# 分析数据模型—自定义类型



## 思路

1. 明确每张表存储的信息
2. 明确每张表中关键字段（主键、外键、非空）
3. 明确数据库中表与表之间的外键关系
4. 明确业务中表与表的关系（建立在具体的业务）

## 图形分析

一对多

订单表

id：订单ID

Number：订单号

User\_id：用户id（外键）

订单明细表

Orders\_id：订单ID（外键）

Items\_id：商品ID（外键）

外键

一对一

外键

一对多

一对一

外键

一对一

一对多

多对多

多对多

商品表

Id：商品ID

用户表

Id：用户ID

## 外键关系

**user和orders：**

user---->orders：一个用户可以创建多个订单，一对多

orders--->user：一个订单只由一个用户创建，一对一

**orders和orderdetail：**

orders-🡪orderdetail：一个订单可以包括 多个订单明细，因为一个订单可以购买多个商品，每个商品的购买信息在orderdetail记录，一对多关系

orderdetail--> orders：一个订单明细只能包括在一个订单中，一对一

**orderdetail和itesm：**

orderdetail---》itesms：一个订单明细只对应一个商品信息，一对一

items--> orderdetail:一个商品可以包括在多个订单明细 ，一对多

## 没有外键关系

**Orders和items：**

这两张表没有直接的外键关系，通过业务及数据库的间接关系分析出它们是多对多的关系。

Orders🡪 orderdetail –>items：一个订单可以有多个订单明细，一个订单明细对应一个商品，所以一个订单对应多个商品

Items-🡪orderdetail🡪orders：一个商品可以对应多个订单明细，一个订单明细对应一个订单，所以一个商品对应多个订单

**User和items：**

这两张表没有直接的外键关系，通过业务及数据库的间接关系分析出它们是多对多的关系。

User🡪orders🡪orderdetail🡪items：一个用户有多个订单，一个订单有多个订单明细、一个订单明细对应一个商品，所以一个用户对应多个商品

Items🡪orderdetail🡪orders🡪user：一个商品对应多个订单明细，一个订单明细对应一个订单，一个订单对应一个用户，所以一个商品对应多个用户

# 一对一查询

## 需求

查询订单信息关联查询用户信息

## sql语句

查询语句：

先确定主查询表：订单信息表

再确定关联查询表：用户信息

通过orders关联查询用户使用user\_id一个外键，只能关联查询出一条用户记录就可以使用内连接

SELECT

orders.\*,

user.username,

user.sex

FROM

orders,

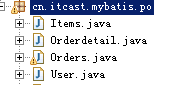
USER

WHERE orders.user\_id = user.id

## 使用resultType实现 (扩展类)

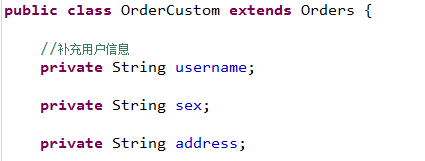
### 一对一查询映射的pojo

基础的单表的 po类：



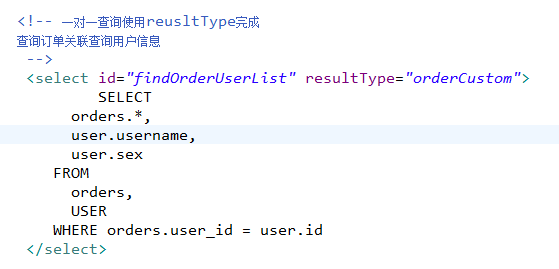
创建pojo包括 订单信息和用户信息，resultType才可以完成映射。

创建OrderCustom作为自定义pojo，继承sql查询列多的po类。

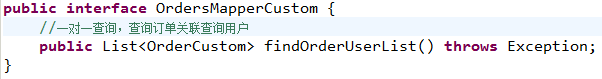


### mapper.xml

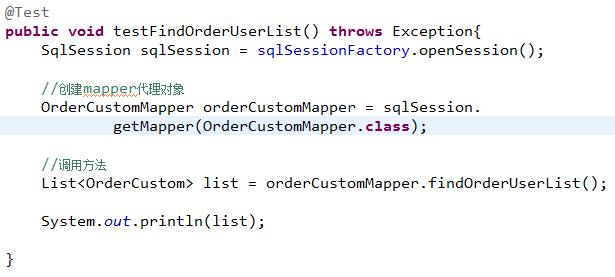
定义mapper.xml文件，



### mapper.java



### 测试

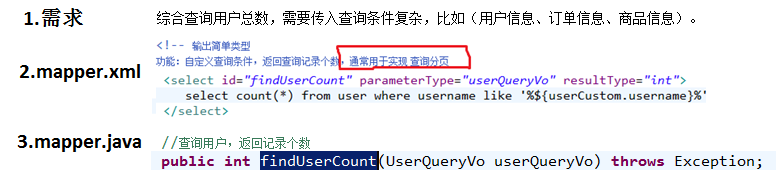


# resultType—输出

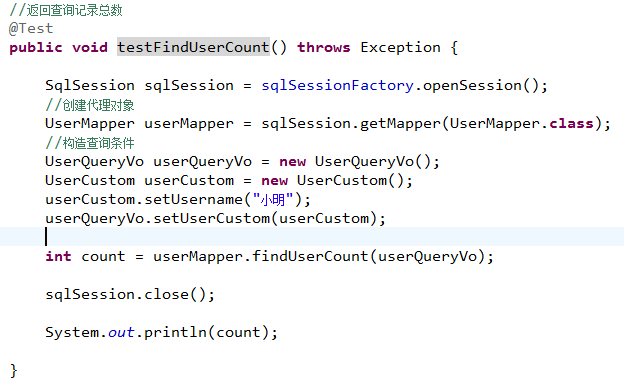
指定输出结果的类型（pojo、简单类型、hashmap..），将sql查询结果映射为java对象 。

## 返回简单类型

如果查询记录结果集为一条记录且一列，使用ResultType返回简单类型。



测试代码：



## 返回pojo对象和list<pojo>

注意：输出单个pojo对象和pojo列表（盛放pojo对象）时，mapper映射文件中的resultType的类型是一样的，mapper接口的方法返回值不同。



总结：同样的mapper映射文件，返回单个对象和对象列表时，mapper接口在生成动态代理的时候，**会根据返回值的类型，决定调用selectOne方法还是selectList方法。**

## 返回自定义pojo对象和list<pojo>（扩展）

两种方式：1.继承pojo并且新增属性

2.自定义一个pojo类



## 返回map或者list<map>

# 高级映射总结

## resultType：

作用：

将查询结果按照sql列名pojo属性名一致性映射到pojo中。

场合：

常见一些明细记录的展示，比如用户购买商品明细，将关联查询信息全部展示在页面时，此时可直接使用resultType将每一条记录映射到pojo中，在前端页面遍历list（list中是pojo）即可。

## resultMap：

使用association和collection完成一对一和一对多高级映射（对结果有特殊的映射要求）。

### association：

作用：

将关联查询信息映射到一个pojo对象中。

场合：

为了方便查询关联信息可以使用association将关联订单信息映射为用户对象的pojo属性中，比如：查询订单及关联用户信息。

使用resultType无法将查询结果映射到pojo对象的pojo属性中，根据对结果集查询遍历的需要选择使用resultType还是resultMap。

### collection：

作用：

将关联查询信息映射到一个list集合中。

场合：

为了方便查询遍历关联信息可以使用collection将关联信息映射到list集合中，比如：查询用户权限范围模块及模块下的菜单，可使用collection将模块映射到模块list中，将菜单列表映射到模块对象的菜单list属性中，这样的作的目的也是方便对查询结果集进行遍历查询。

如果使用resultType无法将查询结果映射到list集合中。

## 对比

**resultType**：

1.使用resultType实现较为简单，

3.如果如果没有查询结果的特殊要求；pojo中没有包括查询出来的列名，需要增加列名对应的属性，即可完成映射。

**3.resultType无法实现延迟加载。（只有一个实体类，想也没法呀）**

**resultMap**：

1.需要单独定义resultMap，实现有点麻烦，

2.如果对查询结果有特殊的要求，使用resultMap可以完成将关联查询映射pojo的对象属性中。

**3.resultMap可以实现延迟加载，**