# 封装dao传统模式

## 搭建环境【步骤略】

## 封装dao，要提供一个dao的实现类，实现类里面有个私有全局属性SqlSessionFactory

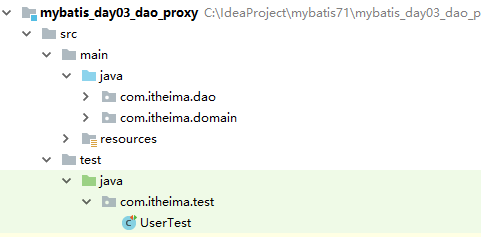


## 封装service，暂时先拿测试类模拟service，service中需要一个私有全局对象属性UserDao

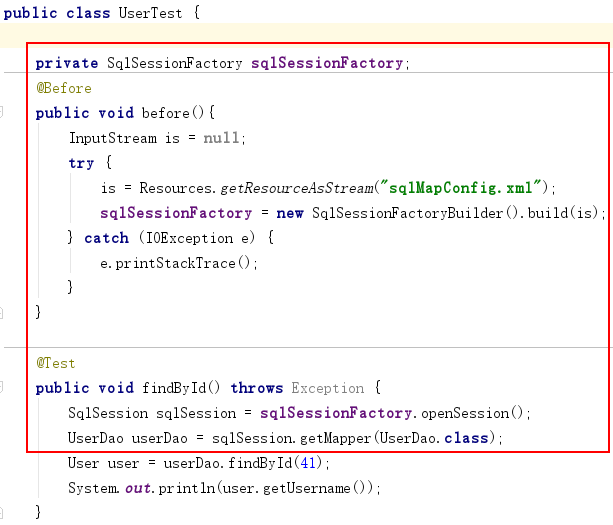


# 封装dao代理模式

## 复制上面项目改名字并删除dao的实现类



## 直接在service中去获取dao的代理对象

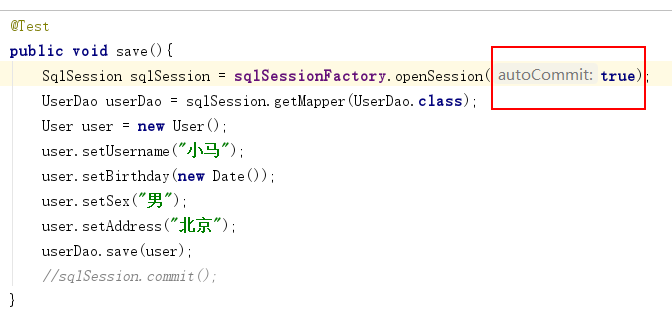


# Mybatis是什么时候从数据库连接池中拿数据库链接的？

Mybatis是在真正开始执行数据库操作的时候才去拿数据库链接。

# Mybatis默认的事务提交方式是什么，可以修改吗？

Mybatis默认事务的不提交的，可以修改。

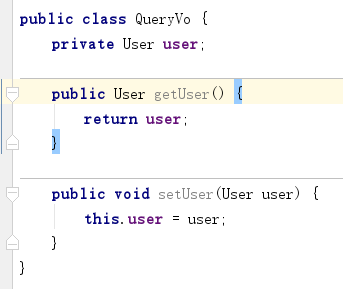


# 动态sql之if判断【动态sql都直接使用反射dao的环境】

## 需求

根据用户id和用户名做查询，但是不确定id和用户名都输入还是只输入一个。

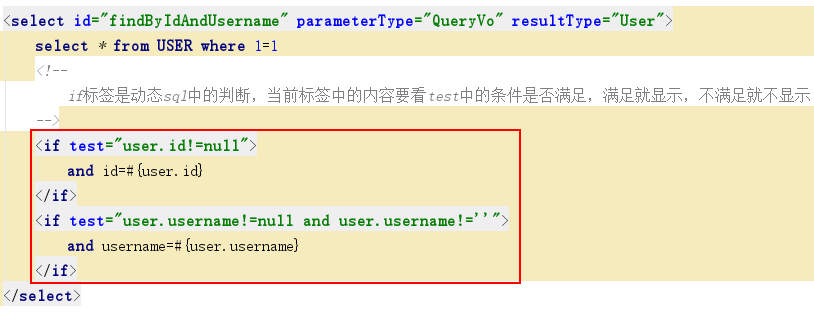
## 封装一个包装类作为查询条件



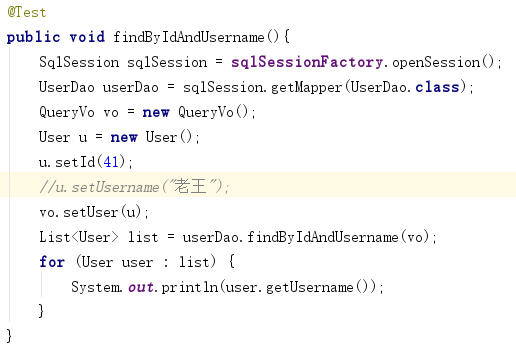
## 接口



## 映射

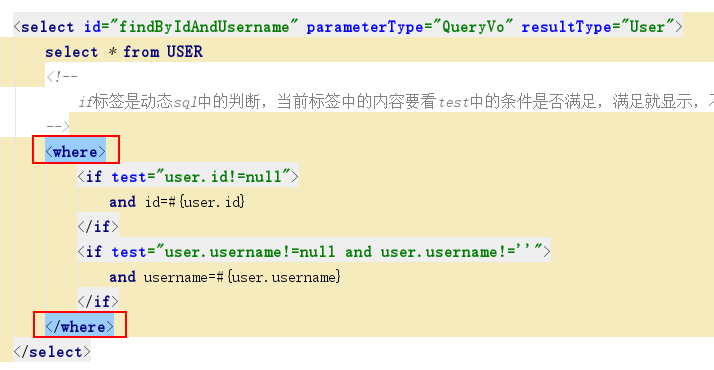


## 测试



# 动态sql之where

<where>标签就等同于在sql语句中拼接了一个where条件，而且如果后面紧挨着的条件中有and，会自动去掉。

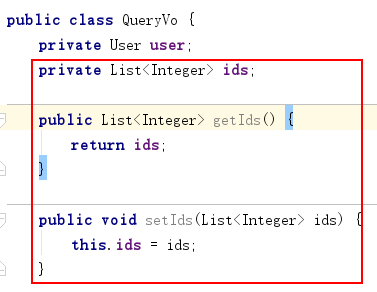


# 动态sql之foreach

## 需求

根据id的集合做查询

## 在包装类中添加是一个id的集合属性



## 接口

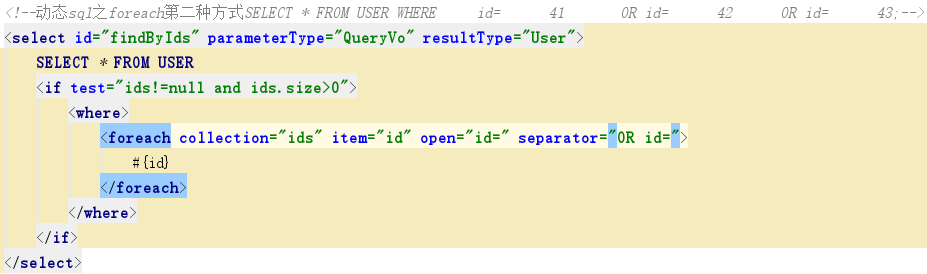


## 映射

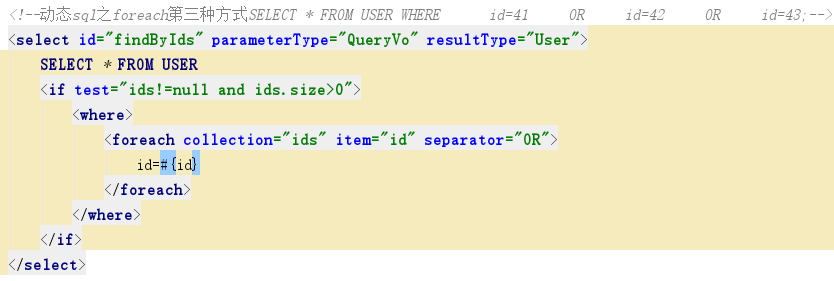
### 方式一



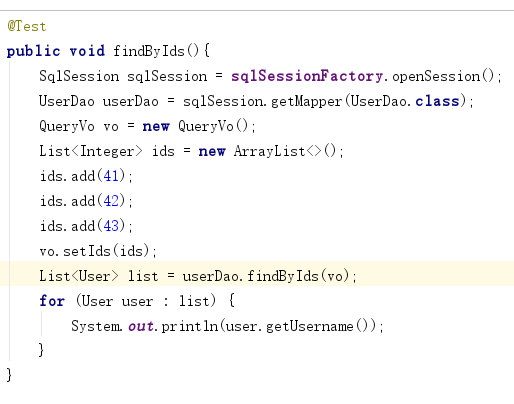
### 方式二



### 方式三

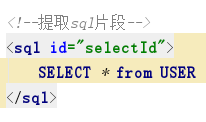


## 测试

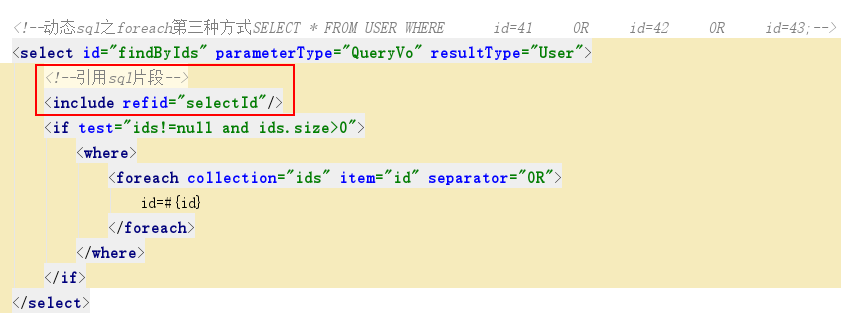


# Sql片段的使用

## 提取sql片段



## 引用sql片段



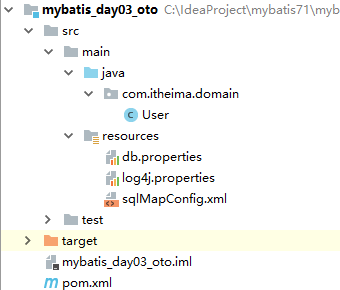
# Mybatis框架中的表关系说明

通常在关系型数据库中，表关系分一对一，一对多，多对多。

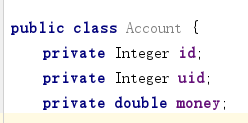
但是在mybatis框架中看表关系是站在一张表的角度去看另一张的。换句话说就mybatis中的表关系具有方向性。拿其中一张表的一条数据向另外一张表去看，如果这一条记录只能对应一条记录，就是一对一，如果可以对应多条记录，就是一对多。也就是说mybatis根本没有绝对多对多，而数据库中的多对多，在mybatis中其实是两个一对多。

# 一对一关联查询

## 搭建环境【略】



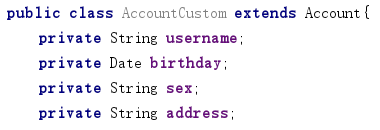
## 创建一个Account对象



## 需求：查询所有账户和当前账户所属用户信息

## 一对一方式一【使用resultType】

### 提供一个包装类来和数据库结果集中的列名能对上

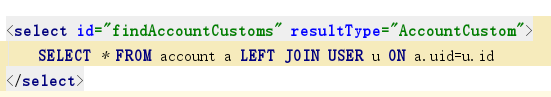


### 注意：在resultSet中有两列名称一模一样，最终以第一列为准，后面被覆盖。

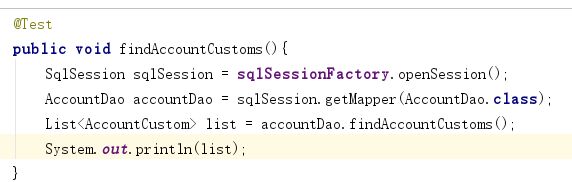
### 提供接口



### 提供映射



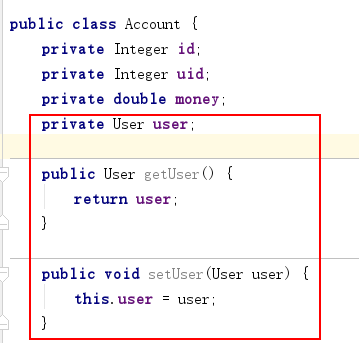
### 测试



## 一对一方式二【使用resultMap】

### 注意：在数据库端如果要体系出一对一或者一对多靠的是主外键。那么在java端如果要体现出A到B是一对一，要在A对象中加一个B的对象属性。

### 在java端体现出一对一关系



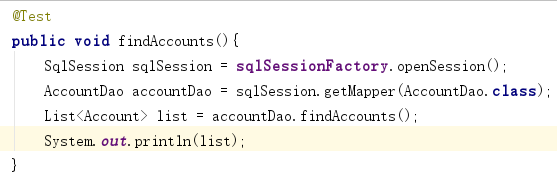
### 接口



### 映射

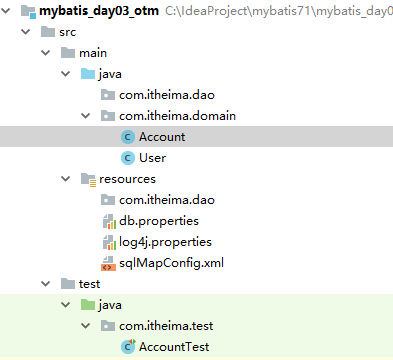


### 测试

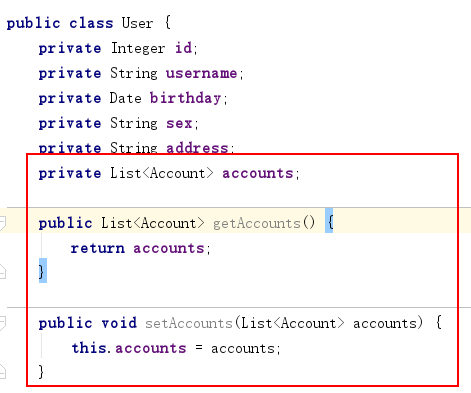


# 一对多关联查询

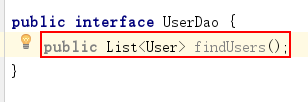
## 搭建环境【复制一对一环境做修改】



## 在java端体现出一对多关联关系



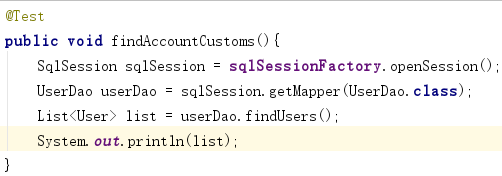
## 接口



## 映射



## 测试



# 多对多关联查询

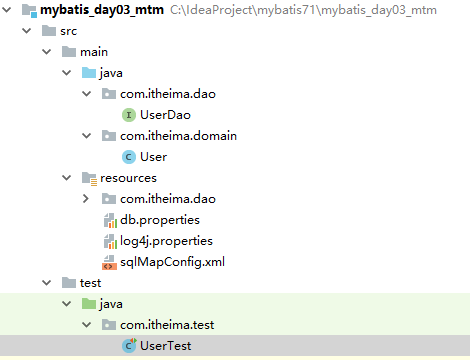
## 多对多在数据库中如何体系？为什么要这么体现？

多对多要靠中间表来体现关系，因为数据库端没有一个对象可以存储集合。

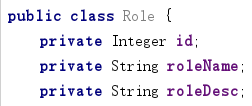
## 多对多关系在java端如何体现？

两个对象如果是多对多关系，那么就在彼此对象中放置对方的集合属性。

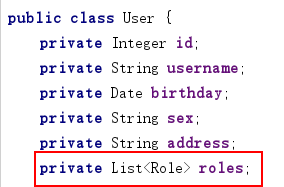
## 准备环境【复制上面环境改名字】

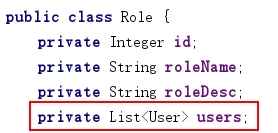


## 创建一个Role对象

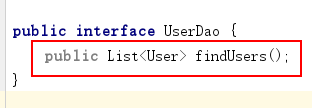


## 在java端体现出多对多关系

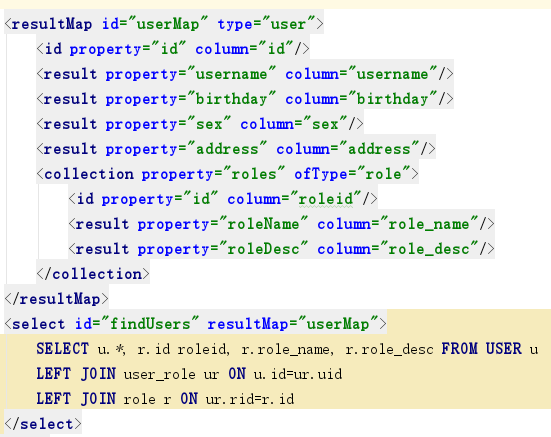




## 接口



## 映射



## 测试

