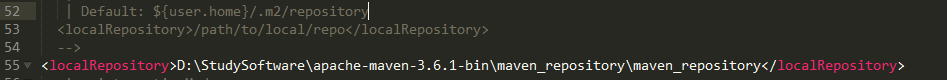
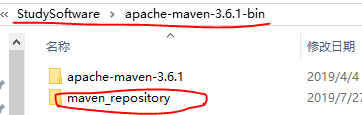
Mybatis遇到问题的解决方法

1. 本地仓库的搭建：

在maven->conf->setting.xml中配置本地仓库的位置：



本地仓库位置：



这样在写代码的时候就不会去中央仓库下载导致工程进度慢的问题，只有当本地仓库没有jar包才会去中央仓库下载。

1. 使用maven中自带tomcat运行工程导致servlet报错状态为500

原因：maven默认运行tomcat的版本为6，我本机安装的jdk为1.8导致不兼容。



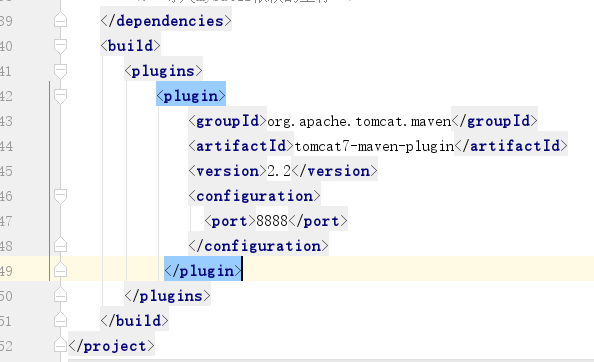
查看本机jdk版本：



解决方法：

方法一：降低jdk版本为1.6，方法比较麻烦，很可能由于jdk版本太低导致以后的各种问题（不建议降低jdk版本）

方法二：在pom.xml配置文件中建立tomcat较高版本的插件（这里添加tomcat插件为版本7）



三、

Mybatis主配置文件约束：

*<?***xml version="1.0" encoding="UTF-8"** *?>***<!DOCTYPE configuration  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd"*>***

映射配置文件约束：

*<?***xml version="1.0" encoding="UTF-8"** *?>***<!DOCTYPE mapper  
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>***

四、使用mybatis向数据库中插入数据导致中文乱码问题（前提数据库连接条件写在XXX.xml文件中）

解决方法：修改mybatis主配置文件中url

<configuration>  
 <!--配置环境-->  
 <environments default="mysql">  
 <!--配置mysql的环境-->  
 <environment id="mysql">  
 <!--配置事务的类型-->  
 <transactionManager type="JDBC"></transactionManager>  
 <!--配置数据源（连接池）-->  
 <dataSource type="POOLED">  
 <property name="driver" value="com.mysql.jdbc.Driver"></property>  
 <property name="url" value="jdbc:mysql://localhost:3306/login?useUnicode=true&amp;characterEncoding=UTF-8"></property>

<property name="username" value="root"></property>  
 <property name="password" value="123456"></property>  
 </dataSource>  
 </environment>  
 </environments>

红色部分是将字符&转义，因为mybatis主配置XXX.xml无法识别&字符，且末尾没有分号。

1. log4j.properties配置文件

*# Set root category priority to INFO and its only appender to CONSOLE.  
#log4j.rootCategory=INFO, CONSOLE debug info warn error fatal***log4j.rootCategory**=**debug, CONSOLE, LOGFILE***# Set the enterprise logger category to FATAL and its only appender to CONSOLE.***log4j.logger.org.apache.axis.enterprise**=**FATAL, CONSOLE***# CONSOLE is set to be a ConsoleAppender using a PatternLayout.***log4j.appender.CONSOLE**=**org.apache.log4j.ConsoleAppender  
log4j.appender.CONSOLE.layout**=**org.apache.log4j.PatternLayout  
log4j.appender.CONSOLE.layout.ConversionPattern**=**%d{ISO8601} %-6r [%15.15t] %-5p %30.30c %x - %m\n***# LOGFILE is set to be a File appender using a PatternLayout.***log4j.appender.LOGFILE**=**org.apache.log4j.FileAppender  
log4j.appender.LOGFILE.File**=**d:**\a**xis.log  
log4j.appender.LOGFILE.Append**=**true  
log4j.appender.LOGFILE.layout**=**org.apache.log4j.PatternLayout  
log4j.appender.LOGFILE.layout.ConversionPattern**=**%d{ISO8601} %-6r [%15.15t] %-5p %30.30c %x - %m\n**

1. 获取插入数据的id

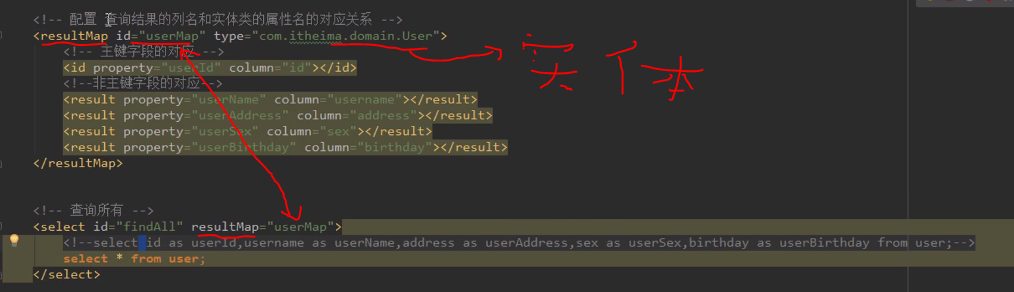


1. 实体属性与表中列明不一致解决方法

方法一：在查询语句中起别名（执行效率高，开发效率低）

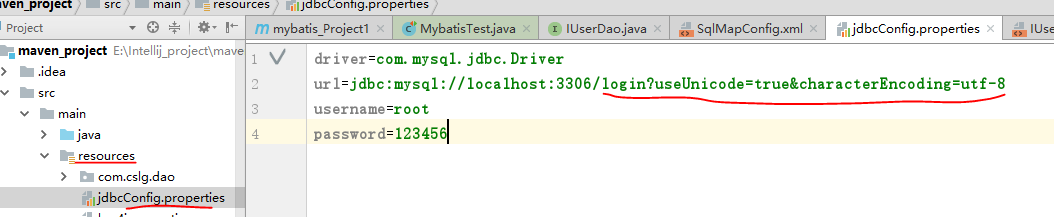
方法二：重新创建实体，或者重新建表

方法三：在配置文件中重新建立实体属性与表列名之间的对应关系

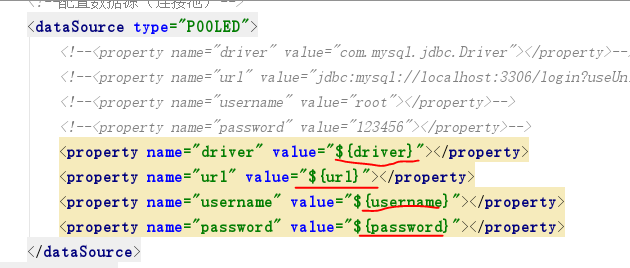
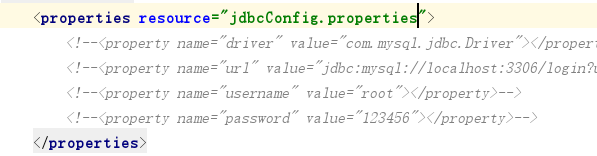


1. 数据库连接信息写在XXX.properties文件中，导致胡乱码解决方法

Xxx.properties文件：



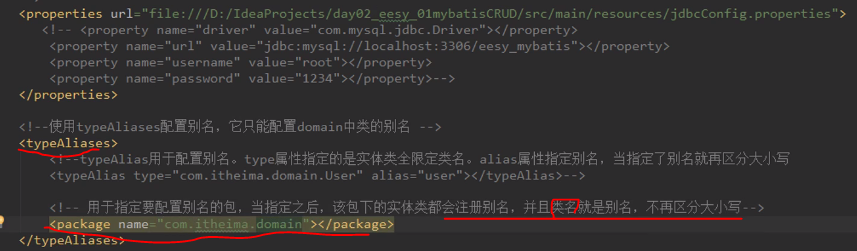
Mybatis主配置文件：



1. package标签用法

都是在mybatis主配置文件下进行配置

一、配置包(typealiases 只能配置domain中的别名)，则包下面类的全类名变为类名且不区分大小写

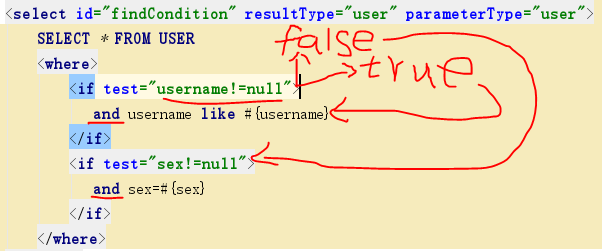


二、配置映射文件包，则不需要写mapper以及resource以及class



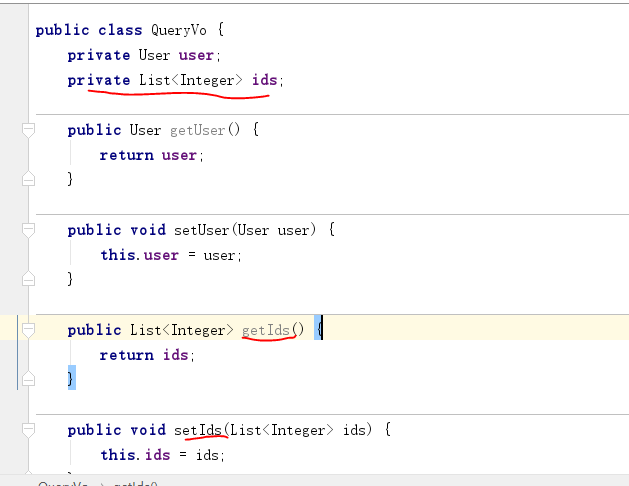
1. 动态sql语句查询

①where与if标签的使用

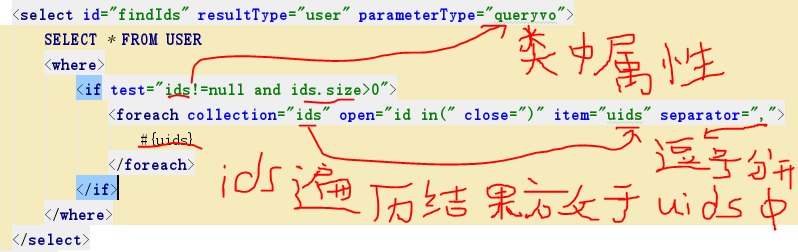


②foreach标签使用（针对 select \* from user where id in(78,12,45,63)）这种情况，可以把78,12,45,63封装到QueryVo类中

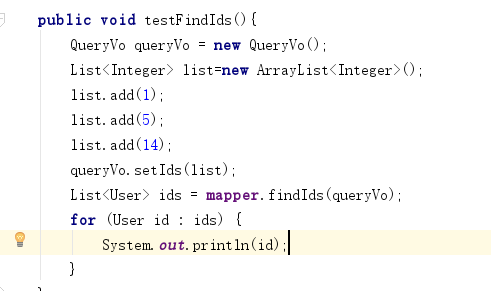
QueryVo类：将查询的多个id封装成queryvo类的属性，并生产getter和setter方法



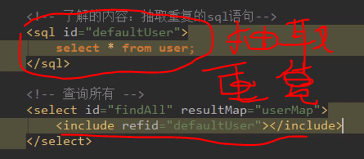
Sql动态语句书写：



测试函数：



1. 抽取重复的sql代码：



1. 查询二张表中的部分信息（**一对一查询**）：

**注意：在数据库创建的时候每张表的主键最好不要一样**

一、此方法不常用

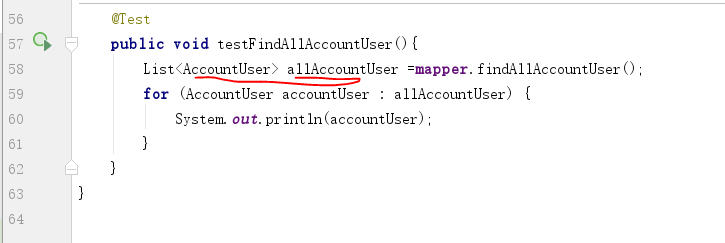
①先为要查的信息封装成一个实体

**package** com.cslg.domain;  
**public class** AccountUser **extends** Account {  
 **private** String **username**;  
 **private** String **address**;  
 **public** String getUsername() {  
 **return username**;  
 }  
 **public void** setUsername(String username) {  
 **this**.**username** = username;  
 }  
 **public** String getAddress() {  
 **return address**;  
 }  
 **public void** setAddress(String address) {  
 **this**.**address** = address;  
 }  
 @Override  
 **public** String toString() {  
 **return super**.toString()+**" AccountUser{"** +  
 **"username='"** + **username** + **'\''** +  
 **", address='"** + **address** + **'\''** +  
 **'}'**;  
 }  
}

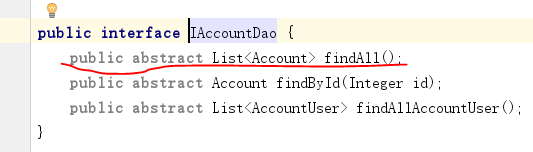
②dao包中接口的方法：注意返回值类型为①中建的类



③：测试类：



1. mybatis常用的方法
2. Dao包接口类查询方法不变，返回类型还是Account类型



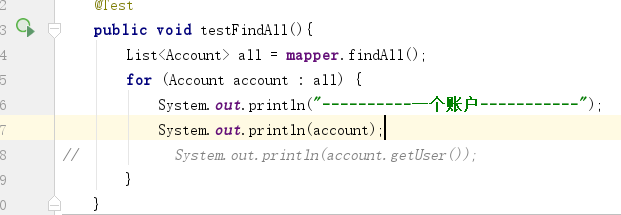
1. 在account(从表实体类)添加User(主表体类)的引用，并添加getter和setter方法（根据需求修改toStrng方法）

**package** com.cslg.domain;  
**import** java.io.Serializable;  
**public class** Account **implements** Serializable{  
 **private** Integer **id**;  
 **private** Integer **uid**;  
 **private** Double **money**;  
*// 从表实体包含主表实体的引用  
// 从表:Account 主表:user* **private** User **user**;  
 **public** Account() {  
 }  
 **public** Account(Integer id, Integer uid, Double money) {  
 **this**.**id** = id;  
 **this**.**uid** = uid;  
 **this**.**money** = money;  
 }  
 **public** Integer getId() {  
 **return id**;  
 }  
 **public void** setId(Integer id) {  
 **this**.**id** = id;  
 }  
 **public** Integer getUid() {  
 **return uid**;  
 }  
 **public void** setUid(Integer uid) {  
 **this**.**uid** = uid;  
 }  
 **public** Double getMoney() {  
 **return money**;  
 }  
 **public void** setMoney(Double money) {  
 **this**.**money** = money;  
 }  
 **public** User getUser() {  
 **return user**;  
 }  
 **public void** setUser(User user) {  
 **this**.**user** = user;  
 }  
 @Override  
 **public** String toString() {  
 **return "Count{"** +  
 **"id="** + **id** +  
 **", uid="** + **uid** +  
 **", money="** + **money** +  
 **", username="** + **user**.getUsername() +  
 **", address="** + **user**.getAddress() +  
 **'}'**;  
 }  
}

3、在该接口类映射文件中添加resultMap便签，用于指明sql查询的返回类型

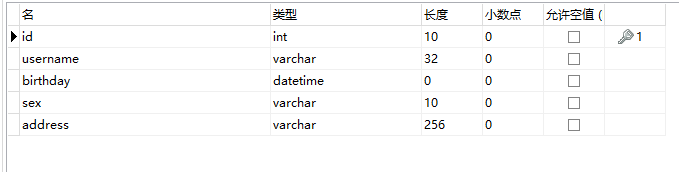


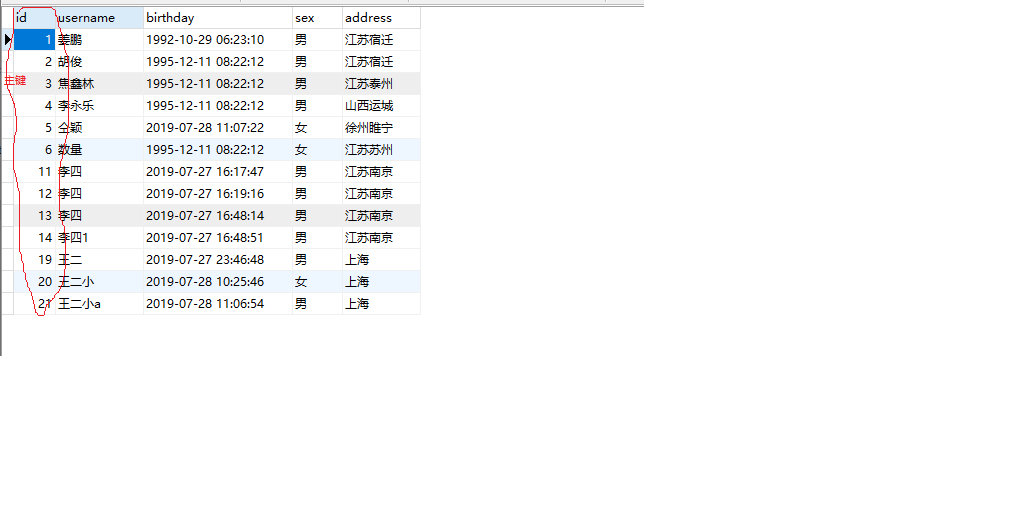
1. 测试方法



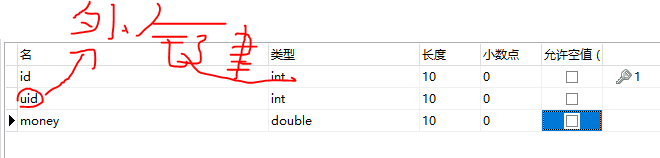
1. 一对多查询
2. 建表的时候需要注意：一的表的主键在多的表中充当外键（一个用户可以拥有多个账户）

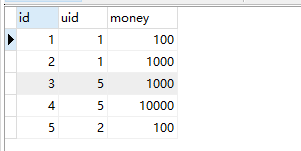
用户表：





账户表：





1. dao包不要变
2. 实体类添加一个集合类型的属性值，用于存放一个人可以拥有多个账户信息，并添加getter和detter方法

**package** com.cslg.domain;  
  
**import** java.io.Serializable;  
**import** java.util.Date;  
**import** java.util.List;  
  
**public class** User **implements** Serializable {  
 **private** Integer **id**;  
 **private** String **username**;  
 **private** Date **birthday**;  
 **private** String **sex**;  
 **private** String **address**;  
*//一对多关系映射，一个人可以有多个账户，故用集合来存取多个账户* **private** List<Account> **accounts**;  
 **public** User() {  
 }  
  
 **public** User(Integer id, String username, Date birthday, String sex, String address) {  
 **this**.**id** = id;  
 **this**.**username** = username;  
 **this**.**birthday** = birthday;  
 **this**.**sex** = sex;  
 **this**.**address** = address;  
 }  
  
 **public** Integer getId() {  
 **return id**;  
 }  
  
 **public void** setId(Integer id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getUsername() {  
 **return username**;  
 }  
  
 **public void** setUsername(String username) {  
 **this**.**username** = username;  
 }  
  
 **public** Date getBirthday() {  
 **return birthday**;  
 }  
  
 **public void** setBirthday(Date birthday) {  
 **this**.**birthday** = birthday;  
 }  
  
 **public** String getSex() {  
 **return sex**;  
 }  
  
 **public void** setSex(String sex) {  
 **this**.**sex** = sex;  
 }  
  
 **public** String getAddress() {  
 **return address**;  
 }  
  
 **public void** setAddress(String address) {  
 **this**.**address** = address;  
 }  
  
 **public** List<Account> getAccounts() {  
 **return accounts**;  
 }  
  
 **public void** setAccounts(List<Account> accounts) {  
 **this**.**accounts** = accounts;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "User{"** +  
 **"id="** + **id** +  
 **", username='"** + **username** + **'\''** +  
 **", birthday="** + **birthday** +  
 **", sex='"** + **sex** + **'\''** +  
 **", address='"** + **address** + **'\''** +  
 **'}'**;  
 }  
}

1. 其映射xml文件添加resultmap标记

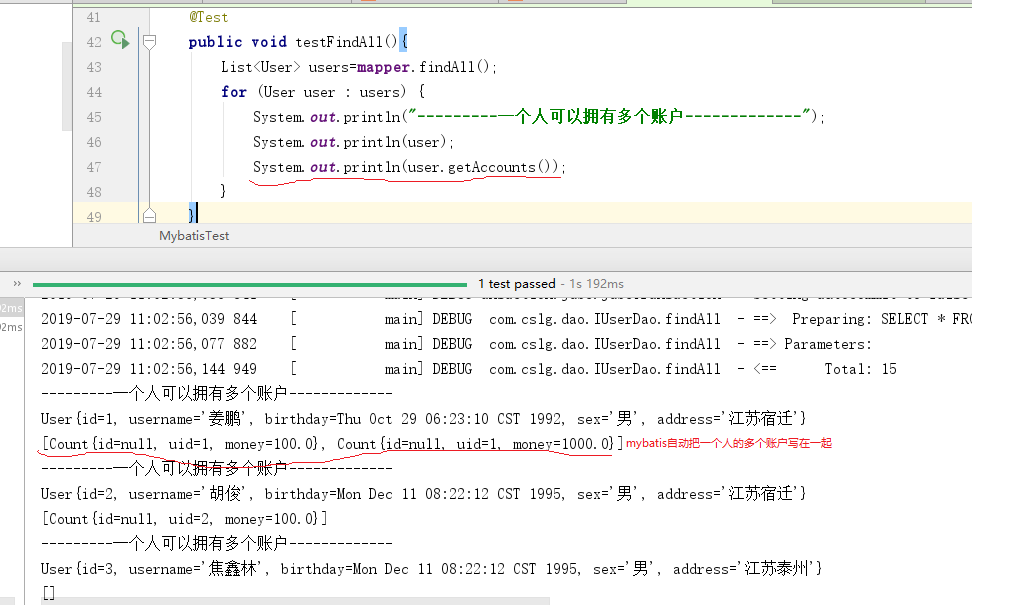
查询语句 ：

SELECT *\** FROM USER LEFT OUTER JOIN account ON user.id=account.uid

可以避免有的用户没有账户却不被显示的bug



1. 测试方法

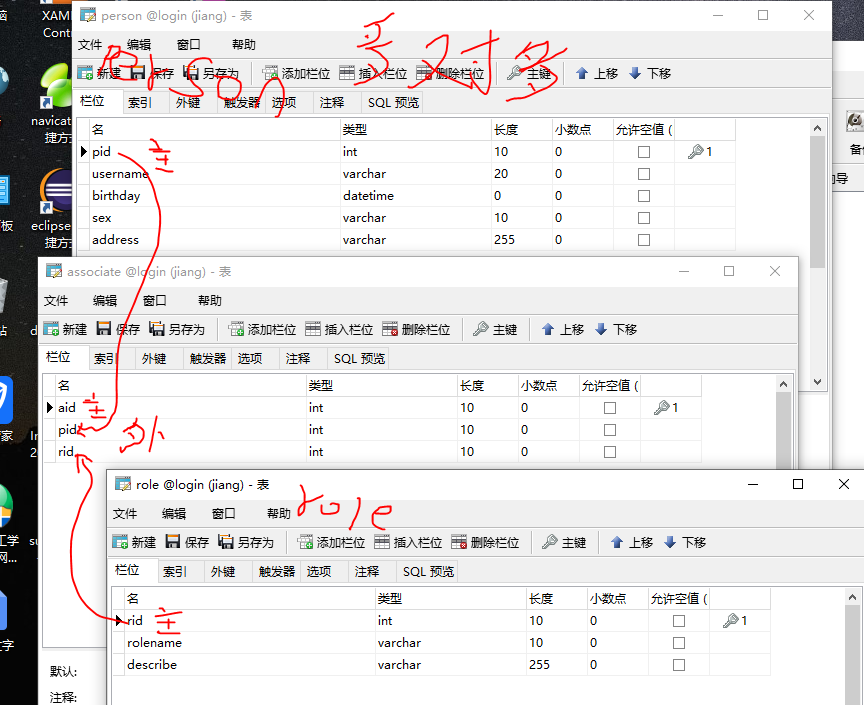


1. 多对多查询

多对多大部分和一对多一样。

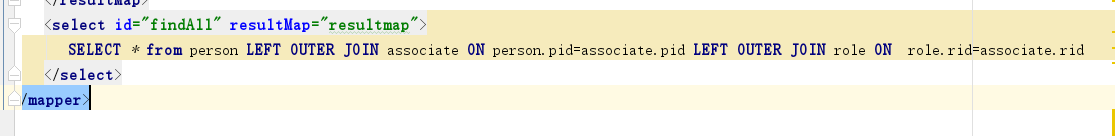
变的地方，要重新抽取出一张表表的外键分别有其他而张表的主键共同担任

查询语句：最好用 left outer join table on 条件 left outer join table on 条件



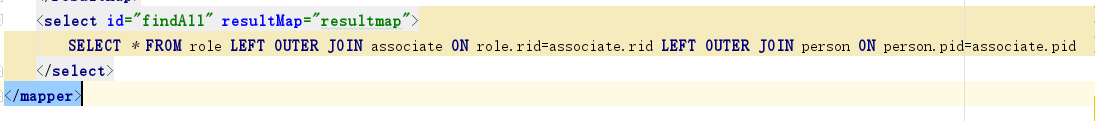
查人担任的角色：

SELECT *\** from person LEFT OUTER JOIN associate ON person.pid=associate.pid LEFT OUTER JOIN role ON role.rid=associate.rid



查角色由人扮演：

SELECT *\** FROM role LEFT OUTER JOIN associate ON role.rid=associate.rid LEFT OUTER JOIN person ON person.pid=associate.pid



1. 延迟加载

①一对一延迟加载

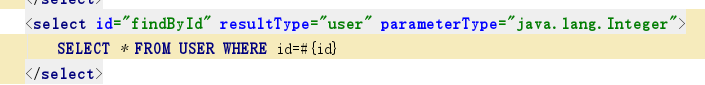
Xml配置：



开启延迟加载：



Findbyid:xlml

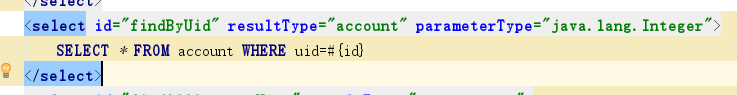


②一对多延迟加载

Xml:



Findbyuid:xml



开启延迟加载：同上

1. 一级缓存（缓存的是对象）：

一级缓存数据存储在sqlsession对象中：

Eg1:sqlsession.cleanCache();可以清楚缓存，之后不要创建sqlsession对象，因为sqlsession对象还存在。

Eg2::sqlsession.close();可以清楚缓存，之后要创建sqlsession对象和代理对象mapper，因为sqlsession对象被关闭就不存在了。



1. 二级缓存（缓存的是数据）

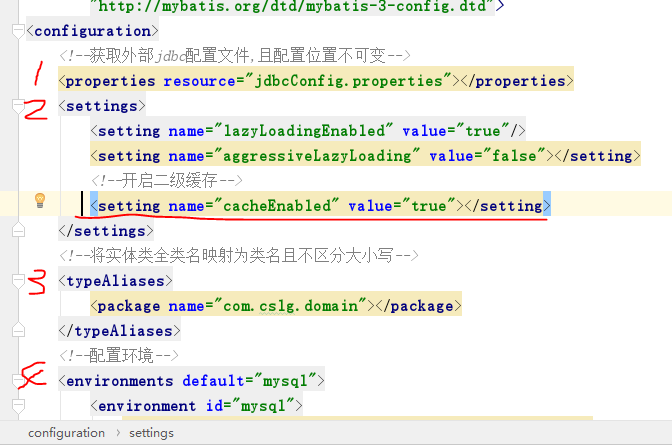
二级缓存：缓存的数据存放在sqlsessionFactory工厂里，有该工厂所创建出来的对象所共同享有，当获取相同的数据时会将数据封装到新的数据所对应的对象中

开启二级缓存的步骤

步骤一、

在sqlmapconfig.xml中开启二级缓存：

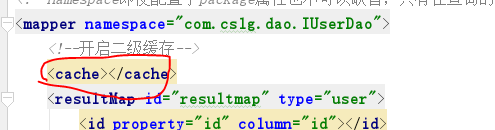
<setting name=”cacheEnabled” value-”true”></setting>



步骤二、

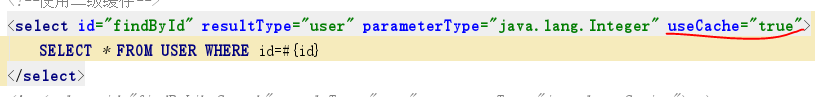
在dao包类的映射xml文件中开启二级缓存：

<cache></cache>

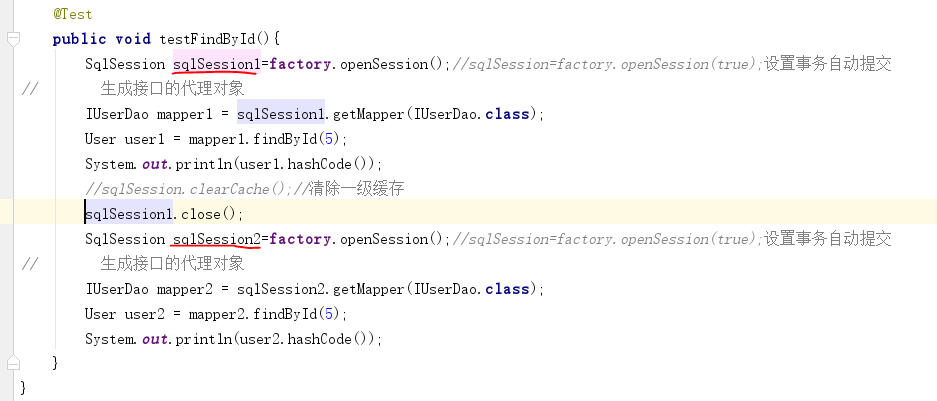


步骤三、

在查询语句中使用二级缓存useCache=”true”



测试：

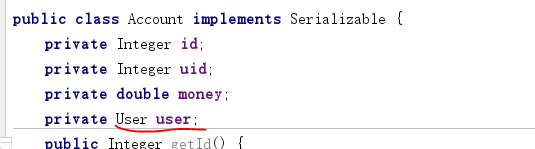


1. 注解配置解决实体名与数据库中表的列名不一致问题：

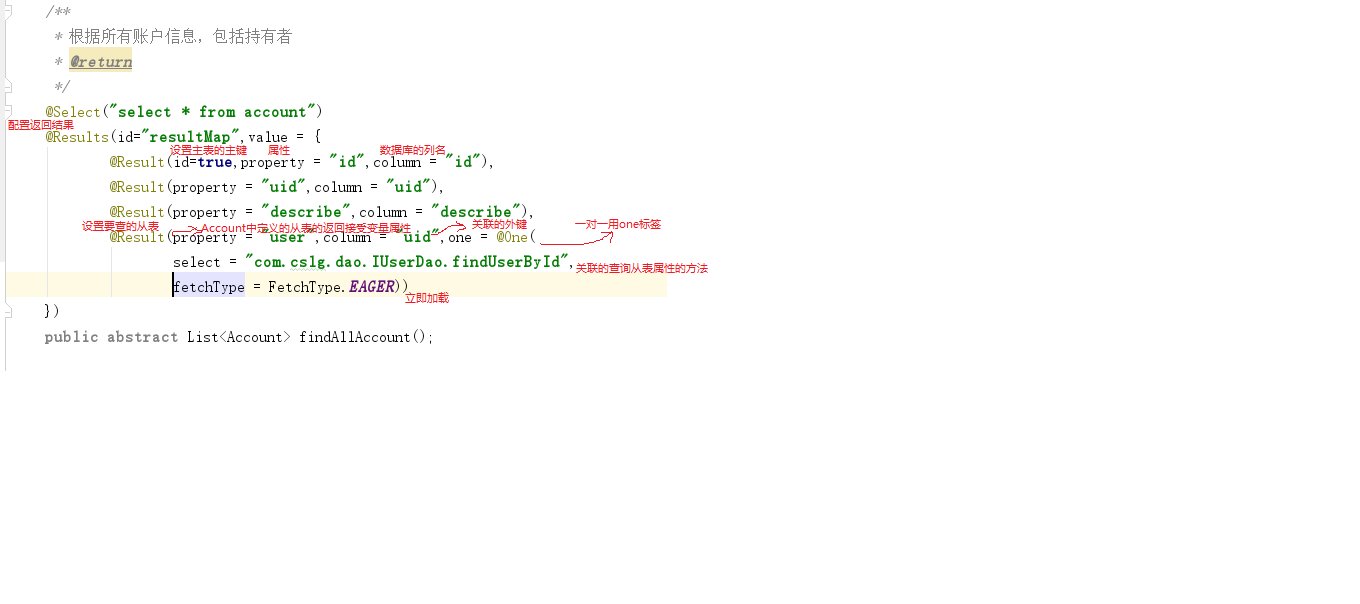


1. 一对一注解查询

①添加获取另一个实体的属性及getter和setter方法和在account的dao中配置查询方法：



Dao:

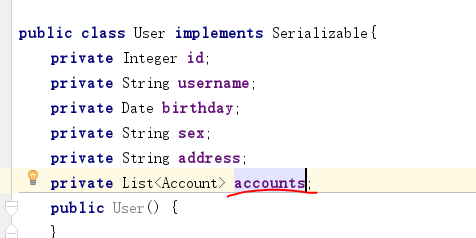


②在另一个实体dao中添加给根据用户id查询的注解查询：

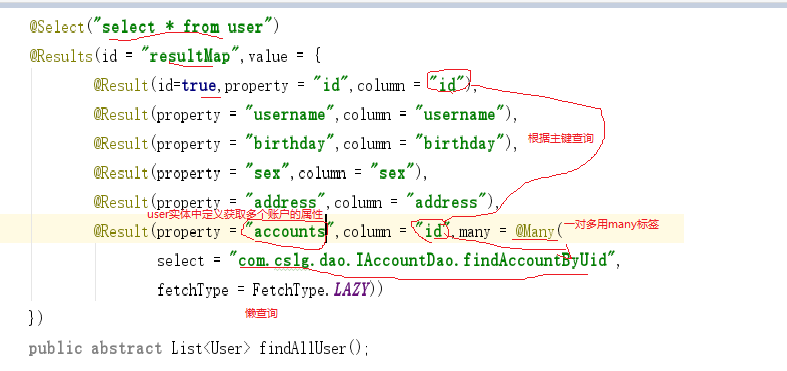


1. 一对多注解查询

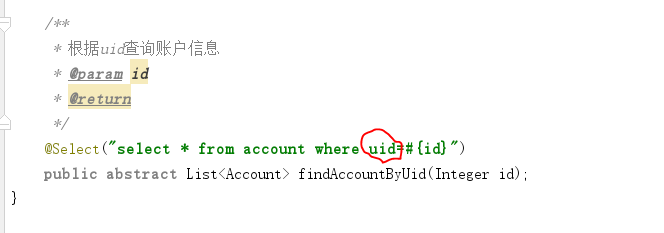
①添加获取另一个实体的属性及getter和setter方法和在account的dao中配置查询方法：



Dao:

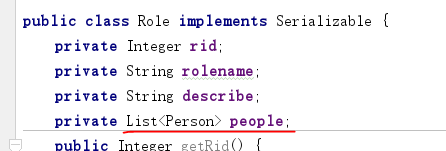


②在另一个实体dao中添加给根据用户id(uid)查询的注解查询：

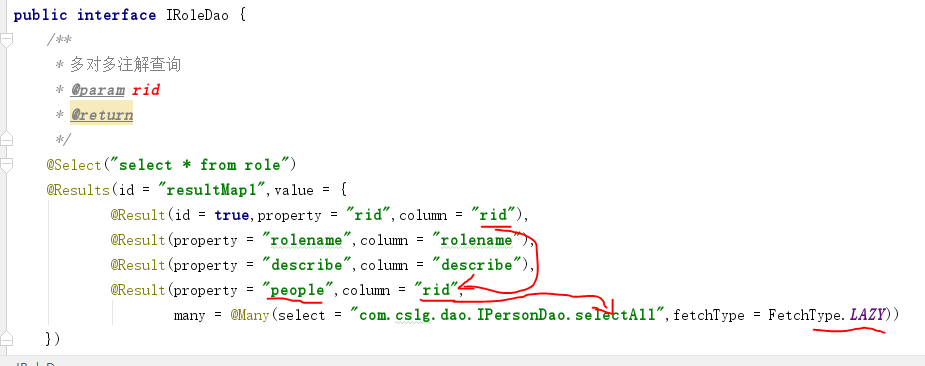


1. 多对多注解查询

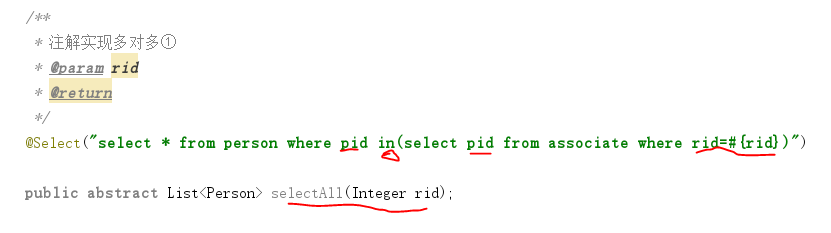
每个实体都要加从表实体作为属性



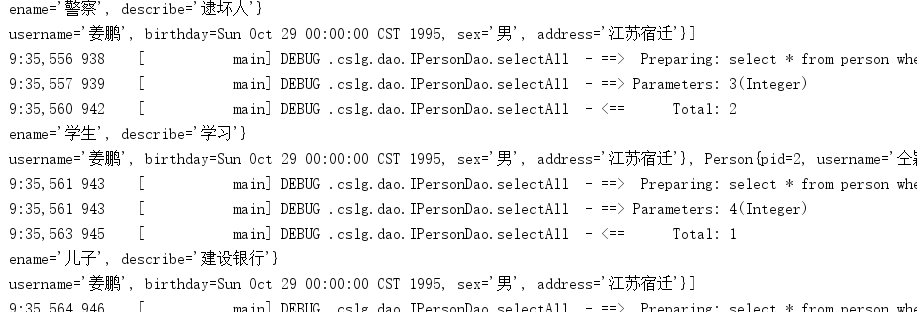
该实体接口的注解查询：



从表实体接口注解查询（涉及二张表）



测试：

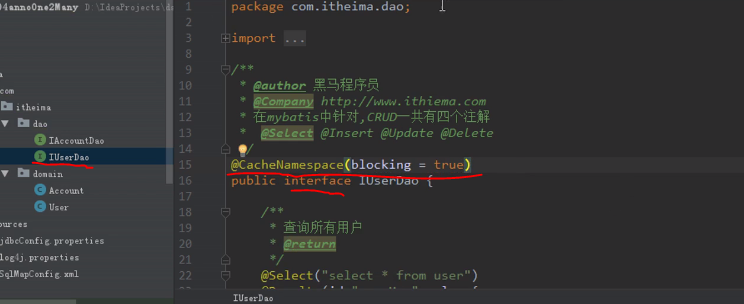


二十、注解配置开启二级缓存：

①、在sqlmapconfig.xml中开启二级缓存：

<setting name=”cacheEnabled” value-”true”></setting>

②、在dao接口中



二十一、mybatis标准

