# Dao层：

利用mybatis逆向工程通过表结构自动生成出对应的PO类，POExample类(经常做按要求查询的时候用),POMapper(相当于是一个dao接口),PoMapper.xml(xml文件 只要它和POMapper接口在同一个包中就不需要在写dao的实现了 查看application.xml下org.mybatis.spring.mapper.MapperScannerConfigurer的配置包地址)

表名映射到类名的规则是t\_order\_out到PO类TOrderOut并且它里面的字段全是小写的（在做扩展的时候和前台datagrid的filed的时候注意）

总共生成四个文件2个JO文件1个接口1个xml文件 2个JO放在pojo.po包下

另两个放在dao.mapper下这些都是可以在generatorSqlMapCustom这个工程里的xml文件中配置指定生成的（已在那个文件中说明配置了）。

# 前台：

再做页面查询的时候，创建一个queryVo类到pojo.vo包中

比如：查询user 可以建一个UserQueryVo 然后包装一个UserCustom（user的扩展类）和一个PageQuery然后进行查询（详细见UsersAction里面的findAll方法 一直从action层看到dao的mapper.xml文件findAll返回一个DatagridResultInfo类里面的total是总数rows是全部的对象list

）

再做添加时，详见UsersAction的add方法它返回的是一个SubmitResultInfo(做提交的时候都用这个，查询用DatagridResultinfo)它包装了一个ResultInfo(三个字段type messageCode message 详细见ResultInfo)在service层中如果出现了前置条件不满足时(比如：用户名已存在). 直接抛一个自定义异常ExceptionResultInfo。（他里面也包装一个ResultInfo）。这是自定义异常解析器会自动处理(详见ExceptionResolverCustom)

com.zt.process.result包里面的有四个resultinfo类(基本resultinfo类,submitresultinfo,

datagridresultinfo,exceptionresultinfo)和一个工具类都需要自己细读，着UsersAction类的两个方法结合ResourceUtil类和resource文件下的messages.properties(下插件查看unicode字符

所有的提示全部在messages.properties中声明

## 开发规格：

生成的mapper和mapper.xml放在dao.mapper包中

生成的Po和PoExample有扩展的Po(命名为PoCustom)放在pojo.po还包中

扩展的Vo放入pojo.vo中命名为PoQueryVo

所有的提示信息全部在resources下的messages.properties中声明

Jsp下每个模块下一个文件夹分为查询添加修改各为一个jsp

# 注意：properties文件在eclipse是Unicode编码自行下载插件 help->eclipse marketplace->搜索properties 下载插件重启后右击properties文件OpenWith->SimplePropertiesEditor