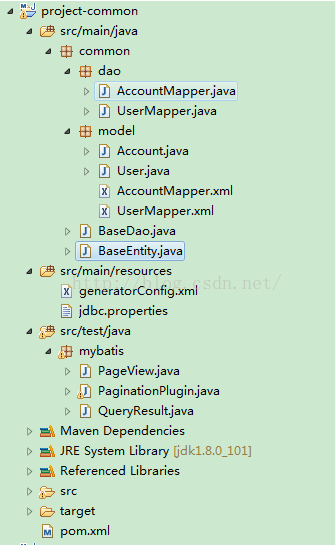
**mybatis自定义代码生成器（Generator）——自动生成model&dao代码**

原创 2016年09月12日 10:20:03

* 标签：
* [mybatis](http://so.csdn.net/so/search/s.do?q=mybatis&t=blog) /
* [maven](http://so.csdn.net/so/search/s.do?q=maven&t=blog) /
* [generator](http://so.csdn.net/so/search/s.do?q=generator&t=blog) /
* [java](http://so.csdn.net/so/search/s.do?q=java&t=blog) /
* [代码自动生成器](http://so.csdn.net/so/search/s.do?q=%E4%BB%A3%E7%A0%81%E8%87%AA%E5%8A%A8%E7%94%9F%E6%88%90%E5%99%A8&t=blog)
* 23213

花了两天的时间研究了下mybatis的generator大体了解了其生成原理以及实现过程。感觉generator做的非常不错，给开发者也留足了空间。看完之后在generator的基础上实现了自定义的生成器。代码start.....

建立了一个maven工程（common）项目结构：



----------------------------------------------------------------pom.xml-----------------------------------------------------------

**[html]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. **<project** xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"**>**
3. **<modelVersion>**4.0.0**</modelVersion>**
4. **<groupId>**com.yangchao**</groupId>**
5. **<artifactId>**project-common**</artifactId>**
6. **<version>**0.0.1-SNAPSHOT**</version>**
7. **<name>**project-common Maven Webapp**</name>**
8. **<url>**http://maven.apache.org**</url>**
9. **<dependencies>**
10. **<dependency>**
11. **<groupId>**junit**</groupId>**
12. **<artifactId>**junit**</artifactId>**
13. **<version>**3.8.1**</version>**
14. **<scope>**test**</scope>**
15. **</dependency>**
17. **<dependency>**
18. **<groupId>**javax**</groupId>**
19. **<artifactId>**javaee-api**</artifactId>**
20. **<version>**7.0**</version>**
21. **</dependency>**
23. <!-- https://mvnrepository.com/artifact/org.compass-project/compass -->
24. **<dependency>**
25. **<groupId>**org.compass-project**</groupId>**
26. **<artifactId>**compass**</artifactId>**
27. **<version>**2.0.2**</version>**
28. **</dependency>**
30. <!-- https://mvnrepository.com/artifact/org.mybatis.generator/mybatis-generator-core -->
31. **<dependency>**
32. **<groupId>**org.mybatis.generator**</groupId>**
33. **<artifactId>**mybatis-generator-core**</artifactId>**
34. **<version>**1.3.2**</version>**
35. **</dependency>**
37. **</dependencies>**
38. **</project>**

-------------------------------------------------------------------jdbc.properties--------------------------------------------------------------------

**[plain]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. jdbc.driver=com.mysql.jdbc.Driver
2. jdbc.url=jdbc:mysql://localhost:3306/demo1
3. jdbc.username=root
4. jdbc.password=root
5. initialSize=0
6. maxActive=20
7. maxIdle=20
8. minIdle=1
9. maxWait=60000

--------------------------------------------------------------------generatorConfig.xml--------------------------------------------------------------------------------------

**[html]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. **<?xml** version="1.0" encoding="UTF-8" **?>**
2. <!DOCTYPE generatorConfiguration PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN" "http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd" **>**
3. **<generatorConfiguration>**
4. <!--加载属性文件 -->
5. **<properties** resource="jdbc.properties" **/>**
6. **<context** id="context1" targetRuntime="MyBatis3"**>**
7. <!-- 实现自定义的代码生成器plugin -->
8. **<plugin** type="mybatis.PaginationPlugin" **/>**
9. **<commentGenerator>**
10. **<property** name="suppressDate" value="true" **/>**
11. <!-- 是否去除自动生成的注释 true：是 ： false:否 -->
12. **<property** name="suppressAllComments" value="true" **/>**
13. **</commentGenerator>**
14. <!-- 数据库连接URL，用户名，密码 -->
15. **<jdbcConnection** driverClass="${jdbc.driver}" connectionURL="${jdbc.url}" userId="${jdbc.username}"
16. password="${jdbc.password}" **/>**
17. <!--生成模型的包名和位置 -->
18. **<javaModelGenerator** targetPackage="common.model" targetProject="project-common/src/main/java/" **/>**
19. <!--映射文件的包名和位置 -->
20. **<sqlMapGenerator** targetPackage="common.model" targetProject="project-common/src/main/java/" **/>**
21. <!--DAO的包名和位置 -->
22. **<javaClientGenerator** targetPackage="common.dao" targetProject="project-common/src/main/java" type="XMLMAPPER" **/>**
23. <!--要生成哪些表 -->
24. **<table** tableName="%" enableSelectByExample="false" enableDeleteByExample="false"
25. enableCountByExample="false" enableUpdateByExample="false"
26. selectByExampleQueryId="false"**>**
27. **<property** name="rootClass" value="common.BaseEntity" **/>**
28. **</table>**
29. **</context>**
30. **</generatorConfiguration>**

-------------------------------------------------------PaginationPlugin.java------------------------------------------------------

**[java]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. /\*\*
2. \* @项目名称：project-common
3. \* @类名称：PaginationPlugin
4. \* @类描述：自定义代码生成器
5. \* @创建人：YangChao
6. \* @作者单位：北京宝库在线网络技术有限公司
7. \* @联系方式：YangChao@baoku.com
8. \* @创建时间：2016年9月5日 下午3:14:38
9. \* @version 1.0.0
10. \*/
11. **public** **class** PaginationPlugin **extends** PluginAdapter {
12. /\*\*
13. \* 生成dao
14. \*/
15. @Override
16. **public** **boolean** clientGenerated(Interface interfaze, TopLevelClass topLevelClass, IntrospectedTable introspectedTable) {
17. FullyQualifiedJavaType fqjt = **new** FullyQualifiedJavaType("BaseDao<" + introspectedTable.getBaseRecordType()  + ">");
18. FullyQualifiedJavaType imp = **new** FullyQualifiedJavaType("common.BaseDao");
19. interfaze.addSuperInterface(fqjt);// 添加 extends BaseDao<User>
20. interfaze.addImportedType(imp);// 添加import common.BaseDao;
21. interfaze.getMethods().clear();
22. **return** **true**;
23. }
25. /\*\*
26. \* 生成实体中每个属性
27. \*/
28. @Override
29. **public** **boolean** modelGetterMethodGenerated(Method method, TopLevelClass topLevelClass,
30. IntrospectedColumn introspectedColumn, IntrospectedTable introspectedTable, ModelClassType modelClassType) {
31. **return** **true**;
32. }
34. /\*\*
35. \* 生成实体
36. \*/
37. @Override
38. **public** **boolean** modelBaseRecordClassGenerated(TopLevelClass topLevelClass, IntrospectedTable introspectedTable) {
39. addSerialVersionUID(topLevelClass, introspectedTable);
40. **return** **super**.modelBaseRecordClassGenerated(topLevelClass, introspectedTable);
41. }
43. /\*\*
44. \* 生成mapping
45. \*/
46. @Override
47. **public** **boolean** sqlMapGenerated(GeneratedXmlFile sqlMap, IntrospectedTable introspectedTable) {
48. **return** **super**.sqlMapGenerated(sqlMap, introspectedTable);
49. }
51. /\*\*
52. \* 生成mapping 添加自定义sql
53. \*/
54. @Override
55. **public** **boolean** sqlMapDocumentGenerated(Document document, IntrospectedTable introspectedTable) {
56. String tableName = introspectedTable.getAliasedFullyQualifiedTableNameAtRuntime();// 数据库表名
57. List<IntrospectedColumn> columns = introspectedTable.getAllColumns();
58. XmlElement parentElement = document.getRootElement();
60. // 添加sql——where
61. XmlElement sql = **new** XmlElement("sql");
62. sql.addAttribute(**new** Attribute("id", "sql\_where"));
63. XmlElement where = **new** XmlElement("where");
64. StringBuilder sb = **new** StringBuilder();
65. **for** (IntrospectedColumn introspectedColumn : introspectedTable.getNonPrimaryKeyColumns()) {
66. XmlElement isNotNullElement = **new** XmlElement("if"); //$NON-NLS-1$
67. sb.setLength(0);
68. sb.append(introspectedColumn.getJavaProperty());
69. sb.append(" != null"); //$NON-NLS-1$
70. isNotNullElement.addAttribute(**new** Attribute("test", sb.toString())); //$NON-NLS-1$
71. where.addElement(isNotNullElement);
73. sb.setLength(0);
74. sb.append(" and ");
75. sb.append(MyBatis3FormattingUtilities.getEscapedColumnName(introspectedColumn));
76. sb.append(" = "); //$NON-NLS-1$
77. sb.append(MyBatis3FormattingUtilities.getParameterClause(introspectedColumn));
78. isNotNullElement.addElement(**new** TextElement(sb.toString()));
79. }
80. sql.addElement(where);
81. parentElement.addElement(sql);
83. //添加getList
84. XmlElement select = **new** XmlElement("select");
85. select.addAttribute(**new** Attribute("id", "getList"));
86. select.addAttribute(**new** Attribute("resultMap", "BaseResultMap"));
87. select.addAttribute(**new** Attribute("parameterType", introspectedTable.getBaseRecordType()));
88. select.addElement(**new** TextElement(" select \* from "+ introspectedTable.getFullyQualifiedTableNameAtRuntime()));
90. XmlElement include = **new** XmlElement("include");
91. include.addAttribute(**new** Attribute("refid", "sql\_where"));
93. select.addElement(include);
94. parentElement.addElement(select);
96. **return** **super**.sqlMapDocumentGenerated(document, introspectedTable);
97. }
99. @Override
100. **public** **boolean** sqlMapUpdateByPrimaryKeyWithoutBLOBsElementGenerated(XmlElement element,
101. IntrospectedTable introspectedTable) {
102. **return** **false**;
103. }
105. @Override
106. **public** **boolean** sqlMapInsertElementGenerated(XmlElement element, IntrospectedTable introspectedTable) {
107. **return** **false**;
108. }
110. @Override
111. **public** **boolean** sqlMapSelectByExampleWithoutBLOBsElementGenerated(XmlElement element,
112. IntrospectedTable introspectedTable) {
113. // LIMIT5,10; // 检索记录行 6-15
114. //      XmlElement isNotNullElement = new XmlElement("if");//$NON-NLS-1$
115. //      isNotNullElement.addAttribute(new Attribute("test", "limitStart != null and limitStart >=0"));//$NON-NLS-1$ //$NON-NLS-2$
116. // isNotNullElement.addElement(new
117. // TextElement("limit ${limitStart} , ${limitEnd}"));
118. // element.addElement(isNotNullElement);
119. // LIMIT 5;//检索前 5个记录行
120. **return** **super**.sqlMapSelectByExampleWithoutBLOBsElementGenerated(element, introspectedTable);
121. }
123. /\*\*
124. \* mapping中添加方法
125. \*/
126. // @Override
127. **public** **boolean** sqlMapDocumentGenerated2(Document document, IntrospectedTable introspectedTable) {
128. String tableName = introspectedTable.getAliasedFullyQualifiedTableNameAtRuntime();// 数据库表名
129. List<IntrospectedColumn> columns = introspectedTable.getAllColumns();
130. // 添加sql
131. XmlElement sql = **new** XmlElement("select");
133. XmlElement parentElement = document.getRootElement();
134. XmlElement deleteLogicByIdsElement = **new** XmlElement("update");
135. deleteLogicByIdsElement.addAttribute(**new** Attribute("id", "deleteLogicByIds"));
136. deleteLogicByIdsElement
137. .addElement(**new** TextElement(
138. "update "
139. + tableName
140. + " set deleteFlag = #{deleteFlag,jdbcType=INTEGER} where id in "
141. + " <foreach item=\"item\" index=\"index\" collection=\"ids\" open=\"(\" separator=\",\" close=\")\">#{item}</foreach> "));
143. parentElement.addElement(deleteLogicByIdsElement);
144. XmlElement queryPage = **new** XmlElement("select");
145. queryPage.addAttribute(**new** Attribute("id", "queryPage"));
146. queryPage.addAttribute(**new** Attribute("resultMap", "BaseResultMap"));
147. queryPage.addElement(**new** TextElement("select "));
149. XmlElement include = **new** XmlElement("include");
150. include.addAttribute(**new** Attribute("refid", "Base\_Column\_List"));
152. queryPage.addElement(include);
153. queryPage.addElement(**new** TextElement(" from " + tableName + " ${sql}"));
154. parentElement.addElement(queryPage);
155. **return** **super**.sqlMapDocumentGenerated(document, introspectedTable);
156. }
158. **private** **void** addSerialVersionUID(TopLevelClass topLevelClass, IntrospectedTable introspectedTable) {
159. CommentGenerator commentGenerator = context.getCommentGenerator();
160. Field field = **new** Field();
161. field.setVisibility(JavaVisibility.PRIVATE);
162. field.setType(**new** FullyQualifiedJavaType("long"));
163. field.setStatic(**true**);
164. field.setFinal(**true**);
165. field.setName("serialVersionUID");
166. field.setInitializationString("1L");
167. commentGenerator.addFieldComment(field, introspectedTable);
168. topLevelClass.addField(field);
169. }
171. /\*
172. \* Dao中添加方法
173. \*/
174. **private** Method generateDeleteLogicByIds(Method method, IntrospectedTable introspectedTable) {
175. Method m = **new** Method("deleteLogicByIds");
176. m.setVisibility(method.getVisibility());
177. m.setReturnType(FullyQualifiedJavaType.getIntInstance());
178. m.addParameter(**new** Parameter(FullyQualifiedJavaType.getIntInstance(), "deleteFlag", "@Param(\"deleteFlag\")"));
179. m.addParameter(**new** Parameter(**new** FullyQualifiedJavaType("Integer[]"), "ids", "@Param(\"ids\")"));
180. context.getCommentGenerator().addGeneralMethodComment(m, introspectedTable);
181. **return** m;
182. }
184. /\*
185. \* 实体中添加属性
186. \*/
187. **private** **void** addLimit(TopLevelClass topLevelClass, IntrospectedTable introspectedTable, String name) {
188. CommentGenerator commentGenerator = context.getCommentGenerator();
189. Field field = **new** Field();
190. field.setVisibility(JavaVisibility.PROTECTED);
191. field.setType(FullyQualifiedJavaType.getIntInstance());
192. field.setName(name);
193. field.setInitializationString("-1");
194. commentGenerator.addFieldComment(field, introspectedTable);
195. topLevelClass.addField(field);
196. **char** c = name.charAt(0);
197. String camel = Character.toUpperCase(c) + name.substring(1);
198. Method method = **new** Method();
199. method.setVisibility(JavaVisibility.PUBLIC);
200. method.setName("set" + camel);
201. method.addParameter(**new** Parameter(FullyQualifiedJavaType.getIntInstance(), name));
202. method.addBodyLine("this." + name + "=" + name + ";");
203. commentGenerator.addGeneralMethodComment(method, introspectedTable);
204. topLevelClass.addMethod(method);
205. method = **new** Method();
206. method.setVisibility(JavaVisibility.PUBLIC);
207. method.setReturnType(FullyQualifiedJavaType.getIntInstance());
208. method.setName("get" + camel);
209. method.addBodyLine("return " + name + ";");
210. commentGenerator.addGeneralMethodComment(method, introspectedTable);
211. topLevelClass.addMethod(method);
212. }
214. **public** **boolean** validate(List<String> arg0) {
215. **return** **true**;
216. }
218. **public** **static** **void** generate() {
219. String config = PaginationPlugin.**class**.getClassLoader().getResource("mybatisConfig.xml").getFile();
220. String[] arg = { "-configfile", config, "-overwrite" };
221. ShellRunner.main(arg);
222. }
224. **public** **static** **void** main(String[] args) {
225. generate();
226. }
227. }

------------------------------------------------BaseDao.java--------------------------------------------

**[java]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. /\*\*
2. \* @项目名称：project-common
3. \* @类名称：BaseDao
4. \* @类描述：
5. \* @创建人：YangChao
6. \* @作者单位：北京宝库在线网络技术有限公司
7. \* @联系方式：YangChao@baoku.com
8. \* @创建时间：2016年9月5日 下午2:51:19
9. \* @version 1.0.0
10. \*/
11. **public** **interface** BaseDao<T> {
12. **public** T selectByPrimaryKey(Integer id);
14. **public** **int** deleteByPrimaryKey(Integer id);
16. **public** **int** insertSelective(T t);
18. **public** **int** updateByPrimaryKeySelective(T t);
20. **public** List<T> getList(T t);
22. // 获取数量
23. **public** **int** getCountSelective(T t);
25. /\*\*
26. \*
27. \* @Title: findPage
28. \* @Description: TODO()
29. \* @param page
30. \*            分页参数
31. \* @param sql
32. \*            mybatis sql语句
33. \* @param values
34. \*            命名参数,按名称绑定
35. \* @return 分页查询结果, 附带结果列表及所有查询时的参数.
36. \* @author YangChao
37. \* @date 2016年9月7日 下午5:30:28
38. \*/
39. **public** PageView<T> findPage(**final** PageView<T> page, **final** String sql, **final** Map<String, Object> values);
40. }

-------------------------------------------------BaseEntity.java---------------------------------------------------

**[java]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. /\*\*
2. \* @项目名称：project-common
3. \* @类名称：BaseEntity
4. \* @类描述：所有实体类的父类。可将公共的属性所有类序列化集中在此类中
5. \* @创建人：YangChao
6. \* @作者单位：北京宝库在线网络技术有限公司
7. \* @联系方式：YangChao@baoku.com
8. \* @创建时间：2016年9月5日 上午11:37:02
9. \* @version 1.0.0
10. \*/
11. **public** **abstract** **class** BaseEntity **implements** Serializable {
12. **private** **static** **final** **long** serialVersionUID = 1L;
13. **private** Integer id;
15. **public** Integer getId() {
16. **return** id;
17. }
19. **public** **void** setId(Integer id) {
20. **this**.id = id;
21. }
23. }

---------------------------------------------------------------------------------**最终生成dao&&model-**-----------------------------------------------------------------------------------------------

--dao--

UserMapper.java:

**[java]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. **package** common.dao;
3. **import** common.BaseDao;
4. **import** common.model.User;
6. **public** **interface** UserMapper **extends** BaseDao<User> {
7. }

---model---

User.java

**[java]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. **package** common.model;
3. **import** common.BaseEntity;
5. **public** **class** User **extends** BaseEntity {
6. **private** Integer accountId;
8. **private** String loginname;
10. **private** String password;
12. **private** Integer status;
14. **private** **static** **final** **long** serialVersionUID = 1L;
16. **public** Integer getAccountId() {
17. **return** accountId;
18. }
20. **public** **void** setAccountId(Integer accountId) {
21. **this**.accountId = accountId;
22. }
24. **public** String getLoginname() {
25. **return** loginname;
26. }
28. **public** **void** setLoginname(String loginname) {
29. **this**.loginname = loginname;
30. }
32. **public** String getPassword() {
33. **return** password;
34. }
36. **public** **void** setPassword(String password) {
37. **this**.password = password;
38. }
40. **public** Integer getStatus() {
41. **return** status;
42. }
44. **public** **void** setStatus(Integer status) {
45. **this**.status = status;
46. }
47. }

UserMapper.xml

**[html]** [view plain](http://blog.csdn.net/yangchao13341408947/article/details/52510766) [copy](http://blog.csdn.net/yangchao13341408947/article/details/52510766)

1. **<?xml** version="1.0" encoding="UTF-8" **?>**
2. <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" **>**
3. **<mapper** namespace="common.dao.UserMapper" **>**
4. **<resultMap** id="BaseResultMap" type="common.model.User" **>**
5. **<id** column="id" property="id" jdbcType="INTEGER" **/>**
6. **<result** column="account\_id" property="accountId" jdbcType="INTEGER" **/>**
7. **<result** column="loginname" property="loginname" jdbcType="VARCHAR" **/>**
8. **<result** column="password" property="password" jdbcType="VARCHAR" **/>**
9. **<result** column="status" property="status" jdbcType="INTEGER" **/>**
10. **</resultMap>**
11. **<sql** id="Base\_Column\_List" **>**
12. id, account\_id, loginname, password, status
13. **</sql>**
14. **<select** id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Integer" **>**
15. select
16. **<include** refid="Base\_Column\_List" **/>**
17. from user
18. where id = #{id,jdbcType=INTEGER}
19. **</select>**
20. **<delete** id="deleteByPrimaryKey" parameterType="java.lang.Integer" **>**
21. delete from user
22. where id = #{id,jdbcType=INTEGER}
23. **</delete>**
24. **<insert** id="insertSelective" parameterType="common.model.User" **>**
25. insert into user
26. **<trim** prefix="(" suffix=")" suffixOverrides="," **>**
27. **<if** test="id != null" **>**
28. id,
29. **</if>**
30. **<if** test="accountId != null" **>**
31. account\_id,
32. **</if>**
33. **<if** test="loginname != null" **>**
34. loginname,
35. **</if>**
36. **<if** test="password != null" **>**
37. password,
38. **</if>**
39. **<if** test="status != null" **>**
40. status,
41. **</if>**
42. **</trim>**
43. **<trim** prefix="values (" suffix=")" suffixOverrides="," **>**
44. **<if** test="id != null" **>**
45. #{id,jdbcType=INTEGER},
46. **</if>**
47. **<if** test="accountId != null" **>**
48. #{accountId,jdbcType=INTEGER},
49. **</if>**
50. **<if** test="loginname != null" **>**
51. #{loginname,jdbcType=VARCHAR},
52. **</if>**
53. **<if** test="password != null" **>**
54. #{password,jdbcType=VARCHAR},
55. **</if>**
56. **<if** test="status != null" **>**
57. #{status,jdbcType=INTEGER},
58. **</if>**
59. **</trim>**
60. **</insert>**
61. **<update** id="updateByPrimaryKeySelective" parameterType="common.model.User" **>**
62. update user
63. **<set** **>**
64. **<if** test="accountId != null" **>**
65. account\_id = #{accountId,jdbcType=INTEGER},
66. **</if>**
67. **<if** test="loginname != null" **>**
68. loginname = #{loginname,jdbcType=VARCHAR},
69. **</if>**
70. **<if** test="password != null" **>**
71. password = #{password,jdbcType=VARCHAR},
72. **</if>**
73. **<if** test="status != null" **>**
74. status = #{status,jdbcType=INTEGER},
75. **</if>**
76. **</set>**
77. where id = #{id,jdbcType=INTEGER}
78. **</update>**
79. **<sql** id="sql\_where" **>**
80. **<where** **>**
81. **<if** test="accountId != null" **>**
82. and account\_id = #{accountId,jdbcType=INTEGER}
83. **</if>**
84. **<if** test="loginname != null" **>**
85. and loginname = #{loginname,jdbcType=VARCHAR}
86. **</if>**
87. **<if** test="password != null" **>**
88. and password = #{password,jdbcType=VARCHAR}
89. **</if>**
90. **<if** test="status != null" **>**
91. and status = #{status,jdbcType=INTEGER}
92. **</if>**
93. **</where>**
94. **</sql>**
95. **<select** id="getList" resultMap="BaseResultMap" parameterType="common.model.User" **>**
96. select \* from user
97. **<include** refid="sql\_where" **/>**
98. **</select>**
99. **</mapper>**

-----------------------------------代码end------------------------------------

心得：由于开始使用原始的generator生成代码存在，生成后的代码继承结构不存在，重复性代码过多。其实生成所有的代码都是可自定义的，在此我只是将dao中所有的crud方法提取出去放到BaseDao中，以方便之后在具体的dao中添加特殊的需求，让代码简单明了。

目前您尚未登录，请 [登录](https://passport.csdn.net/account/login?from=http%3A%2F%2Fblog.csdn.net%2Fyangchao13341408947%2Farticle%2Fdetails%2F52510766) 或 [注册](http://passport.csdn.net/account/register?from=http%3A%2F%2Fblog.csdn.net%2Fyangchao13341408947%2Farticle%2Fdetails%2F52510766) 后进行评论

* [yang420097009](http://blog.csdn.net/yang420097009)

2017-09-07 11:084楼

* 上传代码啊
* [回复](http://blog.csdn.net/yangchao13341408947/article/details/52510766#reply)
* [qq\_27699835](http://blog.csdn.net/qq_27699835)

2017-04-18 13:503楼

* XML Parser Errors occurred:  
    
  This is not a MyBatis Generator Configuration File  
    
  出这个错误呢？
* [回复](http://blog.csdn.net/yangchao13341408947/article/details/52510766#reply) 1条回复
* [kinganpo](http://blog.csdn.net/kinganpo)

2017-02-21 16:542楼

* 这么好的分享为什么有人踩呢？
* [回复](http://blog.csdn.net/yangchao13341408947/article/details/52510766#reply)

查看 6 条热评