

* 1
* 2
* 3
* 4
* 5

<javaClientGenerator type="XMLMAPPER" targetPackage="com.dfz.mybatis.dao" targetProject="src/main/java">

<property name="enableSubPackages" value="true" />

<property name="rootInterface" value="com.dfz.base.BaseMapper"/>

</javaClientGenerator>

* 1
* 2
* 3
* 4

3、创建MapperPlugin

编写自己的Plugin

public class MapperPlugin extends PluginAdapter {

private static final String DEFAULT\_DAO\_SUPER\_CLASS = "com.dfz.base.BaseMapper";

private static final String DEFAULT\_EXPAND\_DAO\_SUPER\_CLASS = "com.dfz.base.BaseExpandMapper";

private String daoTargetDir;

private String daoTargetPackage;

private String daoSuperClass;

// 扩展

private String expandDaoTargetPackage;

private String expandDaoSuperClass;

private ShellCallback shellCallback = null;

public MapperPlugin() {

shellCallback = new DefaultShellCallback(false);

}

/\*\*

\* 验证参数是否有效

\* @param warnings

\* @return

\*/

public boolean validate(List<String> warnings) {

daoTargetDir = properties.getProperty("targetProject");

boolean valid = stringHasValue(daoTargetDir);

daoTargetPackage = properties.getProperty("targetPackage");

boolean valid2 = stringHasValue(daoTargetPackage);

daoSuperClass = properties.getProperty("daoSuperClass");

if (!stringHasValue(daoSuperClass)) {

daoSuperClass = DEFAULT\_DAO\_SUPER\_CLASS;

}

expandDaoTargetPackage = properties.getProperty("expandTargetPackage");

expandDaoSuperClass = properties.getProperty("expandDaoSuperClass");

if (!stringHasValue(expandDaoSuperClass)) {

expandDaoSuperClass = DEFAULT\_EXPAND\_DAO\_SUPER\_CLASS;

}

return valid && valid2;

}

@Override

public boolean sqlMapDocumentGenerated(Document document, IntrospectedTable introspectedTable) {

XmlElement select = new XmlElement("select");

select.addAttribute(new Attribute("id", "selectAll"));

select.addAttribute(new Attribute("resultMap", "BaseResultMap"));

select.addAttribute(new Attribute("parameterType", introspectedTable.getBaseRecordType()));

select.addElement(new TextElement(" select \* from "+ introspectedTable.getFullyQualifiedTableNameAtRuntime()));

XmlElement parentElement = document.getRootElement();

parentElement.addElement(select);

return super.sqlMapDocumentGenerated(document, introspectedTable);

}

public List<GeneratedJavaFile> contextGenerateAdditionalJavaFiles(IntrospectedTable introspectedTable) {

JavaFormatter javaFormatter = context.getJavaFormatter();

List<GeneratedJavaFile> mapperJavaFiles = new ArrayList<GeneratedJavaFile>();

for (GeneratedJavaFile javaFile : introspectedTable.getGeneratedJavaFiles()) {

CompilationUnit unit = javaFile.getCompilationUnit();

FullyQualifiedJavaType baseModelJavaType = unit.getType();

String shortName = baseModelJavaType.getShortName();

GeneratedJavaFile mapperJavafile = null;

if (shortName.endsWith("Mapper")) { // 扩展Mapper

if (stringHasValue(expandDaoTargetPackage)) {

Interface mapperInterface = new Interface(

expandDaoTargetPackage + "." + shortName.replace("Mapper", "ExpandMapper"));

mapperInterface.setVisibility(JavaVisibility.PUBLIC);

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

mapperInterface.addJavaDocLine(" \* " + shortName + "扩展");

mapperInterface.addJavaDocLine(" \*/");

FullyQualifiedJavaType daoSuperType = new FullyQualifiedJavaType(expandDaoSuperClass);

mapperInterface.addImportedType(daoSuperType);

mapperInterface.addSuperInterface(daoSuperType);

mapperJavafile = new GeneratedJavaFile(mapperInterface, daoTargetDir, javaFormatter);

try {

File mapperDir = shellCallback.getDirectory(daoTargetDir, daoTargetPackage);

File mapperFile = new File(mapperDir, mapperJavafile.getFileName());

// 文件不存在

if (!mapperFile.exists()) {

mapperJavaFiles.add(mapperJavafile);

}

} catch (ShellException e) {

e.printStackTrace();

}

}

} else if (!shortName.endsWith("Example")) { // CRUD Mapper

Interface mapperInterface = new Interface(daoTargetPackage + "." + shortName + "Mapper");

mapperInterface.setVisibility(JavaVisibility.PUBLIC);

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

mapperInterface.addJavaDocLine(" \* 由MyBatis Generator工具自动生成，请不要手动修改");

mapperInterface.addJavaDocLine(" \*/");

FullyQualifiedJavaType daoSuperType = new FullyQualifiedJavaType(daoSuperClass);

// 添加泛型支持

daoSuperType.addTypeArgument(baseModelJavaType);

mapperInterface.addImportedType(baseModelJavaType);

mapperInterface.addImportedType(daoSuperType);

mapperInterface.addSuperInterface(daoSuperType);

mapperJavafile = new GeneratedJavaFile(mapperInterface, daoTargetDir, javaFormatter);

mapperJavaFiles.add(mapperJavafile);

}

}

return mapperJavaFiles;

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82
* 83
* 84
* 85
* 86
* 87
* 88
* 89
* 90
* 91
* 92
* 93
* 94
* 95
* 96
* 97
* 98
* 99
* 100
* 101
* 102
* 103
* 104
* 105
* 106
* 107
* 108
* 109
* 110
* 111
* 112
* 113
* 114
* 115
* 116

4、注意事项

1）在添加完成自定义的plugin标签后，直接mvn:mybatis-generator:generate会报classNotFound错误，这是因为maven找不到相应的包导致的。有两种解决方案：   
a）首先使用maven打包项目mvn:install，然后再把打包好的项目添加到插件的依赖中：

<plugin>

<groupId>org.mybatis.generator</groupId>

<artifactId>mybatis-generator-maven-plugin</artifactId>

<version>1.3.2</version>

<configuration>

<verbose>true</verbose>

<overwrite>true</overwrite>

</configuration>

<dependencies>

<dependency>

<groupId>com.dfz</groupId>

<artifactId>Mybatis-Chapter9-GeneratorPlugin</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

</dependencies>

</plugin>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

当然这是比较坑爹的办法，不仅很慢而且无法调试。   
b）创建一个新的类，按照mybatis-generator官方所给出的方法，代码如下：

public class TestGenerator {

private File configFile;

@Before

public void before() {

//读取mybatis参数

configFile = new File("/Users/Documents/MybatisFun/Mybatis-Chapter9-GeneratorPlugin/src/main/resources/generatorConfig.xml");

// configFile = new File("/Users/zhangsiyuan/Documents/MybatisFun/Mybatis-Chapter9-GeneratorPlugin/src/main/resources/mybatisConfig.xml");

}

@Test

public void test() throws Exception{

List<String> warnings = new ArrayList<String>();

boolean overwrite = true;

ConfigurationParser cp = new ConfigurationParser(warnings);

Configuration config = cp.parseConfiguration(configFile);

DefaultShellCallback callback = new DefaultShellCallback(overwrite);

MyBatisGenerator myBatisGenerator = new MyBatisGenerator(config, callback, warnings);

myBatisGenerator.generate(null);

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24

直接运行就可以生成代码了。   
2）plugin里头的方法挺多的，大多数用反射和动态代理实现的，需要好好看一下，以后用得到的时候再来。