mybatis第二部分 高级映射

复习：

1. SqlMapConfig.xml 文件的配置 ：
2. 什么 是 类型别名， typeAlias ？ 有 什么 用处 ？
3. 主要用的 方式 。A。自定义， package
4. Mappers。 Mapper， 定义讲 了 三种 方式 。

： resources, class, package name = “”

1. Mybatis 的注解开发 ， mappers 还是需要定义。 (resultmap)
2. javaBean的扩展。扩展 以前的javaBean，paramterType .
3. Select \* from user where ... limit ?, ?; 分页
4. resultMap ：可以 将输出的类型 转换 想要的类型。影射， map。 列名： 对象的属性。初级
5. 动态SQL：，

if test where ？ foreach ？（in） 可以处理， 列表属性 。

1. resultMap, 一对一的实现：三种方式

Resulttype。 Javabean 的扩展

resultMap： 关键词， assocation， 一对一 。 javaType: 对象， 什么对象

javaBean，多写一个关联表对象的属性 。

1. 一对 多。关键词 collection， 一对 多 。 对应的属性。 list对象 。 ofType:list 里面的对象
2. <id : 定义的主键，
3. <result： 定义是 普通 字段
4. <property >: javaBean 里面的成员变量
5. Column: 执行 sql 语句后 形成的 column 字段（ 如果 有 字段别名，就是 使用 别名）
6. 其他

# 课程计划

1. 对上次商品订单模型的数据库进行分析
2. 学习 resultmap 的高级映射知识 (重点)

一对一、一对多、多对多、延迟加载

# 商品订单数据模型

## 首先了解 表的名称和内容（要了解 表和业务的对应关系）

### user 表

表设计的惯例 。

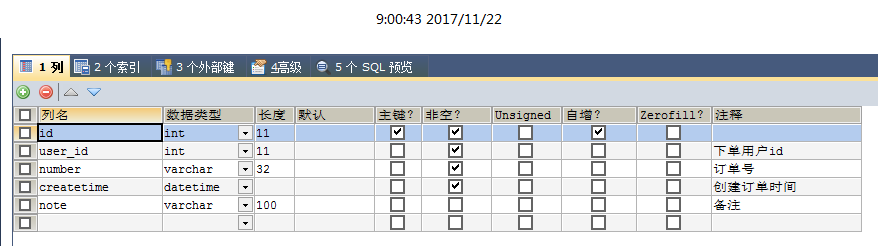
UUID（）

用户表，客户表 。



### 是 orders表

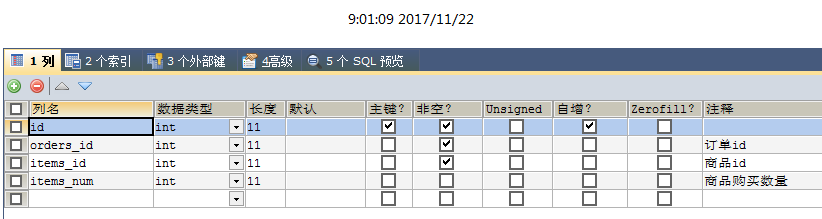
订单 。



### orderdetail

订单详情、

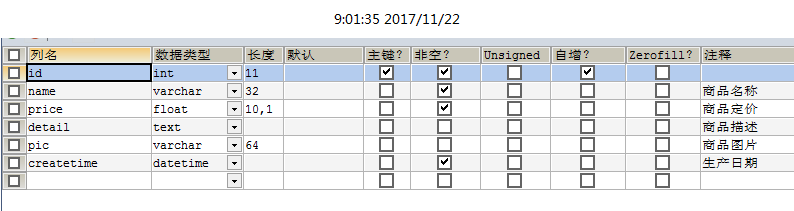
订单详情 。



### items

商品表。

商品表 。



## 了解 表里面的重要字段。（主键、外键、非空字段）

## 看看表与表之间的关系 （业务是核心）

orders 订单表

user， 用户表

orderdetail 订单详情表

Items 商品表表

orders\_id

Items\_id

use\_id

一对一

一对多

一对一

一对多

一对一

一对多

一对多

一对一

多对多

订单表 order

用户表（user

订单详情表

ordersdetail

商品表（items）

user\_id

order\_id

Item\_id

## 再分析 具体的关系。 一对一、一对多、多对多。 （）

用户表 订单详情表的对应关系：

用户表 对 orderdetail ： 一对多

orderdetail 对 user： 一对一

用户表 对 items： 一对多

items表 对 user：一对多

用户表 和 items 是 多对多 关系 。

orders 对 items： 一对多

items 对 orders： 一对 多 。

orders 对 items 也是 多对 多 。

个订单明细 一对多

# 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,

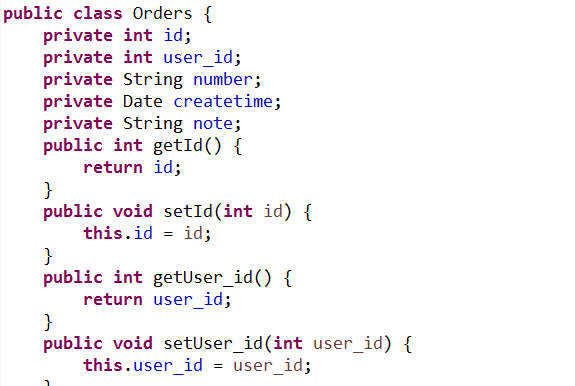
USER

WHERE orders.user\_id = user.id

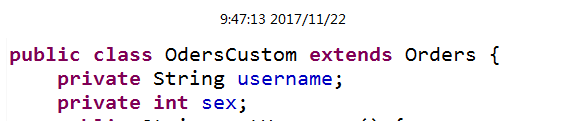
## 使用resultType实现

### 创建po类

基础的单表的 po类：



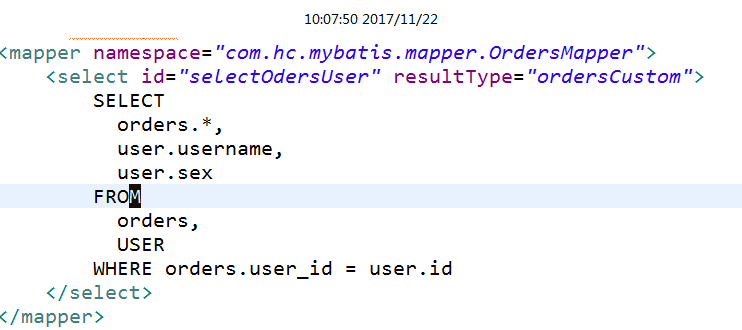
### 一对一查询映射的pojo



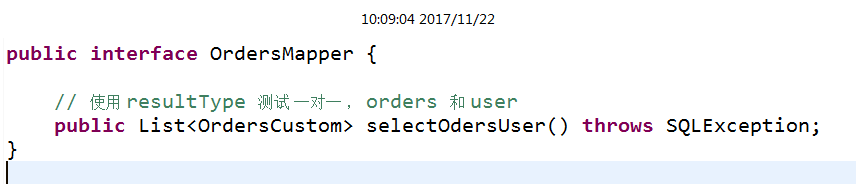
### mapper.xml

定义mapper.xml文件，

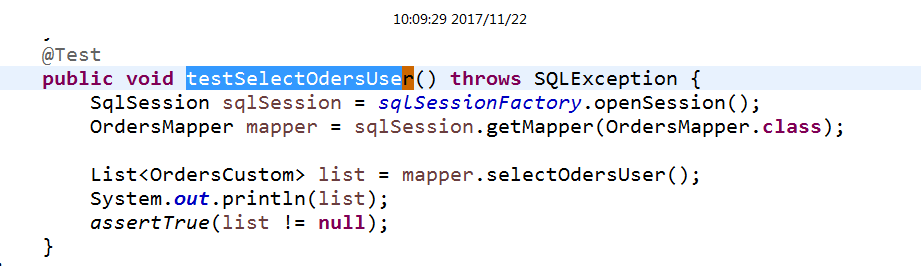
使用 一对一 查询 resulttype 。



### mapper.java



### 测试代码



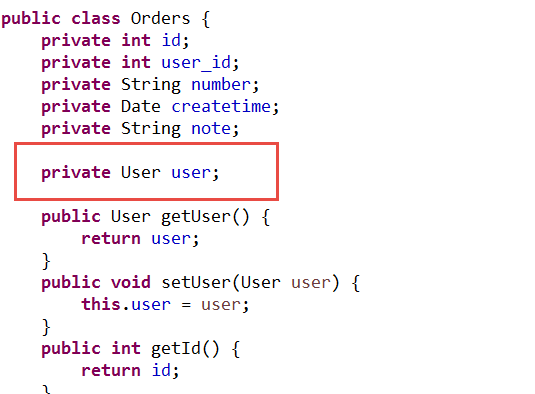
## 使用resultMap实现一对一

### resultMap映射思路

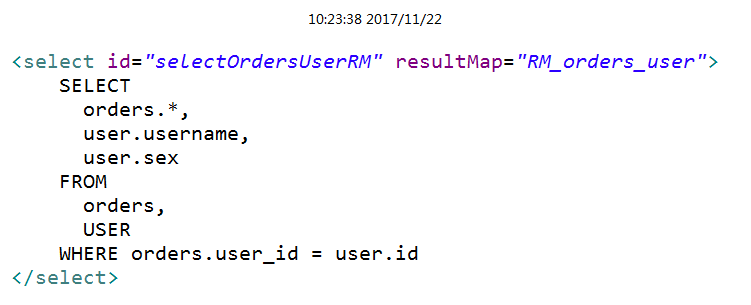
resultMap提供一对一关联查询的映射和一对多关联查询映射，一对一映射思路：将关联查询的信息映射到pojo中，如下：

在Orders类中创建一个User属性，将关联查询的信息映射到User属性中。

### Pojo 对象的定义

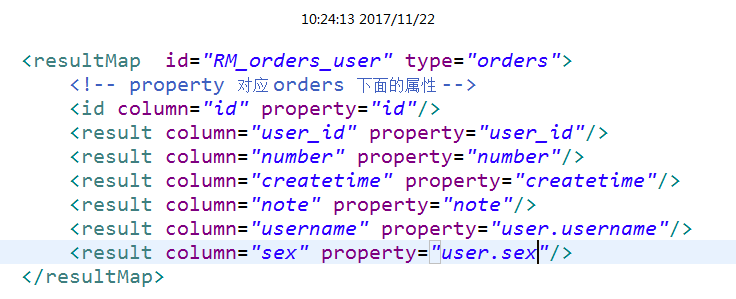


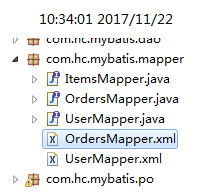
### mapper.xml



### resultMap定义

#### 不使用 associate 关键字



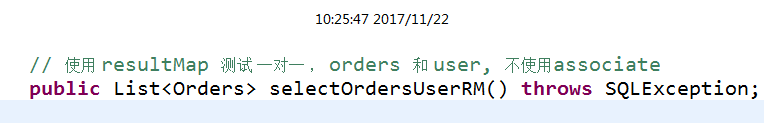


#### 使用 associate 关键字

associate 关联， 解决 一对一 的关联问题：



### mapper.java



测试 代码 省略

## 小结

resultType：要自定义pojo 保证sql查询列和pojo的属性对应，这种方法相对较简单，所以应用广泛。

resultMap：使用association完成一对一映射需要配置一个resultMap，过程有点复杂，

如果要实现延迟加载就只能用resultMap实现 ，如果为了方便对关联信息进行解析，也可以用association将关联信息映射到pojo中方便解析。

# 一对多查询

## 需求

查询所有订单信息及订单下的订单明细信息及用户信息。

## sql语句

主查询表：订单表

关联查询表：订单明细，用户表

SELECT

orders.\*,

user.username,

user.sex ,

orderdetail.id orderdatail\_id,

orderdetail.items\_num,

orderdetail.items\_id

FROM

orders,

user,

orderdetail

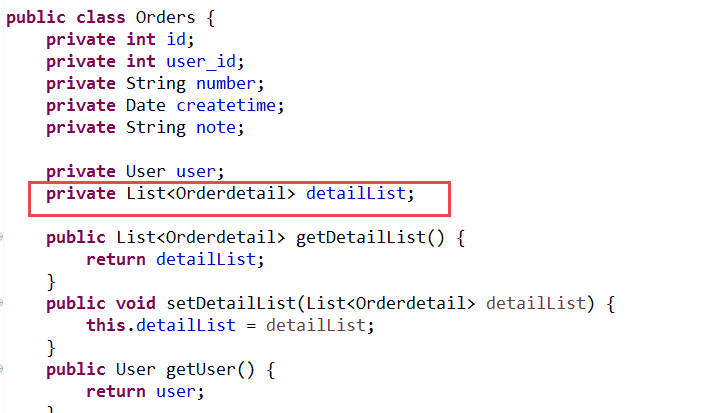
WHERE orders.user\_id = user.id AND orders.id = orderdetail.orders\_id

## resultMap进行一对多映射思路

resultMap 提供collection完成关联信息映射到集合对象中。



在orders类中创建集合属性：



## mapper.xml

SELECT

orders.\*,

user.username,

user.sex ,

orderdetail.id orderdetail\_id,

orderdetail.items\_num,

orderdetail.items\_id

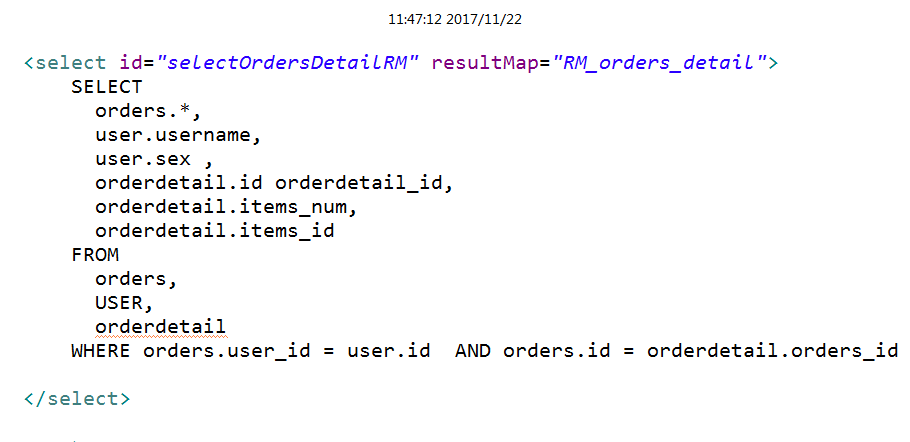
FROM

orders,

USER,

orderdetail

WHERE orders.user\_id = user.id AND orders.id = orderdetail.orders\_id



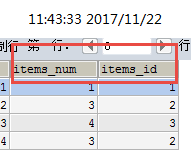
## resultMap定义

这里 注意 使用了 extends 关键字， 同 java 对象 扩展的意义基本一样 。

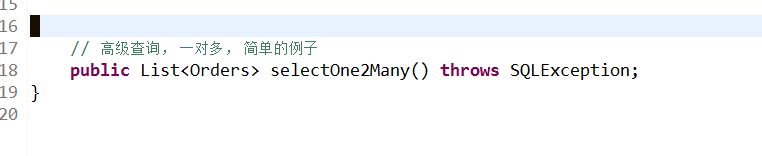
注意 collection 的定义： property为 Orders 对象 里面的属性

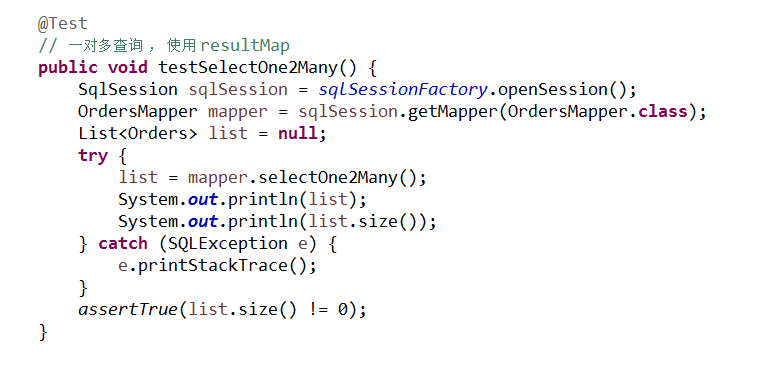
ofType 为 List<T> 的类型， 就是 在 Orders 里面 进行了定义

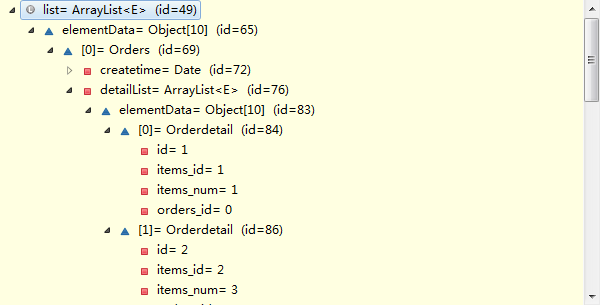




## mapper.java







# 一对多查询(复杂) 作业 ！

## 需求

查询所有用户信息，关联查询订单及订单明细信息及商品信息，订单明细信息中关联查询商品信息

用户 –》 orders 1 对 多

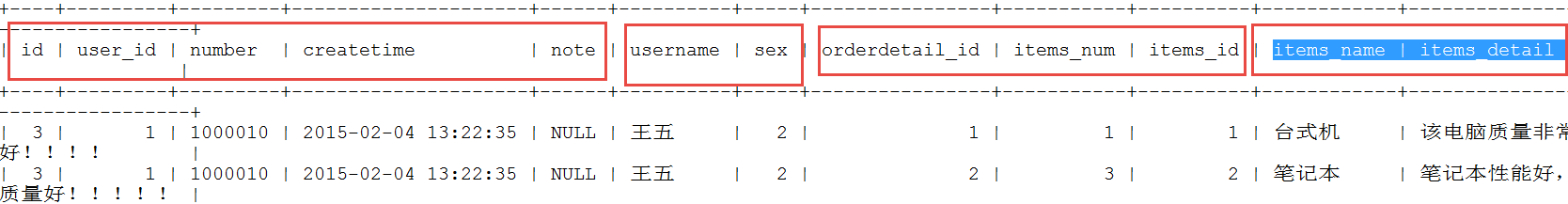
orders -》 orderdetail 1 对 多

orderdetail -》 items 一 对 一

## sql

主查询表：用户信息

关联查询：订单、订单明细，商品信息



SELECT

orders.\*,

user.username,

user.sex ,

orderdetail.id orderdetail\_id,

orderdetail.items\_num,

orderdetail.items\_id,

items.name items\_name,

items.detail items\_detail

FROM

orders,

user,

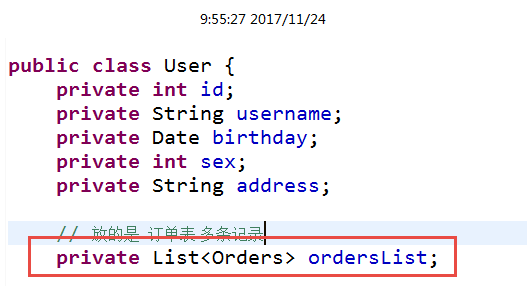
orderdetail,

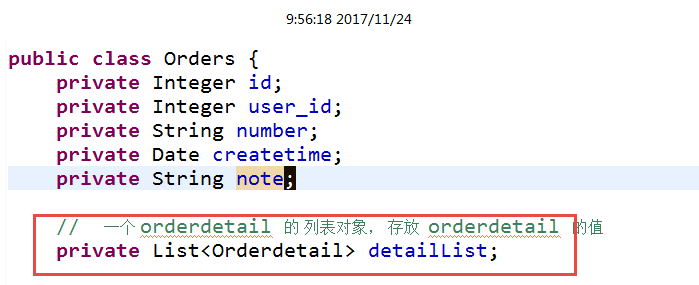
items

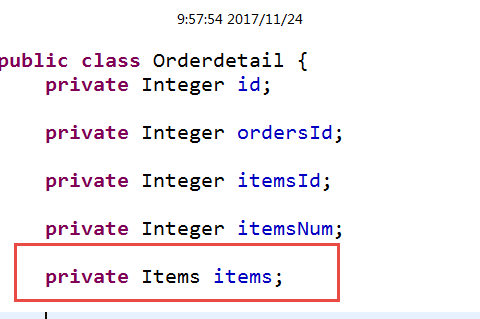
WHERE orders.user\_id = user.id AND orders.id = orderdetail.orders\_id AND items.id = orderdetail.items\_id

## pojo定义

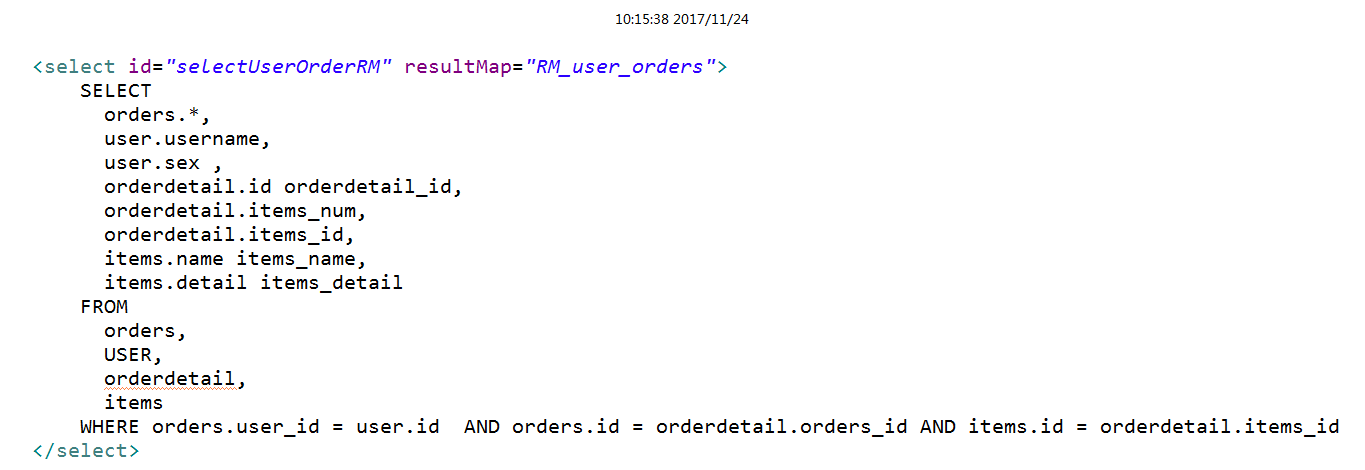
user.java orders.java orderdetail.java



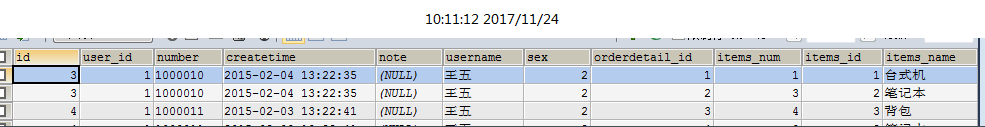




## mapper.xml

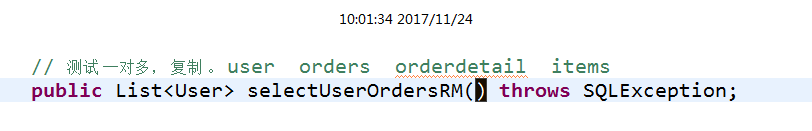


## resultMap

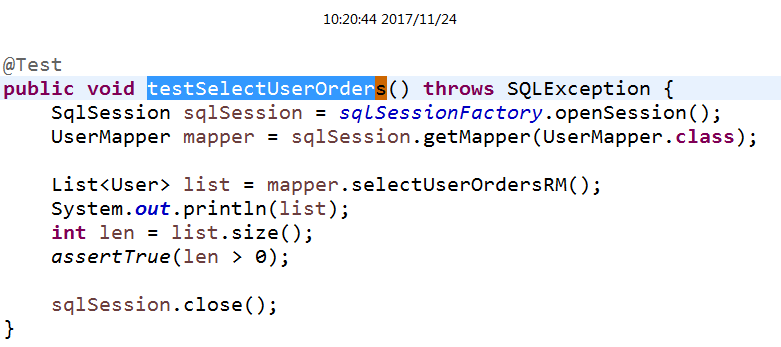




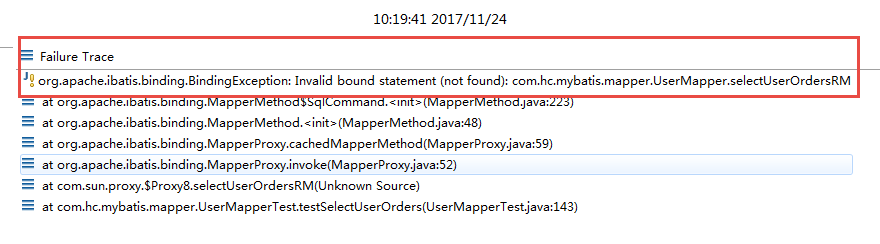
## mapper.java

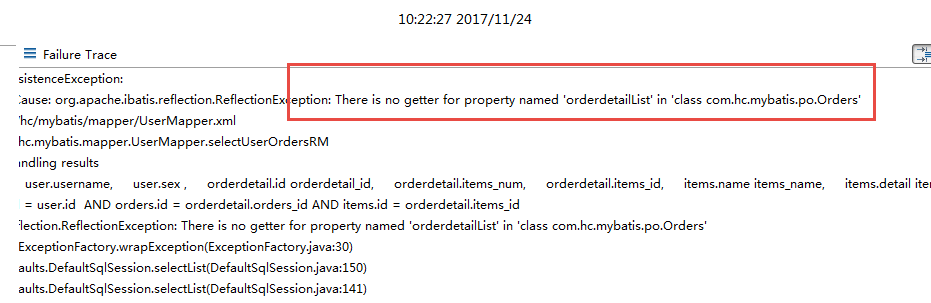


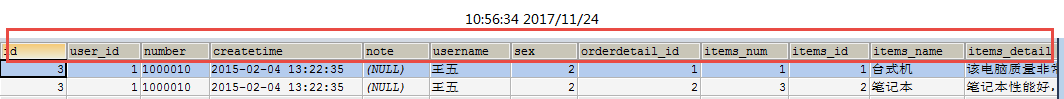
测试方法：



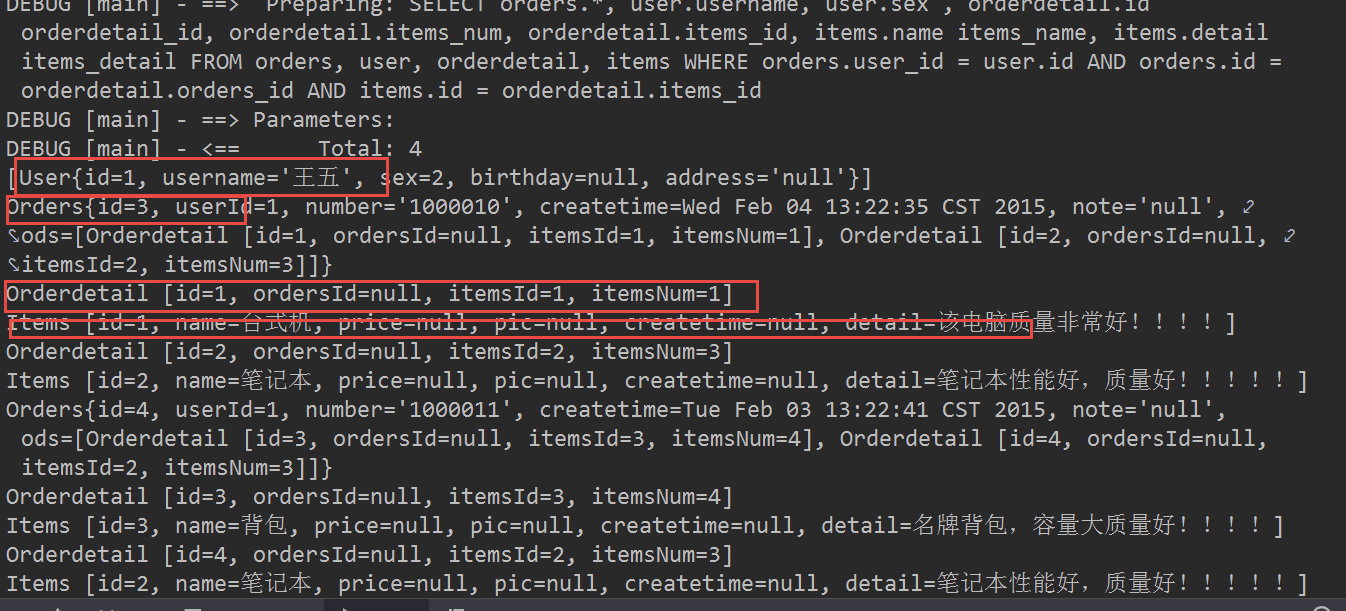
## 错误

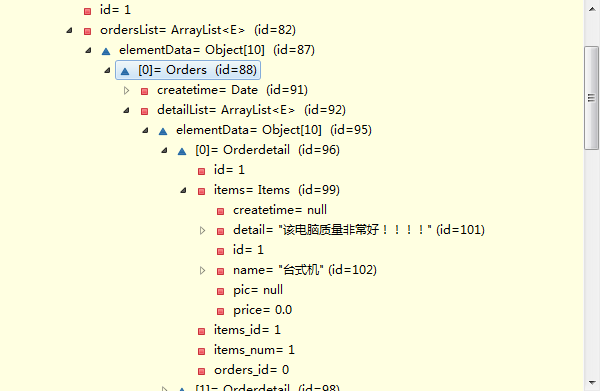












# 多对多查询 （ 一对多）

1. 》B （ 一对多）
2. 》A （ 一对 多）

多对 多

一对多是多对多的特例。

多对多的实质：

是 一对 多 和

多 对 一 。 一对 多 。

User， Items。

Public class User {

...

List<Items> itemsList;

}

Public class Items {

...

List<User> userList;

}

需求1：（resulttype）

查询显示字段：用户账号、用户名称、用户性别、商品名称、商品价格(最常见)

企业开发中常见明细列表，用户购买商品明细列表，

使用resultType将上边查询列映射到pojo输出。

需求2：( 一对 多）

查询显示字段：用户账号、用户名称、购买商品数量、商品明细

使用resultMap将用户购买的商品明细列表映射到user对象中。

# 延迟加载

## 使用延迟加载意义

使用单表查询 提高效率，当我们 需要进一步的数据才进行 进一步的查询 。

## 配置mybatis支持延迟加载

|  |  |  |  |
| --- | --- | --- | --- |
| 设置项 | 描述 | 允许值 | 默认值 |
| lazyLoadingEnabled | 全局性设置懒加载。如果设为‘false’，则所有相关联的都会被初始化加载。 | true | false | false |
| aggressiveLazyLoading | 当设置为‘true’的时候，懒加载的对象可能被任何懒属性全部加载。否则，每个属性都**按需加载**。 | true | false | true |
|  |  |  |  |

aggressive 侵略 。

**aggressiveLazyLoading是做什么么的？**  
它是控制具有懒加载特性的对象的属性的加载情况的。  
true表示如果对具有懒加载特性的对象的任意调用会导致这个对象的完整加载，false表示每种属性按照需要加载。

<!-- 全局配置参数 -->

<settings>

<!-- 延迟加载总开关 -->

<setting name=*"lazyLoadingEnabled"* value=*"true"* />

<!-- 设置按需加载 -->

<setting name=*"aggressiveLazyLoading"* value=*"false"* />

</settings>

## 延迟加载实现

### 实现思路

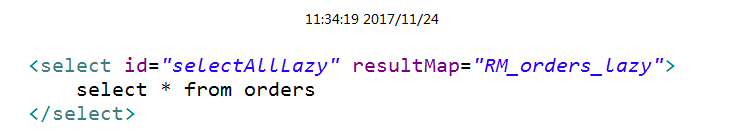
需求：

查询订单及用户的信息，一对一查询。

刚开始只查询订单信息

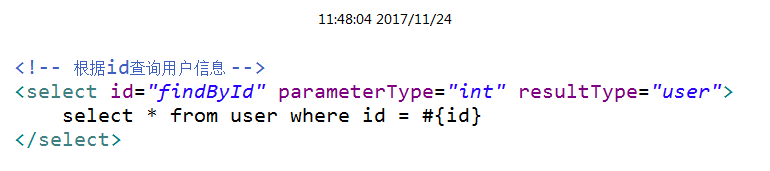
当需要用户时调用 Orders类中的getUser()方法执行延迟加载 ，向数据库发出sql。

### mapper.xml



### UserMapper.xml





### resultMap

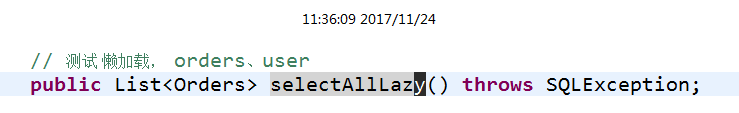
使用了 association 或 collection 。

关键字： property: 是 orders 的属性， select： 对应的 是 xml 中的 sql 语句。Mapper xml文件的 Namesapce.statement\_id，

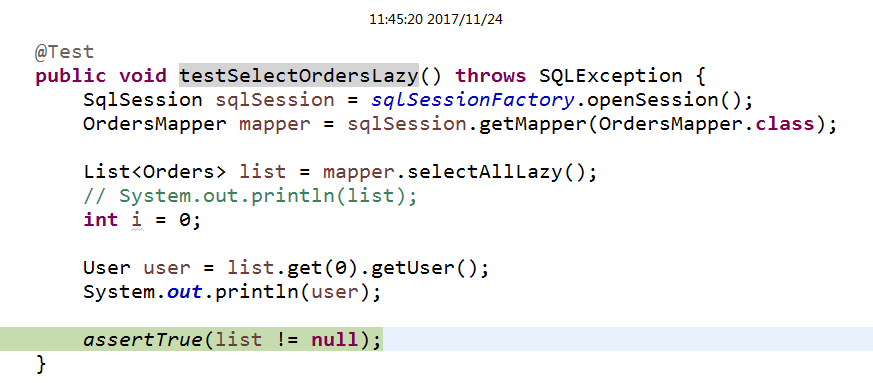
Column： 定义了 sql 语句的的输入参数 ， 与 statement\_id 里面进行对应 。



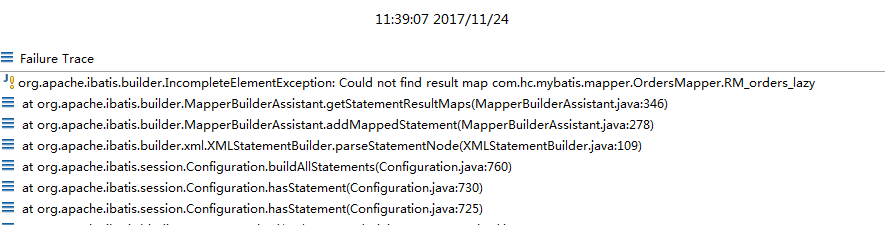
### mapper.java



### 测试代码



### 错误



## 一对多延迟加载（作业）

一对多延迟加载的方法同一对一延迟加载，在collection标签中配置select内容。

本部分内容自学， 实训时间 进行 练习 。

首先加载 orders 表里面的数据。 然后延迟加载 orderdetail 里面的数据 。

# resultType、resultMap、延迟加载使用场景总结

延迟加载：

延迟加载实现的方法多种多样，在只查询单表就可以满足需求，为了提高数据库查询性能使用延迟加载，再查询关联信息。

mybatis提供延迟加载的功能用于service层。

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，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。

association：

作用：

将关联查询信息映射到一个pojo类中。

场合：

为了**方便获取关联**信息可以使用association将关联订单映射为pojo，比如：查询订单及关联用户信息。

collection：

作用：

将关联查询信息映射到一个list集合中。

场合：

为了**方便获取关联信息**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和功能，可使用collection将模块和功能列表映射到list中。基于对象的数据展现