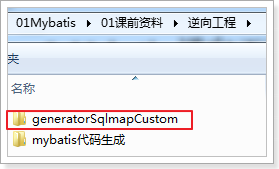
# Mybatis逆向工程

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

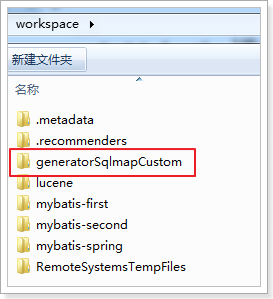
## 导入逆向工程

使用课前资料已有逆向工程，如下图：



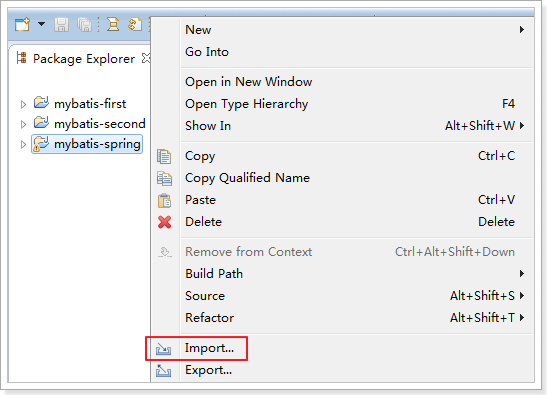
### 复制逆向工程到工作空间中

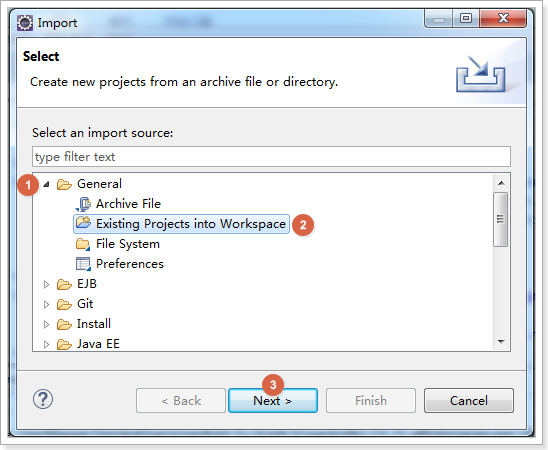
复制的效果如下图：

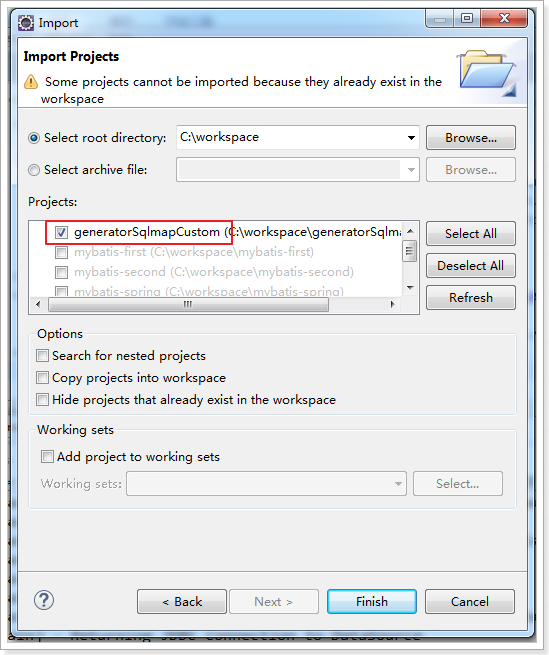


### 导入逆向工程到eclipse中

如下图方式进行导入：

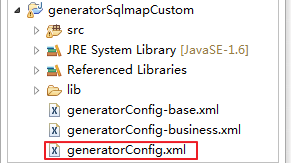






## 修改配置文件

在generatorConfig.xml中配置Mapper生成的详细信息，如下图：



注意修改以下几点:

1. 修改要生成的数据库表
2. pojo文件所在包路径
3. Mapper所在的包路径

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="root"*>

</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 schema=*""* tableName=*"user"*></table>

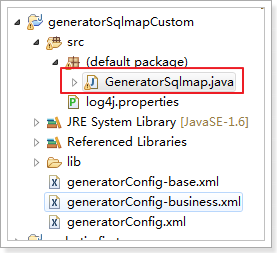
<table schema=*""* tableName=*"order"*></table>

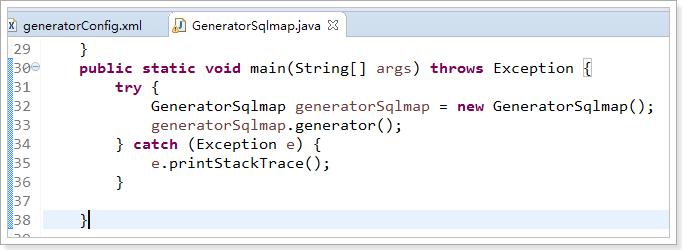
</context>

</generatorConfiguration>

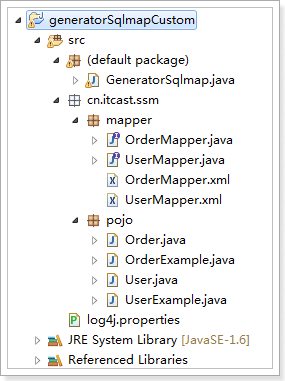
## 生成逆向工程代码

找到下图所示的java文件，执行工程main主函数,



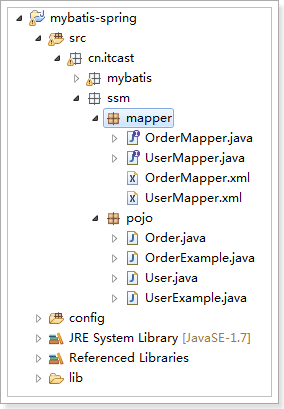


刷新工程，发现代码生成，如下图：



## 测试逆向工程代码

1. 复制生成的代码到mybatis-spring工程，如下图



2. 修改spring配置文件

在applicationContext.xml修改

<!-- Mapper代理的方式开发，扫描包方式配置代理 -->

<bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>

<!-- 配置Mapper接口，如果需要加载多个包，直接写进来，中间用，分隔 -->

<!-- <property name="basePackage" value="cn.itcast.mybatis.mapper" /> -->

<property name=*"basePackage"* value=*"cn.itcast.ssm.mapper"* />

</bean>

3. 编写测试方法：

**public** **class** UserMapperTest {

**private** ApplicationContext context;

@Before

**public** **void** setUp() **throws** Exception {

**this**.context = **new** ClassPathXmlApplicationContext("classpath:applicationContext.xml");

}

@Test

**public** **void** testInsert() {

// 获取Mapper

UserMapper userMapper = **this**.context.getBean(UserMapper.**class**);

User user = **new** User();

user.setUsername("曹操");

user.setSex("1");

user.setBirthday(**new** Date());

user.setAddress("三国");

userMapper.insert(user);

}

@Test

**public** **void** testSelectByExample() {

// 获取Mapper

UserMapper userMapper = **this**.context.getBean(UserMapper.**class**);

// 创建User对象扩展类，用户设置查询条件

UserExample example = **new** UserExample();

example.createCriteria().andUsernameLike("%张%");

// 查询数据

List<User> list = userMapper.selectByExample(example);

System.***out***.println(list.size());

}

@Test

**public** **void** testSelectByPrimaryKey() {

// 获取Mapper

UserMapper userMapper = **this**.context.getBean(UserMapper.**class**);

User user = userMapper.selectByPrimaryKey(1);

System.***out***.println(user);

}

}

注意：

1. 逆向工程生成的代码只能做单表查询
2. 不能在生成的代码上进行扩展，因为如果数据库变更，需要重新使用逆向工程生成代码，原来编写的代码就被覆盖了。
3. 一张表会生成4个文件