

**Mybatis**

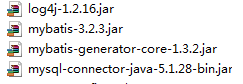
**逆向工程**

# Mybatis逆向工程

使用官方网站的mapper自动生成工具mybatis-generator-core-1.3.2来生成po类和mapper映射文件。

作用:mybatis官方提供逆向工程,可以使用它通过数据库中的表来自动生成Mapper接口和映射文件(单表增删改查)和Po类.

导入的jar包有:



## 第一步：mapper生成配置文件：

在generatorConfig.xml中配置mapper生成的详细信息，注意改下几点：

1. 添加要生成的数据库表
2. po文件所在包路径
3. mapper文件所在包路径

配置文件如下：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!DOCTYPE generatorConfiguration  PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"  "http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd">  <generatorConfiguration>  <context id=*"testTables"* targetRuntime=*"MyBatis3"*>  <commentGenerator>  <!-- 是否去除自动生成的注释 true：是 ： false:否 -->  <property name=*"suppressAllComments"* value=*"true"* />  </commentGenerator>  <!--数据库连接的信息：驱动类、连接地址、用户名、密码 -->  <jdbcConnection driverClass=*"com.mysql.jdbc.Driver"*  connectionURL=*"jdbc:mysql://localhost:3306/mybatis"* userId=*"root"*  password=*"admin"*>  </jdbcConnection>  <!-- <jdbcConnection driverClass="oracle.jdbc.OracleDriver"  connectionURL="jdbc:oracle:thin:@127.0.0.1:1521:yycg"  userId="yycg"  password="yycg">  </jdbcConnection> -->  <!-- 默认false，把JDBC DECIMAL 和 NUMERIC 类型解析为 Integer，为 true时把JDBC DECIMAL 和  NUMERIC 类型解析为java.math.BigDecimal -->  <javaTypeResolver>  <property name=*"forceBigDecimals"* value=*"false"* />  </javaTypeResolver>  <!-- targetProject:生成PO类的位置 -->  <javaModelGenerator targetPackage=*"cn.itcast.ssm.po"*  targetProject=*".\src"*>  <!-- enableSubPackages:是否让schema作为包的后缀 -->  <property name=*"enableSubPackages"* value=*"false"* />  <!-- 从数据库返回的值被清理前后的空格 -->  <property name=*"trimStrings"* value=*"true"* />  </javaModelGenerator>  <!-- targetProject:mapper映射文件生成的位置 -->  <sqlMapGenerator targetPackage=*"cn.itcast.ssm.mapper"*  targetProject=*".\src"*>  <!-- enableSubPackages:是否让schema作为包的后缀 -->  <property name=*"enableSubPackages"* value=*"false"* />  </sqlMapGenerator>  <!-- targetPackage：mapper接口生成的位置 -->  <javaClientGenerator type=*"XMLMAPPER"*  targetPackage=*"cn.itcast.ssm.mapper"*  targetProject=*".\src"*>  <!-- enableSubPackages:是否让schema作为包的后缀 -->  <property name=*"enableSubPackages"* value=*"false"* />  </javaClientGenerator>  <!-- 指定数据库表 -->  <table tableName=*"items"*></table>  <table tableName=*"orders"*></table>  <table tableName=*"orderdetail"*></table>  <table tableName=*"user"*></table>  <!-- <table schema="" tableName="sys\_user"></table>  <table schema="" tableName="sys\_role"></table>  <table schema="" tableName="sys\_permission"></table>  <table schema="" tableName="sys\_user\_role"></table>  <table schema="" tableName="sys\_role\_permission"></table> -->    <!-- 有些表的字段需要指定java类型  <table schema="" tableName="">  <columnOverride column="" javaType="" />  </table> -->  </context>  </generatorConfiguration> |

## 第二步：使用java类生成mapper文件：

**Public void** generator() **throws** Exception{

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

**boolean** overwrite = **true**;

File configFile = **new** File("generatorConfig.xml");

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**);

}

**Public static void** main(String[] args) **throws** Exception {

**try** {

GeneratorSqlmap generatorSqlmap = **new** GeneratorSqlmap();

generatorSqlmap.generator();

} **catch** (Exception e) {

e.printStackTrace();

}

}

## 第三步：拷贝生成的mapper文件到工程中指定的目录中

### Mapper.xml

Mapper.xml的文件拷贝至mapper目录内

### Mapper.java

Mapper.java的文件拷贝至mapper 目录内

注意：mapper xml文件和mapper.java文件在一个目录内且文件名相同。

## 第四步Mapper接口测试

学会使用mapper自动生成的增、删、改、查方法。

//删除符合条件的记录

**int** deleteByExample(UserExample example);

//根据主键删除

**int** deleteByPrimaryKey(String id);

//插入对象所有字段

**int** insert(User record);

//插入对象不为空的字段

**int** insertSelective(User record);

//自定义查询条件查询结果集

List<User> selectByExample(UserExample example);

//根据主键查询

UserselectByPrimaryKey(String id);

//根据主键将对象中不为空的值更新至数据库

**int** updateByPrimaryKeySelective(User record);

//根据主键将对象中所有字段的值更新至数据库

**int** updateByPrimaryKey(User record);

## 逆向工程注意事项

### Mapper文件内容不覆盖而是追加

XXXMapper.xml文件已经存在时，如果进行重新生成则mapper.xml文件内容不被覆盖而是进行内容追加，结果导致mybatis解析失败。

解决方法：删除原来已经生成的mapper xml文件再进行生成。

Mybatis自动生成的po及mapper.java文件不是内容而是直接覆盖没有此问题。

### Table schema问题

下边是关于针对oracle数据库表生成代码的schema问题：

Schma即数据库模式，oracle中一个用户对应一个schema，可以理解为用户就是schema。

当Oralce数据库存在多个schema可以访问相同的表名时，使用mybatis生成该表的mapper.xml将会出现mapper.xml内容重复的问题，结果导致mybatis解析错误。

解决方法：在table中填写schema，如下：

<table schema="XXXX" tableName=" " >

XXXX即为一个schema的名称，生成后将mapper.xml的schema前缀批量去掉，如果不去掉当oracle用户变更了sql语句将查询失败。

快捷操作方式：mapper.xml文件中批量替换：“from XXXX.”为空

Oracle查询对象的schema可从dba\_objects中查询，如下：

select \* from dba\_objects