链滴

# 使用 thymeleaf 生成 mybatis 模板

作者：xiaodaojava
原文链接：https://ld246.com/article/1563869858459
来源网站：链滴

# 技术背景

我们在使用mybatis做数据库访问的时候,有很多重复的东西要写,如DAO里面的增删改查,mapper/provder里面的各种东东,还有实体类,其实这些都是可以通过名称关联起来的,因此就有这么一个需求,去写个工具类,根据模板去生成这些文件

# 以前的解决方案及问题

以前公司里面有前辈写过用的是beetl做的模板处理,没什么问题,之所以改用thymeleaf原因有以下两点:

1. thymeleaf是springboot官方指定模板处理

2. 想别的项目都依赖于这个工具包,所以工具包的依赖要尽可能少,在web应用已经引入了thymeleaf的提下,尽可能不再引包了

# 引入依赖

gradle:

```
compile group: 'org.thymeleaf', name: 'thymeleaf', version: '3.0.11.RELEASE'
compile group: 'ognl', name: 'ognl', version: '3.2.10'
```

maven:

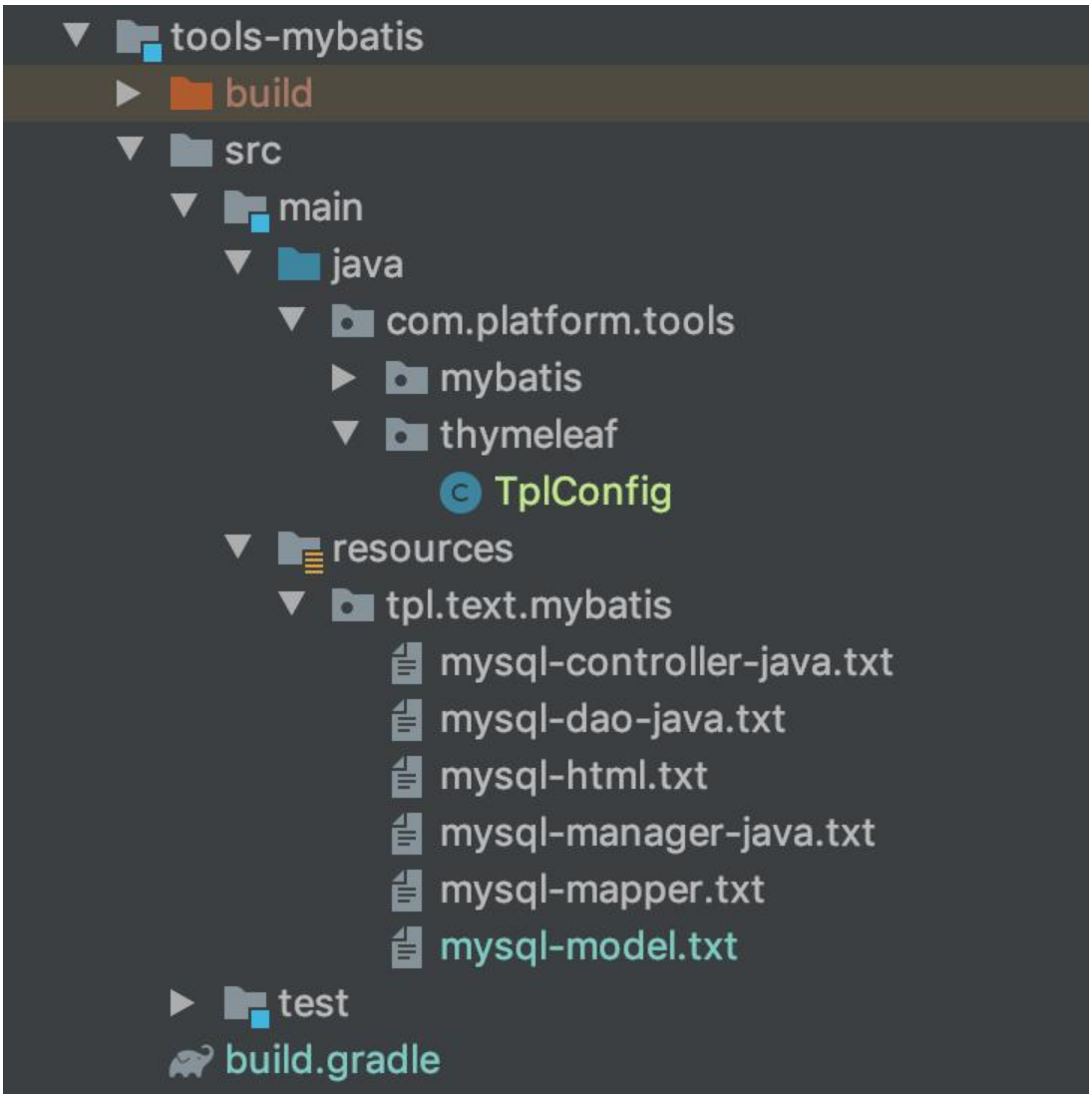
```
<dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf</artifactId>
    <version>3.0.11.RELEASE</version>
</dependency>
<dependency>
```

```
    <groupId>ognl</groupId>
    <artifactId>ognl</artifactId>
    <version>3.2.10</version>
</dependency>
```

# 配置Thymeleaf

```
public class TplConfig {

    /**
     * 使用静态内部类的方式来实现单例模式
     */
    static class InitTplConfig{
        final static TemplateEngine templateEngine = new TemplateEngine();
        static TemplateEngine getEngine(){
            return templateEngine;
        }
    }

    /**
     * 别的地方想使用templateEngine来处理模板,调用这个方法来获取
     * @return
     */
    public static TemplateEngine getTemplateEngine() {
        final TemplateEngine templateEngine = InitTplConfig.getEngine();
        // 处理txt的
        templateEngine.addTemplateResolver(textTemplateResolver());

        return templateEngine;
    }


    /**
     * 配置模板的位置,前缀,后缀等
     * @return
     */
    private static ITemplateResolver textTemplateResolver() {
        final ClassLoaderTemplateResolver templateResolver = new ClassLoaderTemplateResolv
r();
        templateResolver.setOrder(1);
        templateResolver.setResolvablePatterns(Collections.singleton("text/*"));
        templateResolver.setPrefix("/tpl/");
        templateResolver.setSuffix(".txt");
        templateResolver.setTemplateMode(TemplateMode.TEXT);
        templateResolver.setCharacterEncoding("UTF-8");
        templateResolver.setCacheable(false);
        return templateResolver;
    }

}
```

与此同时,对应的项目结构为:




# 获取数据库表的相关信息

这块有两种方式:

1. 通过jdbc的方式,比较传统,注册驱动,获取连接等等

2. 直接注入datasource,然后通过datasource.getConnection获取连接,因为现在开发基本上都是sprig环境,本文采用第2种方式

```
/** 存数据库类型和java类型的映射 */
  private Map<String,String> property2BeanMap = new HashMap<>();

  public MybatisGenerateUtils(DataSource dataSource) {
    this.dataSource = dataSource;
    property2BeanMap.put("BIGINT UNSIGNED","Long");
    property2BeanMap.put("DATETIME","Date");
    property2BeanMap.put("TIMESTAMP","Date");
    property2BeanMap.put("VARCHAR","String");
    property2BeanMap.put("DECIMAL","BigDecimal");
    property2BeanMap.put("BIGINT","Long");
    property2BeanMap.put("TEXT","String");
```

```
        property2BeanMap.put("TINYINT","Integer");
        property2BeanMap.put("INT","Integer");
        property2BeanMap.put("BIT","Integer");
        property2BeanMap.put("CHAR","String");
    }

 /**
     * 获取mysql里面的表信息,读取表里面所有字段,
     * 并封到java实体类里面
     *
     * @param tableName
     * @return
     */
    public  SqlTable getTableInfo(String dbName, String tableName) {

        SqlTable table = new SqlTable();
 //把tableName转成驼峰式,并且首子母改成大写
        String javaTableName = StringUtils.first2BigLetter(StringUtils.underScope2Camel(tableN
me));
        table.setTableName(tableName).setJavaTableName(javaTableName);
        Connection conn = null;
        try {
            conn = dataSource.getConnection();

            DatabaseMetaData metaData = conn.getMetaData();
            ResultSet resultSet = metaData.getColumns(null, null, tableName, "%");
            String columnName;
            String columnType;

            List<SqlField> fieldList = new ArrayList<>();
            while (resultSet.next()) {
                if (!resultSet.getString("TABLE_CAT").equals(dbName)) {
                    continue;
                }
                SqlField field = new SqlField();
                columnName = resultSet.getString("COLUMN_NAME");
                columnType = resultSet.getString("TYPE_NAME");
                String remark = resultSet.getString("REMARKS");
                field.setName(columnName).setRemark(remark).setType(columnType);
                field.setCamelName(StringUtils.underScope2Camel(columnName));
        field.setJavaType(property2BeanMap.get(columnType));
                fieldList.add(field);
            }
            table.setFieldList(fieldList).setTableName(tableName);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        return table;
    }
```

SqlTable.java内容如下:

```
    /** 表名 */
    private String tableName;
```

```
    /** 对应的java实体类的名称 */
    private String javaTableName;

    /** 表里面的字段名 */
    private List<SqlField> fieldList;
```

SqlField.java内容如下:

```
/**字段名称*/
    private String name;

    /**驼蜂式命名*/
    private String camelName;

    /**字段类型*/
    private String type;

    /**java实体类的类型*/
    private String javaType;

    /**备注*/
    private String remark;

    /** 数据库类型 */
    private String jdbcType;
```

数据库的模板文件如下:

```
package com.platform.forest.models;

/**
 * @author lixiang
 */
public class [(${table.javaTableName})]DO {

   [# th:each="field : ${table.fieldList}"]
   /** [(${field.remark})]  */
   private [(${field.javaType})] [(${field.camelName})];

   [/]


    /**
     * 这两个方法是用建设者模式来build DO
     */
    public static [(${table.javaTableName})]DO create(){
        return new [(${table.javaTableName})]DO();
    }

    public [(${table.javaTableName})]DO build(){
        return this;
    }
```

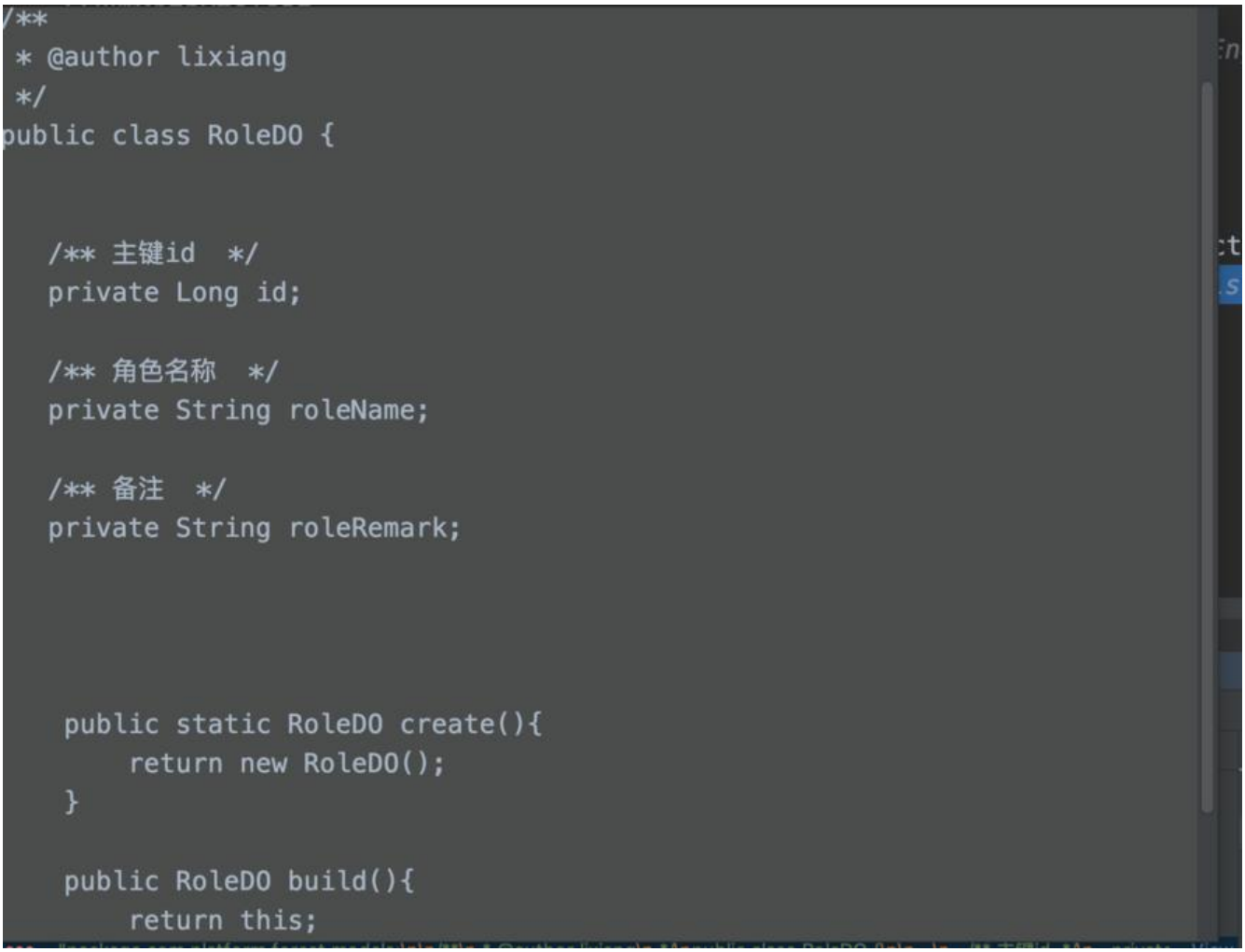
```
}
```

# 最后的调用

推荐这些使用单元测试的方式来生成这些东东

```
public class MybatisTests extends ToolsApplicationTests {

    @Autowired
    private DataSource dataSource;

    @Test
    public void testGenerate() {
        //获取数据库信息
        MybatisGenerateUtils utils = new MybatisGenerateUtils(dataSource);
        SqlTable tableInfo = utils.getTableInfo("passport", "role");
        //初始化thymeleaf
        TemplateEngine engine = TplConfig.getTemplateEngine();
        final Context ctx = new Context(Locale.CHINA);
        Map<String, Object> map = new HashMap<>();
        map.put("table",tableInfo);
        ctx.setVariables(map);
        String process = engine.process("text/mybatis/mysql-model.txt", ctx);
        System.out.println(process);
    }
}
```

# 最后的成果



```
/**
 * @author lixiang
 */
public class RoleDO {


    /** 主键id  */
    private Long id;

    /** 角色名称  */
    private String roleName;

    /** 备注  */
    private String roleRemark;



    public static RoleDO create(){
        return new RoleDO();
    }

    public RoleDO build(){
        return this;
```

# 最后说两句

大家有什么问题,欢迎加微信一起讨论: best396975802