# Mybatis

## 一、框架的基本概念

框架：是软件开发中的一套解决方案，不同的框架解决的是不同的问题。

使用框架的好处：框架封装了很多的细节，使开发者可以使用非常简单的方式实现功能，大大提高了开发效率。

三层框架：

①表现层：是用于展示数据的。

②业务层：是用来处理业务需求的。

③持久层：是和数据库交互的。

Mybatis是属于持久层的框架，SpringMVC框架则是属于表现层的框架。

### 1.持久层技术的解决方案：

**①JDBC技术**：即Connection、PreparedStatement、Resultst等

**②Spring的JdbcTemplate**：它是Spring中对jdbc的简单封装。

**③Apache的DBUtils**：它也是对jdbc的简单封装。

上面三者都不是框架，JDBC是一种规范，而JdbcTemplate和DBUtils都只是工具类。

## 二、Mybatis的基本概念

mybatis是一个基于java的持久层框架，它内部封装了jdbc操作的很多细节，使开发者只需要关注sql语句本身，而不用花费时间去处理加载驱动、创建连接等过程，它使用了ORM思想实现了结果集的封装。

**ORM(Object Relational Mapping)**：对象关系映射，简单的说就是把数据库表和实体类及实体类的属性对应起来，让我们操作实体类就可以实现操作数据库。

**mybatis的优点**：

**①**简单易学，是基于SQL编程的

**②**与JDBC相比，消除了JDBC大量冗余的代码，不需要手动开关连接。

**③**提供了很多第三方插件(如分页插件)

**④**能与Spring很好的集成。

**⑤**mybatis中，SQL可以写在XML里，从程序代码中彻底分离，解除sql与程序代码的耦合，便于统一管理和优化，并可重用。

**⑥**支持编写动态SQL语句。

**⑦**支持对象与数据库表的ORM关系映射。

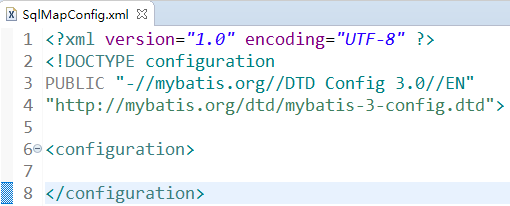
**mybatis的缺点**：SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

### 1.Mybatis中xml自动提示

为了在编写使Mybatis的xml文件中有提示，需要配置两个信息，一个是Config，一个是Mapper。

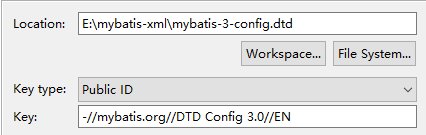
**配置Config**：

①下载对应文件，并复制下列红框中的内容。



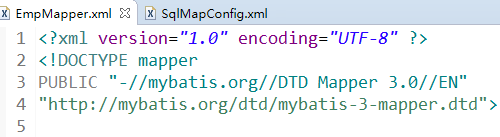
**按住Ctrl+点击上面这个http://中的内容，下载对应的文件到本地。**

②打开window–>Preferences–>XML–>XML catalog，点击add，在Location中点击右边的File System，把选择刚刚下载的文件。然后Key type选择Public ID，对应的key则是刚才复制的内容。



**配置Mapper**：

所对应的操作和上面类似：复制红框中的内容并下载文件，然后在window–>Preferences–>XML–>XML catalog，点击add，选择文件，粘贴内容即可。



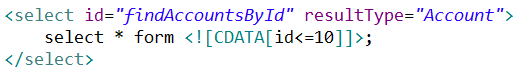
**按住Ctrl+点击上面的http中的内容，然后下载相应的文件。**

做完上面这几部，然后重启Eclipse即可。

### 2.mybatis中特殊字符处理

①使用**<![CDATA[]]>。推荐这种方式**。

如id>10中的>是特殊字符，在mybatis中直接使用是不行的。



②使用转义：&lt(<)、&gt(>)、&amp(&)、&apos(‘)、&quot(“)。



### 3.内置的数据类型的别名

byte(\_byte)、short(\_short)、long(\_long)、int(\_int、\_integer)、double(\_double)、float(\_float)、boolean(\_boolean)、String(string)、Byte(byte)、Long(long)、Short(short)、Integer(int、integer)、Double(double)、Float(float)、Boolean(boolean)、Date(date)、BigDecimal(decimal、bigdecimal)、**Object(object)、Map(map)、List(list)、Set(set)**。

## **三、mybatis使用xml的方式**

### 1.mybatis的环境搭建

**①创建一个简单的Maven工程，在pom.xml中添加mybatis、mysql的依赖，这两个依赖是必须的**。还可以添加一些其它依赖比如log4j、Junit等，还可以添加jdk1.8的插件。

**②创建实体类和相应的dao接口**。

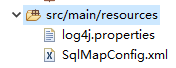
**③创建mybatis的主配置文件**：如SqlMapConfig.xml。

在src/main/resources目录下创建一个SqlMapConfig.xml文件，**这个xml文件的约束必须使用对应的约束**，主配置文件只有一个。

注意：在SqlMapConfig.xml文件中需要使用<mapper>配置指定映射配置文件的位置，映射配置文件是xml的形式的话需要使用resource属性指定该如UserDao.xml文件的位置。

[SqlMapConfig.xml的配置](#_十二、SqlMapConfig.xml的配置)

**强烈建议**：**最好将log4j.properties文件直接粘贴到和主配置文件同样的位置下，这样方便查看日志详情**。



**④创建映射配置文件**：如UserDao.xml。

在src/main/resources目录下在创建相应的包用来存放新建的映射配置文件，如com/aowin/dao/UserDao.xml，每一个实体类的Dao接口都对应一个这样的文件，命名规则最好是**实体类+Dao.xml**，**这个xml文件的约束也必须使用对应的约束**。xml中的sql语句可以写分号也可不写。[映射配置文件中的属性](#_八、映射配置文件中的属性)

### 2.环境搭建的注意事项

**①**实体类的Dao都写成接口，一个实体类的Dao对应一个实体类的Dao.xml文件。(推荐)。在mybatis中它把持久层的操作接口和映射文件称作Mapper。

**②**mybatis中的主配置文件和映射配置文件最好放在src/main/resources文件夹下。(推荐)

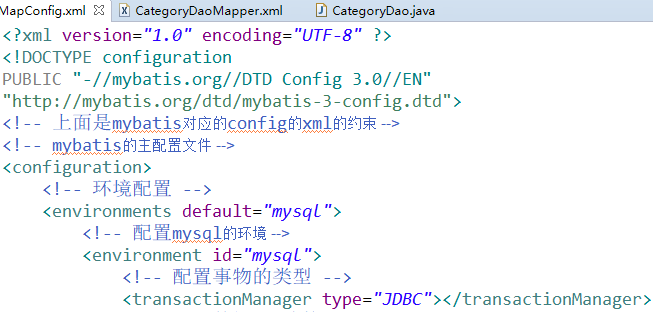
**③mybatis的映射配置文件的位置必须和dao接口的位置是处于同级目录下且是在同样的包中**，如UserDao.xml在src/main/resources目录下的com.aowin.dao包中，而UserDao则在src/main/java目录下的com.aowin.dao包中。Mybatis中接口和对应的映射配置文件不一定要放在同一个包下，放在一起的目的是为了Mybatis进行**自动扫描**，并且要注意此时**dao接口的名称和映射配置文件的名称要相同**，否则可能会报错。

**④映射配置文件的mapper标签的namespace属性的值必须是dao接口的全名称(包名+类名)**。(**强制**)

**⑤映射配置文件的操作配置中，比如<select>、<insert>等操作，它们的id属性的取值必须是dao接口的方法名**。(**强制**)。**执行完sql语句之后的结果集会存放在一个List集合中，而resultType属性的值则是指定这个List集合中存放的元素类型**。执行查询语句，resultType的值一般是对应的实体类的全名称。

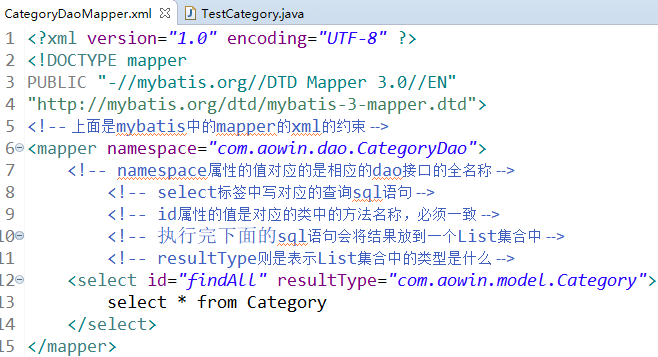
在遵循了上面的第③④⑤点之后，在开发中无需再写dao接口的实现类。

**SqlMapConfig.xml配置文件的信息范例**：





**每个dao接口对应的映射配置文件的范例**：



### 3.最好遵循如下约束

**①**dao接口放在src/main/java中，dao接口对应的映射配置文件放在src/main/resource目录中。dao接口的包名和映射配置文件的包名一致，如都是com.aowin.dao。dao接口的名称和映射配置文件的名称一致，如UserDao.java对应的映射配置文件为UserDao.xml，或者UserMapper.java和UserMapper.xml。

满足上面这个条件之后，则可以在主配置文件中配置映射文件时使用<package>标签配置。[子标签<package>](#_⑵子标签<package>)。若dao接口和对应的配置映射文件不在同级目录下或包名不同或名称不一致，则使用<package>标签进行自动映射时会报错。

若是在主配置文件中使用<mapper>标签的resource属性或者class属性进行定位映射配置文件，则上面的名称、包名可以不一致。

[dao接口和mapper文件的存放位置](#_十三、dao接口和mapper文件的存放位置)

### 4.mybatis的入门案例

当mybatis的环境搭建好之后，则可以进行下面的测试操作。

在src/test/java中创建com/aowin/test包的一个测试类。在该测试类中进行如下操作并运行代码：

注意下面导包导的是org.apache.ibatis下面的。

**步骤**：

**①读取配置文件：Resources.getResourceAsStream(主配置文件)，返回一个InputStream流。**

**②创建SqlSessionFactory工厂：SqlSessionFactoryBuilder调用build()方法接收文件流，并创建该工厂。**

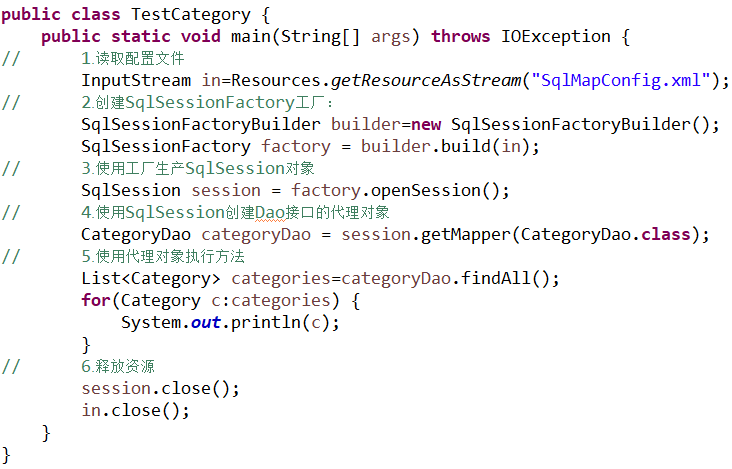
**③使用工厂的openSession()方法生产SqlSession对象**

**④使用SqlSession对象的getMapper()方法创建Dao接口对象**

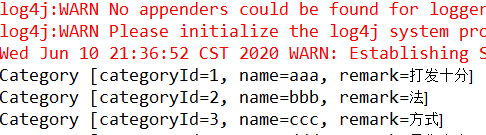
**⑤使用Dao接口对象调用相应的方法**

**⑥关闭SqlSession和InputStream资源**。

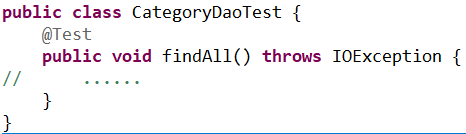
#### ⑴测试实例：



控制台上输出的结果如下：



**上面的测试类的代码最好写成下面这种形式**：



#### ⑵针对测试实例作分析

**①读取配置文件使用路径问题**：

可以使用绝对路径、相对路径，但是会出现如下问题：

当使用绝对路径时，比如/D:\xxx\SqlMapConfig.xml，在别人的电脑上没有D盘的话就会找不到这个配置文件。

当使用相对路径时，如src/main/resources/SqMapConfig.xml，如果是web工程，则一部署的话这个src目录就会没了，也会导致找不到这个配置文件。

因此，在读取配置文件的路径上不怎么使用绝对路径和相对路径，而是采用如下方式：

**法一**：使用类加载器，但它只能读取类路径的配置文件。

**法二**：使用ServletContext对象的getRealPath()，它能得到一个当前项目部署的绝对路径，项目部署在哪，这个绝对路径就在哪。

**②创建SqlSessionFactory工厂，mybatis使用了构建者模式**。(这个有空要好好理解)。构建者者模式的优点是把对象的创建细节隐藏，使用者直接调用方法即可获得对象。

**③创建SqlSession对象，mybatis使用了工厂模式**。工厂模式的优点是解耦(降低类与类之间的依赖关系)。

**④创建相应实体类的dao接口，mybatis使用了代理模式**。该模式的优点是在不修改源码的基础上对已有方法增强。

## 四、mybatis使用注解的方式

注意mybatis中使用注解的方式进行开发，项目中最好不要包含dao接口对应的映射配置文件，否则运行时可能会报错。

使用注解的方式导包：**import** org.apache.ibatis.annotations.\*。注意当某个注解报错xxx cannot be resolved to a type时，可能是没导入包，直接使用上面这种方式导包，或者找到对应的包导入。

### 1.mybatis的环境搭建

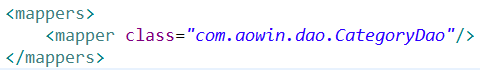
区别在于不再需要配置映射配置文件，而是采用注解的方式。同时主配置文件中的<mapper>不再使用resource属性指定配置文件的位置，而是使用class属性指定被注解的dao的全名称。

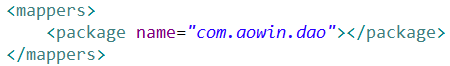
**①创建一个简单的Maven工程，在pom.xml中添加相应的依赖和插件**。(同前面的笔记类似)

**②创建实体类和相应的dao接口**。

**③创建mybatis的主配置文件**。如SqlMapConfig.xml。同样加那个log4j.properties文件放到和SqlMapConfig.xml同样的位置。

**注意**：使用注解的方式的话，主配置文件中的配置映射文件的位置的<mapper>标签不再使用resource属性指定xml文件的位置，而是使用class属性指定相应接口的全名称。也可使用<package>标签的name属性指定dao接口所在的位置。如下所示：





[SqlMapConfig.xml的配置](#_十二、SqlMapConfig.xml的配置)

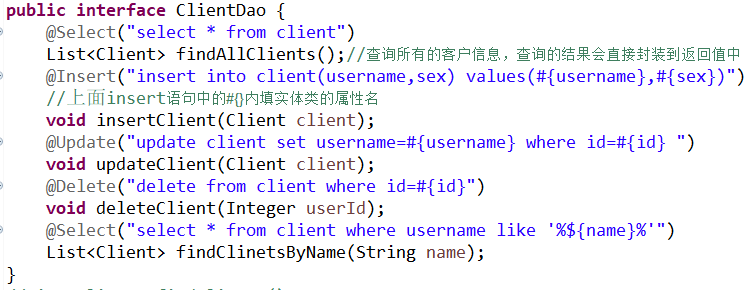
**④在dao接口中配置映射**：采用注解的方式的话，则不再需要写相应的xml文件，如UserDao.xml。而是直接在Dao接口的方法上使用相应的注解，并写相应的sql语句。

### 2.mybatis中的注解

**①@Select()、@Insert()、@Update()、@Delete()**

当上面注解中都是对应单条SQL语句时，可以直接写成@Select(“单条SQL语句”)，但标准写法是@Select(value=”SQL语句”)。

下面Client实体类的属性和数据库中client表的字段一致。



**②@Results()**

当实体类的属性名和数据库对应表的字段名不一致时，使用@Results()标签进行配置。

**@Results(id=“起名”，value={**配置多个@Result注解**})**，每个@Result之间用逗号隔开。

**@Result(property=“类的属性名”,column=“表的字段名或别名”)**

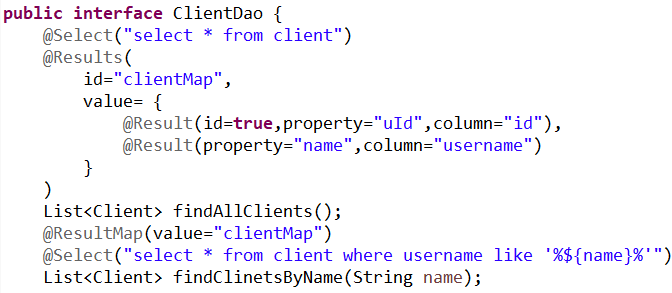
**若是主键，则还要添加一个id=true**。



配置成上面这样只能对当前方法有效，若要使其它方法也有效，则要在其它方法中使用@ResultMap()注解引用这个@Results()。

**③@ResultMap()**

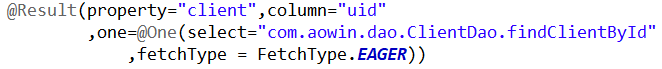
[标准写法@ResultMap(value=”clientMap”)](mailto:标准写法@ResultMap(value=)，若只有一个参数则可以省略value，@ResultMap(“clientMap”)。



### 3.一对一使用注解配置

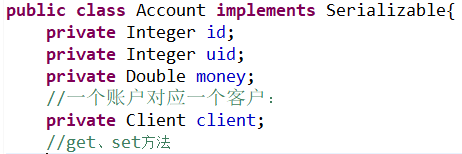
这个@One注解是定义在@Result中的。

**@One**：表示一对一的情况，**select属性**的值为**包名+类名+方法名**对应的方法。**fetchType属性**的值用来设置是延迟加载还是立即加载。若是延迟加载则fetchType=FetchType.LAZY，若是立即加载则是fetchType=FetchType.EAGER。

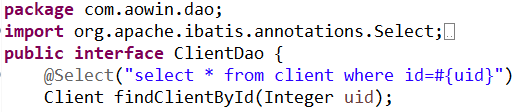


**例子：**

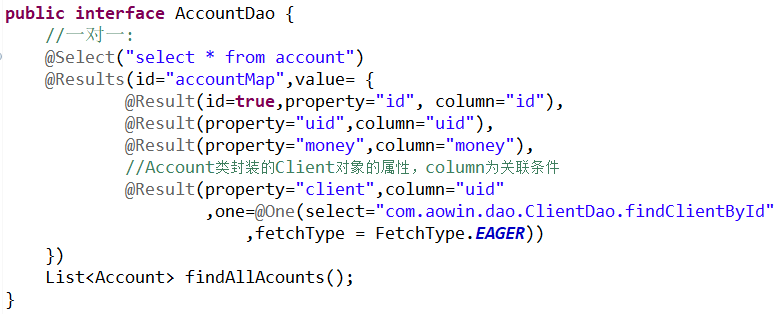
**Account实体类**



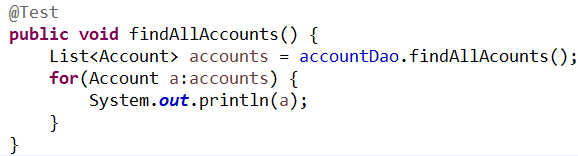
**ClientDao接口**



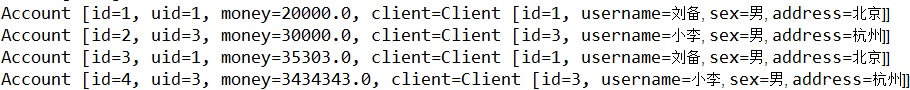
**AccountDao接口**



**测试方法**



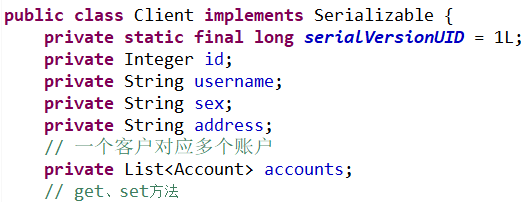
**结果**



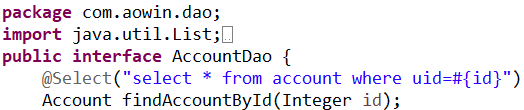
### 4.一对多使用注解配置

**@Many**：表示一对多的情况。用法和@One一样。一对多采用延迟加载。

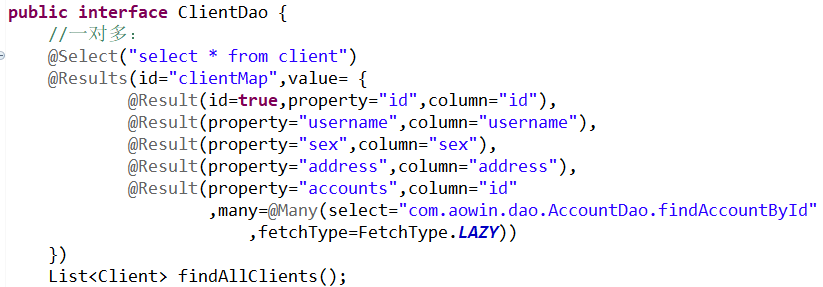
**Client实体类**



**AccountDao接口**



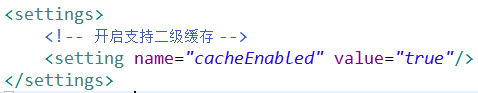
**ClientDao接口**



### 5.使用注解配置二级缓存

二级缓存的概念：[缓存](#_2.缓存)

**步骤一**：在主配置文件中配置



**步骤二**：在对应的Dao接口上使用**@CacheNamespace(blocking=true)**即可使用二级缓存了。

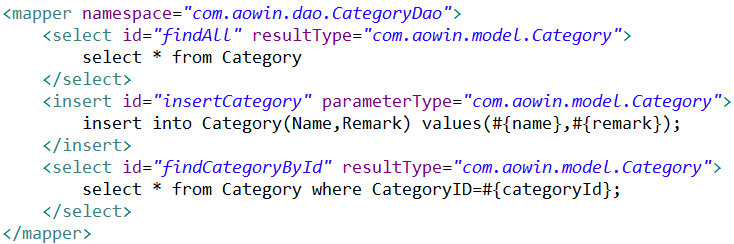


## 五、实现接口的方式

### 1.实现接口的方式

也可采用对dao接口进行实现的方式进行操作，但是不推荐这样使用，只是为了说明和以前每个dao接口对应一个实现类联系起来。

**①**CategoryDao.xml的映射配置文件部分内容如下所示：

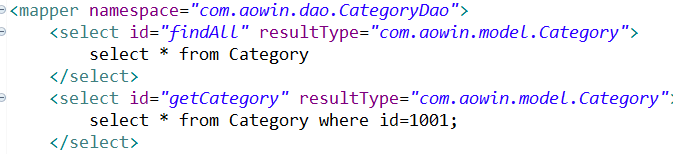


**②**写一个CategoryDao接口的实现类CategoryDaoImpl，重写相应的方法。

查询列表使用selectList()方法，查询单个则用selectOne()方法。

在重写的方法中封装SqlSession对象的创建，并调用SqlSession对象相应的方法如：selectList()方法执行查询sql语句，

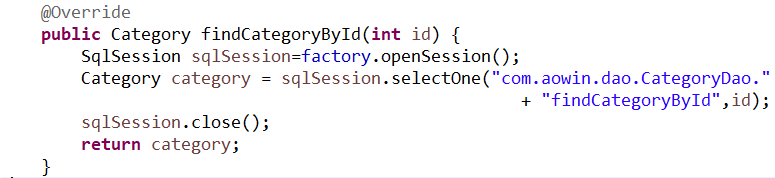
注意SqlSession的selectList()方法要执行哪条sql语句是由配置映射文件CategoryDao.xml文件中<mapper>的属性namespace值和相应的<select>的id属性值指定的。如namespace的值为com.aowin.dao.CategoryDao，<select>的id值为findAll，则session.selectList("com.aowin.dao.CategoryDao.findAll")表明要执行的是定位到的下面这条select \* from Category的sql语句。



**接口实现类代码例子**：

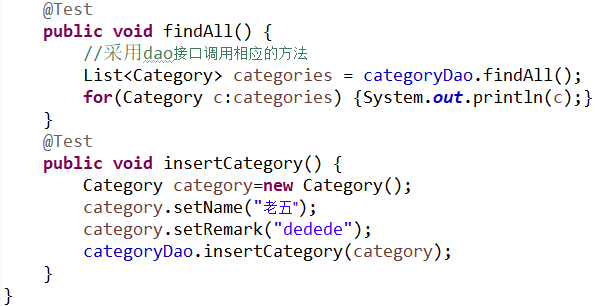


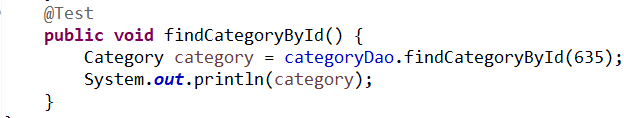




**③**测试类的代码如下所示：







### 2.分析上面代码的执行过程

#### ⑴主配置信息SqlMapConfig.xml中的内容：

**①**有下面四个连接数据库的基本信息就可以创建Connection对象。

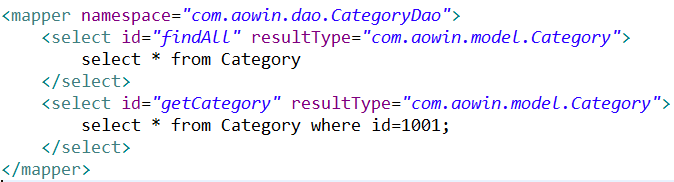


**②**有了下面这个信息就有了映射配置信息，就映射到相应的映射配置文件中。



#### ⑵映射配置文件CategoryDao.xml中的内容：

**①**在映射配置文件中有如下的信息则就有了执行的SQL语句，就可以获取PreparedStatement对象，此配置中的namespace属性还指定了对应的实体类的全名称，id属性指定了相应的方法名称。



**<select>标签包含两个信息**：**执行的SQL语句和用来封装结果的实体类的全名称**。将这两个信息组合起来定义成一个对象，这个对象有一个String类型的sql和String类型的实体类的ClassPath，若有多个标签则有多个对象。这些对象被存放在key-value的Map中，value是这些对象，key则是String类型的namespace和id组合的值。

#### ⑶读取上面的配置文件信息：

使用到的技术就是解析xml文件的技术，此处使用的是dom4j解析xml的技术。

#### ⑷ selectList()方法的实际执行过程：(好好理解)

**实际上是根据反射的方式进行数据封装。**

**①**根据配置文件的信息创建Connection对象。这里包括注册驱动、获取连接。

**②**获取预处理对象PreparedStatement。创建PreparedStatement对象采用的是con.prepareStatement(sql)方法，所以这里需要使用到SQL语句。

**③**执行SQL语句，并获得结果。

如执行查询：ResultSet rs=ps.execute.Query()。

**④**遍历结果集并封装。

List<E> lis=new ArrayList<E>

while(rs.next()){  
 E e=xxxx;

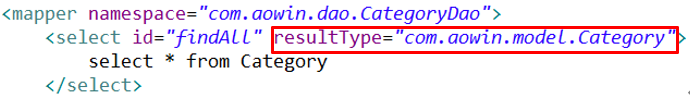
把每个rs中的内容添加到e中，然后再把e添加到list中。

list.add(e)

}

**⑤**将结果返回。

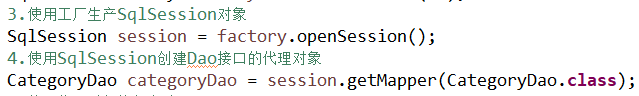
**注意**上面的这个E e=xxx，实际上是通过反射的方式创建的对象，在映射配置文件的resultType属性中指定了类的全名称，通过反射Class.forName(类全名称)的方式创建出该对象，然后再进行强转。即E e=(E)Class.forName(类的全名称);



**注意**：由于我们的实体类属性和表的列名是一致的，因此可以把表的列名看成是实体类的属性，就可以利用反射的方式根据名称获取每个属性，然后把值赋进去。这就是上面的把每个rs中的内容添加到e中。

#### ⑸分析SqlSession创建代理对象

分析下面这部分代码：



根据SqlSession的getMapper(Class<T> c)方法创建代理对象，实际上是使用的jdk代理，getMapper()方法中实际上封装了Proxy.newProxyInstance()方法创建代理对象。

### 3.自定义mybatis的案例

根据2中的代码执行过程分析，尝试在不使用mybatis的依赖情况下自己完成mybatis的功能，黑马视频中讲过，尝试自己写出完整代码。(非常有时间的情况下做这事)。

## 六、SqlSession对象

采用SqlSessionFacotry工厂创建的SqlSession对象有**两大作用**：

**①生成相应的dao接口代理对象**。

调用getMapper()方法创建接口代理对象，参数为相应的接口的Class。调用getMapper()创建代理对象实际过程分两步：

第一步：先用SqlSessionFactory读取数据库连接信息创建Connection对象。

第二步：通过jdk代理模式创建出代理对象，Proxy类的newProxyInstance()方法，该方法的第三个参数InvocationHandler 对应的invoke()方法中对应获得的sql语句进行CRUD(增删改查)操作。

**②定义通用的增删改查方法**。如selectList()、insert等方法。这些方法实际也分两步：

第一步：先用SqlSessionFactory读取数据库连接信息创建Connection对象

第二步：直接得到Sql语句，然后使用Connection对象进行CRUD操作。

**注意SqlSession在操作增、删、改操作时，默认不是自动提交的，因此需要自己手动提交**。

## 七、解决报错

当在创建Maven工程时，当该工程的依赖导入时报错，更新一个下该工程，若还报错，则去本地仓库中查找报错的依赖，找到对应的版本的依赖，可能是该依赖下载的内容不全，清除依赖中带有lastUpdate的内容的文件，然后重新更新一下项目，或者将这个版本的依赖删除，重新更新项目重新下载完整的依赖。

**注意xml中的注释使用<!-- -->这种方式，不支持/\* \*/这种**。

## 八、映射配置文件中的标签

### 1.<mapper>标签中的属性

**namespace属性**：**填对应的dao接口的全名称**。

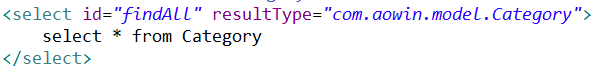


### 2.<select>、<insert>等标签中的属性

**id属性**：**对应dao接口中的方法名称**。

**resultType属性**：**结果类型**。执行sql查询语句获得的数据存放到List集合中时，resultType的值就是指定这个List集合中存放什么类型的数据，如List<Category>，注意这里resultType的值必须是类的全名称，因为是通过Class.forName()方式创建这个类的对象实例，然后给这个对象的属性赋值，然后再存放到List集合中。

注意当数据库中表的字段名和实体类的属性名不一致时，则不能使用resultType，而是使用<resultMap>标签进行配置。[<resultMap>标签中的属性](#_5.<resultMap>标签中的属性)

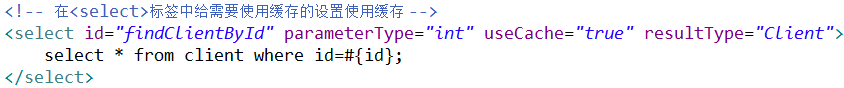


**parameterType属性**：**参数类型，指定方法的参数是什么类型的**。对于方法中传入的参数有多个，则可以不用指定parameterType。

**resultMap属性**：这个属性的值和<resultMap>标签的id属性值一致。用法：[<resultMap>标签中的属性](#_5.<resultMap>标签中的属性)

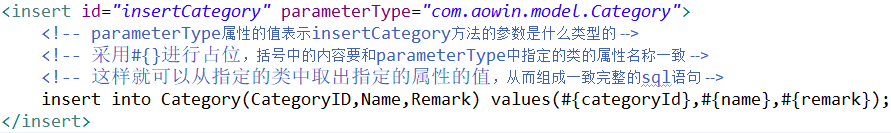
在执行查询sql语句时，若dao接口中的与查询语句有关的方法含参数，则在配置映射配置文件时，其对应的<select>标签中的属性则有id、resultType和parameterType。若为无参的方法，则<select>标签中的属性有id、resultType。

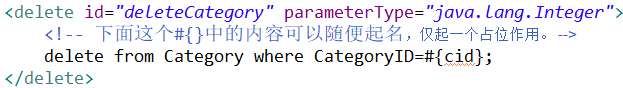
**useCache=“true”**：当<select>标签对应的查询SQL语句操作需要使用二级缓存时，设置这个。具体操作：[二级缓存的使用](#_⑵二级缓存)



### 3.占位符：#{}、${}

**#{}**：进行占位，**①**括号中的内容要和parameterType中指定的类的属性名称一致，同时类的属性名称必须有对应的get、set方法，因为mybatis是根据parameterType中指定的类的全名称，采用反射的方式获得这个类和这个类中的get方法，从而获得相应属性的值。**②**对于parameterType参数类型是基本数据类型或者基本数据类型对应的包装类时，只有一个参数时，#{}中的内容可以随意起名。





#{}和${}都可以用来占位，获取参数的值给SQL语句赋值。其中#{}是以预编译的方式，采用PreparedStatement方式将参数设置到sql语句中，可以有效防止SQL注入。而${}采用Statement的方式，将取出的值直接拼接到sql语句中，会有SQL注入问题。

注意关键字拼接时只能用$，不能用#，因为#采用的是PreparedStatement方式，对于字符串类型会自动添加单引号。

### 4.<selectKey>标签中的属性

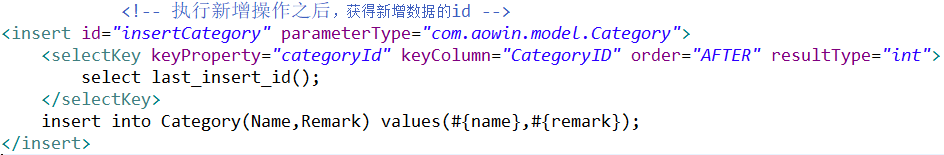
**<selectKey>标签主要是用来设置主键值的返回。**

**KeyProperty属性**：**表示返回的主键值存放到哪**。一般对应model类中的属性名(这个属性是和数据库中主键名相对应的属性)，如当传入的参数类型paramerType的值为自定义的类时，KeyProperty的值为这个类的属性名。

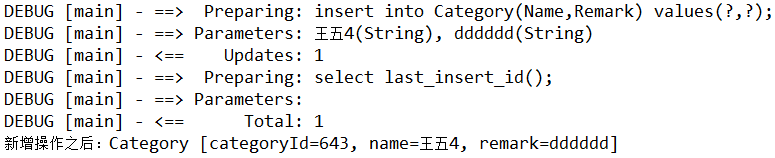
**KeyColumn属性**：**是数据库中表的主键名**。

**order属性**：**代表什么时候执行操作**，插入之前执行则为BEFORE，插入之后执行则是AFTER。AFTER和BEFORE必须都大写。**其中当表的主键是自增时，使用AFTER，当表的主键不是自增时，使用BEFORE**(为什么自增就用after，不是自增就用before，要弄清楚)。**AFTER表示先执行插入语句，然后再返回插入数据的主键值**，即先执行insert into....插入语句，然后再执行<selectKey>中的select last\_insert\_id()语句返回主键值。**Before则是先返回主键值，然后再执行插入语句**。

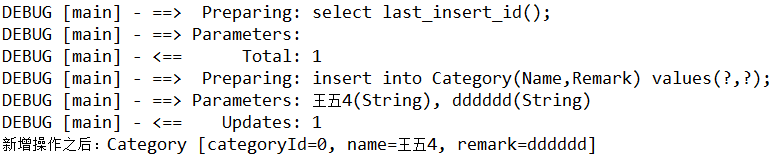
**resultType属性**：**表示查询的数据库中表的主键的数据类型**。



①当order的值为After时，执行语句顺序如下：



②当order的值为before时，执行语句顺序如下：



### 5.<resultMap>标签中的属性

<resultMap>标签主要用来配置查询结果的列名和实体类的属性名的对应关系。

**type属性**：实体类的全名称。若在主配置文件中起了别名则可以使用别名。[子标签<package>](#_⑵子标签<package>)

**id属性**：随便起名。

**在<resultMap>标签下采用<id>标签来配置主键字段的对应关系**，其中，**<id>标签的属性如下**：

**property属性**：实体类中的属性。

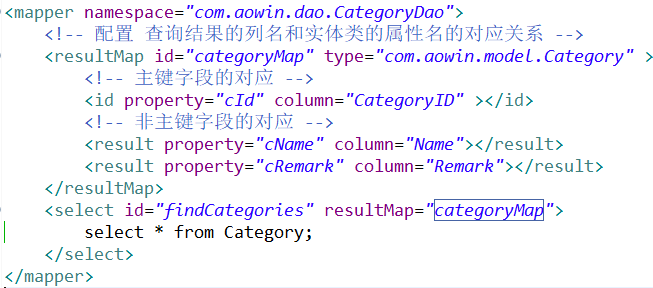
**columnn属性**：表中的列名，也可以是select语句中起的别名

**javaType属性**：就是实体类中属性对应的数据类型。

**jdbcType属性**：就是表的列名所对应的数据类型。

一般实体类的属性的数据类型和数据库中表的列名的数据类型是一致的，所以javaType属性和jdbcType属性可以省略。

**在<resultMap>标签下采用<result>标签来配置非主键字段的对应关系**。<result>标签中的属性和<id>标签中的属性一样。



### 6.<association>标签

在mybatis中，一对一的关系映射采用<association>标签。

例子：账户表中的一个账户只对应客户表中的一个客户，账户表的uid和客户表的id关联起来了。要求查询出账户表的所有信息以及显示客户的名称、性别和地址信息。

可以在Account实体类中封装一个Client实体类的对象，这样Account实体类中就包含Client的名称、性别、地址等信息。在映射配置文件AccountDao.xml中可以很容易将查询结果映射到对应的Acount实体类的基本类型的属性，但如何同样将查询结果映射到Acount实体类的属性Client对象的属性中，则要通过<association>标签将查询结果中关于client表的信息映射到Account实体类中的Client对象中。

#### ⑴该标签的属性

**property属性**：为实体类中的一个属性名，一般值可以为A类中封装的另一个B类的**对象引用变量**，**不能是类名**。

**column属性**：表示用表的哪个字段的值作为关联查询的条件。可指定，也可不指定，**当<association>嵌套select语句时才必须使用这个属性**。column的值一般是key-value形式。 当只有一个关联条件时可以不写key，若有多个关联条件则要使用key-value的形式。

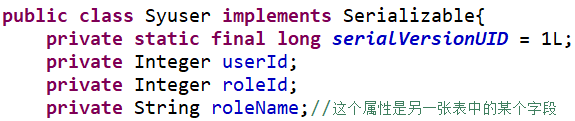
**javaType属性**：指定property属性中对象引用变量的类型。可以不用指定，mybatis中可以自动识别。若没有起别名则要写全名称。

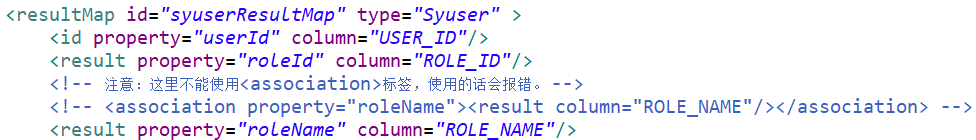
**resultMap属性**：对应一个定义好的<resultMap>标签的属性id值。

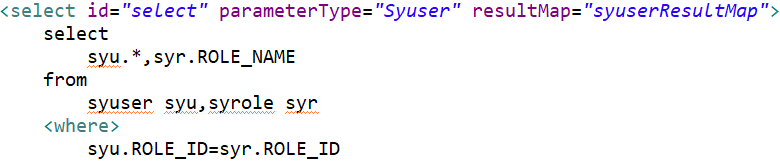
**select属性**：表示使用哪条<select>标签中的sql语句。

**fetchType属性**：指定在关联查询时是否启用延迟加载。该属性有lazy和eager两个属性值。**默认值为lazy(即默认关联映射延迟加载)**。

注意<association>标签只能用在映射对象，不能用在String等类型中。比如A类中有一个属性的值是B类中的String类型的name属性，A类中包含这个name属性，但没有B的实例对象，则不能使用<association>标签进行映射，而是直接使用<result>标签。





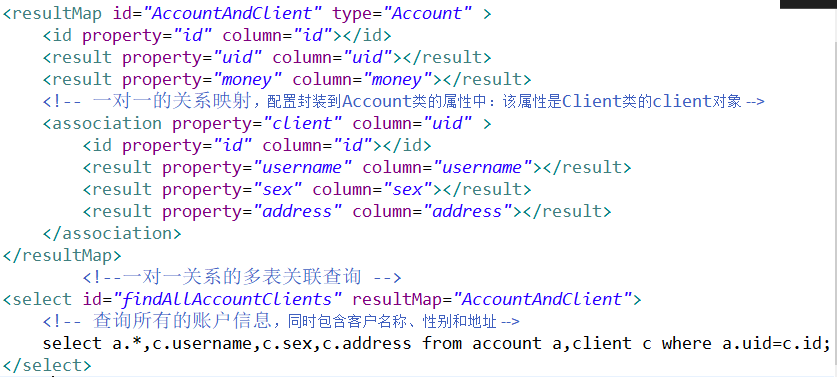


#### ⑵<association>的两种方式

**mybatis中使用<association>标签进行关系映射有两种方式如下**：**一种是在<association>标签中嵌套resultMap，另一种是嵌套select语句**。但不建议使用嵌套select语句，因为这会产生N+1问题。

**注意<association>标签要放在<resultMap>标签内的最后面，不然的话会报错**。

**例1：**



**方式一：嵌套resultMap**

**例1的另一种写法：**



例1和方法一中<association>标签中可以不使用column属性。

这种方式的**特点**是：只会执行一条复杂的sql语句，sql语句编写较复杂。

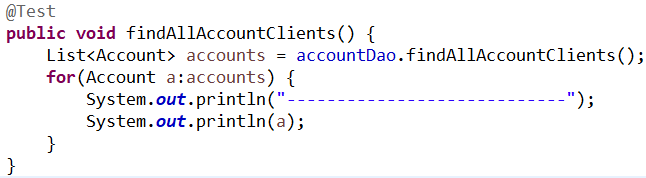
**方式二：嵌套select语句**

这种方式的**特点**是：在查询sql中嵌入一个子查询sql，会执行多条sql语句，且查询的sql语句编写较简单。缺点是会有很多条sql语句被执行，这会极大消耗数据库的性能并降低查询的效率。

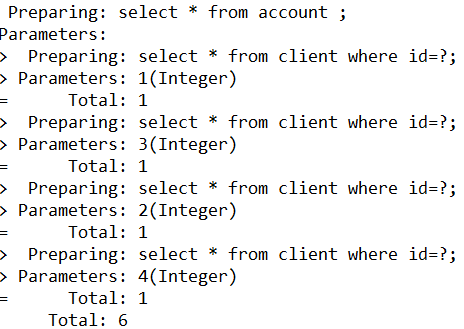
**例1的另一种写法：**

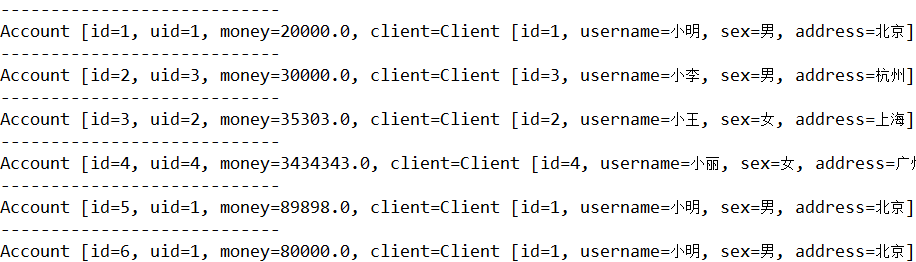


执行测试方法



则mybatis中执行的sql语句过程如下：





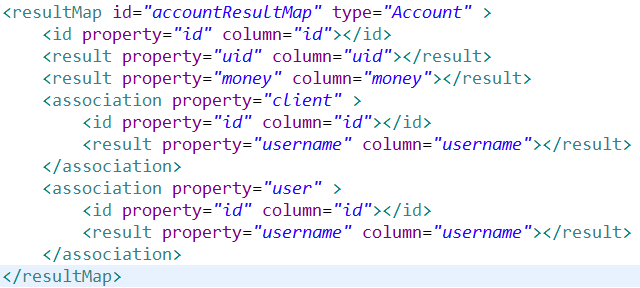
从上面的执行过程中可以看出，在执行完select \* from account语句之后，mybatis会根据联合条件uid自动给select \* from client where id=?赋值。

一个<resultMap>中也可以配置多个<association>标签，如下所示：

①



②



### 7.<collection>标签

在mybatis中，一对多关系映射中，使用<collection>标签。其属性如下：

**property属性**：实体类中属性的名称，这个属性一般是集合类型的，property的值为集合的引用变量名称。注意proerty的值不是List、set、Map。

**ofType属性**：用来指定实体类中的集合属性中的数据类型。如List<T >集合中的T就是用ofType属性指定。若没有起别名则要写全名称。这个同javaType一样可以省略。

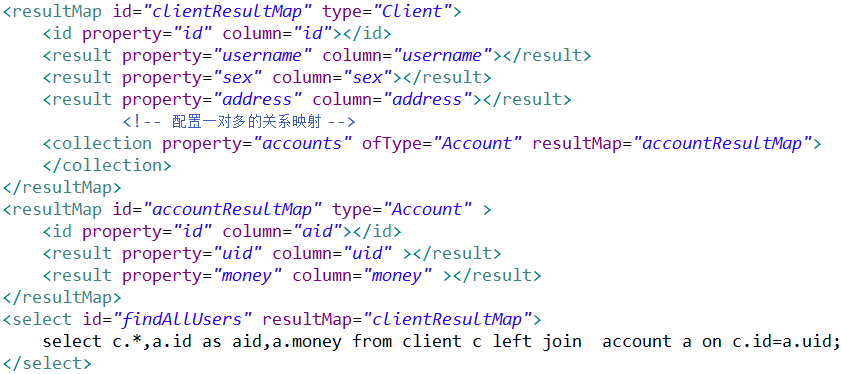
**resultMap属性、select属性、javaType属性**和<association>标签的这些属性的用法一样。

**例1**：

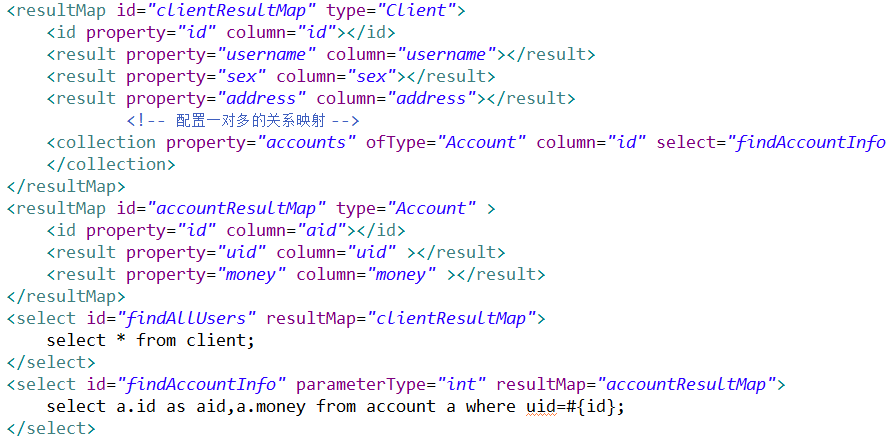


<collection>标签同<association>标签一样，同样有两种方式：

**方法一**：**嵌套resultMap**



**方法二**：**嵌套select语句**



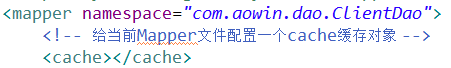
### 8.<cache>和<cache-ref>标签

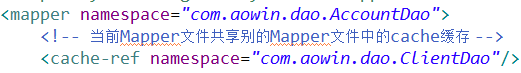
这两个标签是在映射配置文件中用来设置二级缓存的。

<cache></cache>标签是用来为当前的Mapper文件分配一个缓存对象。

<cache-ref namspace=“”>标签，当当前的Mapper文件需要使用别的Mapper文件中的缓存时，使用这个标签配置。namespace属性对应的是别的Mapper文件中<Mapper>标签的namespace属性。

注意一个Mapper文件只能配置一个<cache>或<cache-ref>标签，即只能使用一个cache缓存对象。





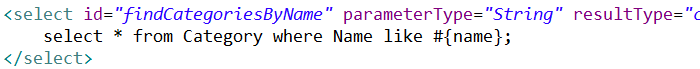
## 九、增删改查(CRUD)操作

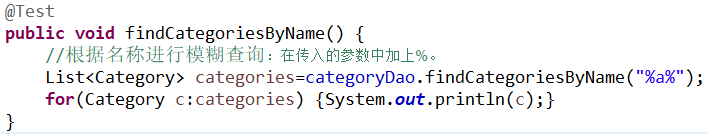
### 1.模糊查询

注意mybatis中进行模糊查询有**三种方式**：

**法一**：**在方法传入的参数中加上%**

在映射配置文件中的sql语句写成下面这种形式：



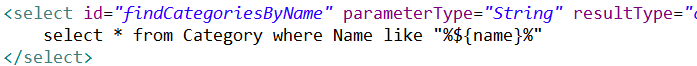


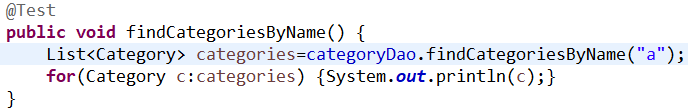
**最终执行的sql语句的形式**如下：



**法二**：采用“**%${value}%**”

**不推荐这种方式**，因为可能会有sql注入的风险。



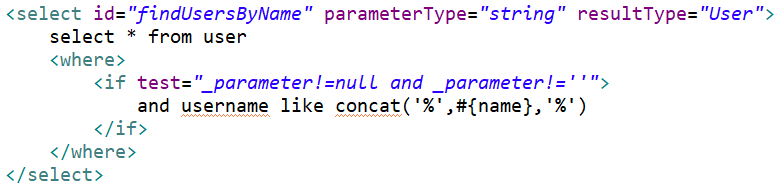


**最终执行的sql语句的形式**如下：



对比两者执行Sql语句的形式，可以发现**法一采用的是PreparedStatement的？问号占位符进行占位，进行预处理，然后再赋值。而美元符法二的方式则是采用是Statement的方式进行字符串的拼接**。

**法三**：采用mysql的**concat()**连接字符串函数。



### 2.新增操作后主键的返回值

执行一条新增操作之后，若还需要获得当前新增操作的主键值，可以采用如下方式获取：

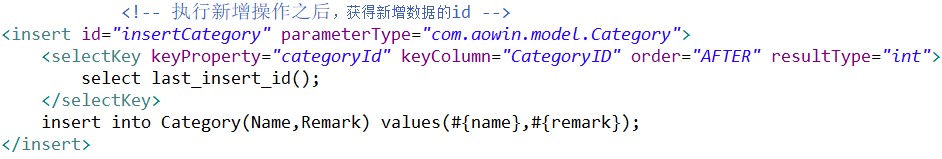
select **last\_insert\_id();**语句可以获得新增操作之后的主键值。

其中<selectKey>标签中各个属性表示什么含义在上面的笔记中写了：[<selectKey>标签中的属性](#_4.<selectKey>标签中的属性)

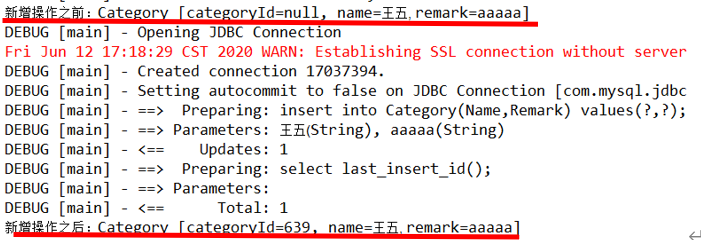
<selectKey>标签中的resultType属性和KeyColumn属性一般可以省略。

**(主键自增)例子1**：

配置映射文件中的sql语句如下：

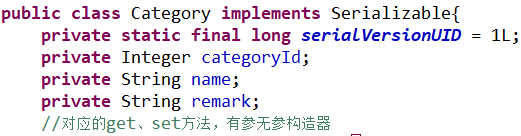




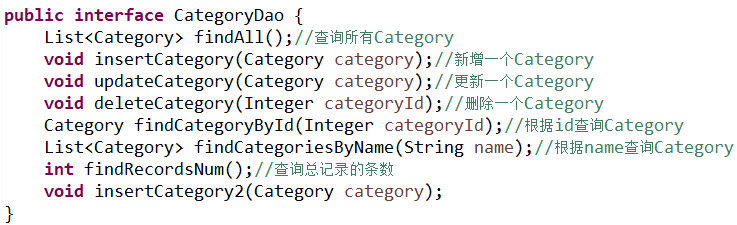


### 3.增删改查操作例子

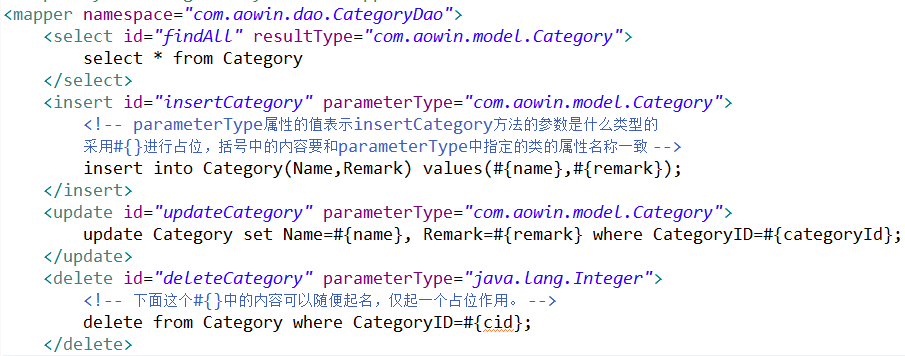
①实体类：



②实体类的Dao接口



③映射配置文件

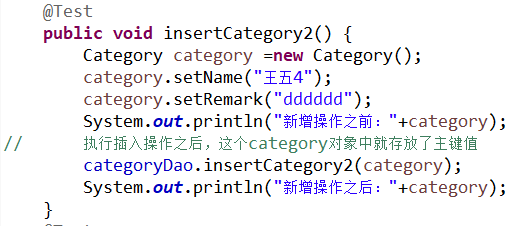


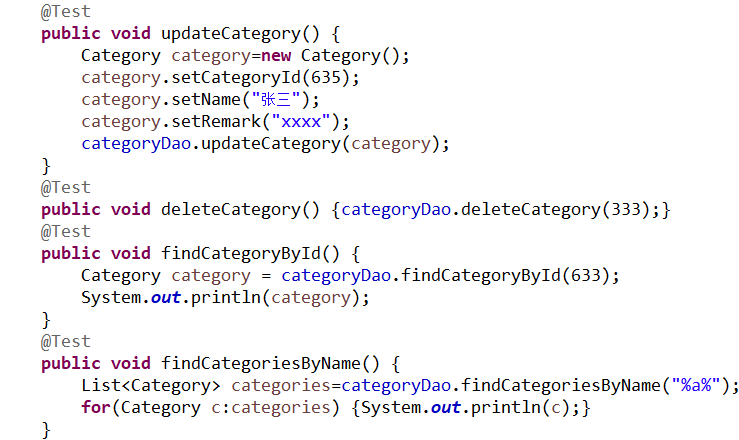


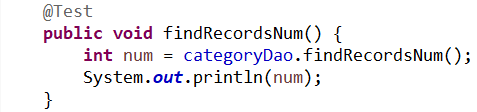
④测试类







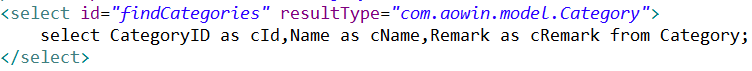




### 4.表的列名和实体类属性不一致的查询

当数据库表的列名和实体类的属性名称不一致时，直接将查询结果的记录封装到实体类中是会报错，解决办法：

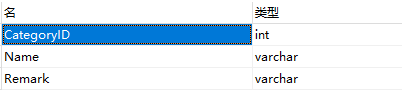
**法一**：在查询语句中，给表的列名取别名，别名的名称和类的属性名一致，这样就能一一对应了。

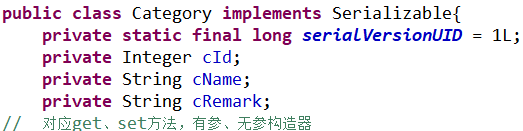


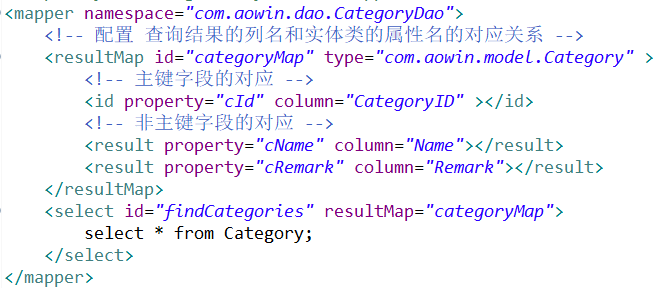
**法二**：采用<resultMap>标签配置查询结果的列名和实体类的属性名的对应关系，然后将<select>标签中的resultType属性改成resultMap属性。

法二的方式执行效率没有法一的高。

**例子**：







在实际开发中，如果追求执行效率，要么在查询sql语句中给表的列名起别名使其和实体类的属性名的对应起来。若要追求开发效率，则要么把实体类的属性名和表的列名写成一致，要么采用<resultMap>方式进行配置。

## 十、OGNL表达式

**OGNL**：Object Graphic Navigation Language 对象图形导航语言。**它是通过对象的取值方法来获取数据，在写法上把get给省略了**。

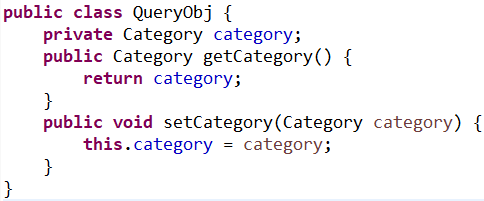
比如：获取用户的名称，在类中的写法是user.getUsername()，而OGNL表达式的写法是user.username，即直接用对象的属性名称。

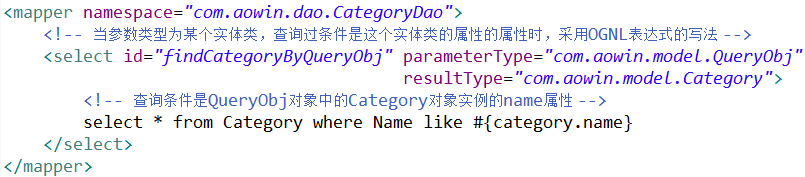
在mybatis中在#{}或${}可以直接写属性名称如username，而不用user.的方式，是因为在parameterType中已经提供了该属性所属的类，所以此时不需要再写对象。

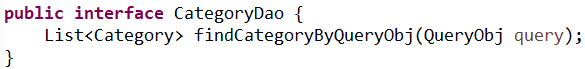
**POJO(Plain Ordinary Java Object)对象**，就是简单的java对象，也就是普通的javaBean对象。

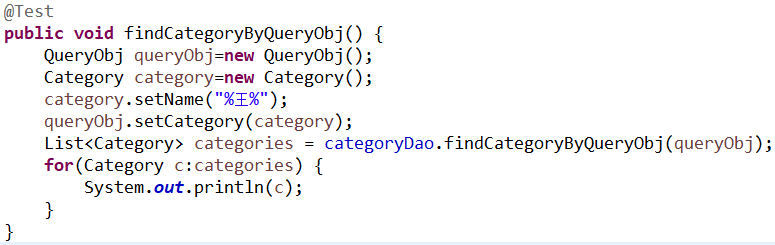
**由多个对象的属性一起组成查询条件，这时可以写一个POJO类用来存放这几个对象，然后将POJO类作为参数进行传递，然后采用OGNL表达式将这几个对象的属性点出来作为查询条件**。

简单例子：在一个类中封装另一个类









## 十一、mybatis中CRUD执行过程分析

### 1.编写dao实现类的CRUD执行过程分析

黑马的视频中讲了，有空自己进行分析。

在采用SqlSession的insert()、delete()方法时，这两个方法实际上是在调用update()方法，最后对数据库进行操作调用的是jdbc中的PreparedStatement的execute()方法进行执行。因此增加、删除操作直接调用SqlSession相应的update()方法也能完成操作。

### 2.代理dao的CRUD执行过程分析

在使用SqlSession的getMapper()方法时创建代理对象的过程实际上内部调用了相应的SqlSession的selectList()、insert()、update()、delete()等方法。

详细过程有空自己进行分析，黑马的视频中讲了。

## 十二、主配置文件中的标签

在主配置文件中有如下的标签，注意使用这些标签的使用顺序：



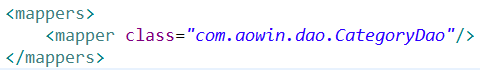
### 1.<mappers>标签

#### ⑴子标签<mapper>

**resource属性**：mybatis中使用的是**xml的方式**，则SqlMapConfig.xml中的配置映射文件的位置时，该属性值为配置映射文件的路径。



**class属性**：采用**注解的方式**，则SqlMapConfig.xml中的配置映射文件的位置时，该属性值为对应接口的全名称。



**url属性**：也可用这个属性配置映射文件的位置，这个用法和下面<properties>标签啊的url属性一样。仅作了解。

#### ⑵子标签<package>

这个标签可以用来指定dao接口所在的包，当指定了这个之后，所有在这个包中的dao接口所对应的映射配置文件，则可以不用在主配置文件中使用<mapper>标签的resource属性或class属性指定这个映射配置文件的位置了。

**注意能使用这个标签自动映射到相应的配置文件的前提是接口和mapper文件在同一个包中，且名称一样**。

**name属性**：指定对应的映射配置文件所在的包。



[接口和映射文件在同一个包下](#_1.接口和映射文件在同一个包下)

### 2.<properties>标签

<properties>标签：可以在<properties>标签内部配置连接数据库的基本信息，也可以通过resource属性或者url属性引用外部配置文件信息。

**resource属性**：用于指定外部配置文件的位置。**resource更常用**。

**url属性**：也可用来指定外部配置文件的位置，但是它必须按照url的写法来写地址。将外部配置文件采用浏览器打开，然后网页上的路径复制到url中即可。**这种方式仅作了解**。

url(Uniform Resource Loctor)：统一资源定位符，它可以唯一标识一个资源的位置。写法是：协议 主机 端口号 URI，如<http://localhost:8080/scmProject/index.jsp>。

uri(Uniform Resource Identifier)：统一资源标识符，它**在应用中**可以唯一定位一个资源。

即url是可以在整个网络中定位资源的，而uri则是只能在项目/应用中定位资源。



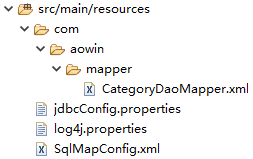
#### ⑴标签内配置连接数据库的信息

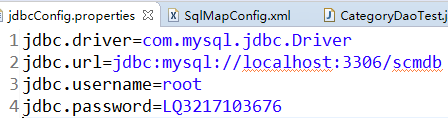
在配置数据源(连接池)的四个基本信息时，支持使用<properties>标签的子标签<property>配置好四个基本信息，然后在然后在数据源的四个基本信息中采用占位符${}占位进行匹配。如下所示：



#### ⑵引用外部配置文件信息

在<properties>标签的属性resource或者url用来指定配置文件的位置，这个配置文件一般和主配置文件放置的位置一样，直接放在src/test/resources目录下。然后在配置数据源的基本信息中采用${}进行占位，这里注意{}括号中的内容填写的是外部配置文件jdbcConfig.properties中的key。其中url的写法必须写协议、主机、端口号、uri等。

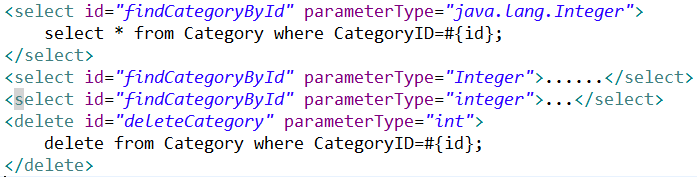






### 3.<typeAliases>标签

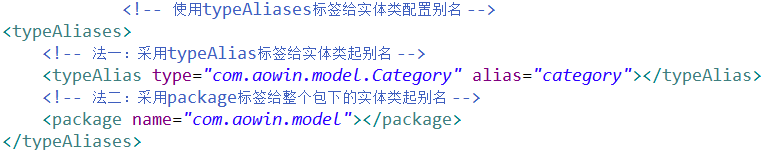
当在映射配置文件中，在<select>、<insert>、<update>、<delete>标签中的parameterType属性的值是基本数据类型、基本数据类型对应的包装类或String类型，可以不用使用全名称，如下的写法都正确：



上面的写法都正确是因为mybatis自动将基本数据类型、基本数据类型对应的包装类或String类型起了别名。

为了能在映射配置文件中的parameterType中让自定义的实体类可以不用使用全限定类名，则也可以给这个实体类取别名。

**自定义实体类的别名有两种方式：**

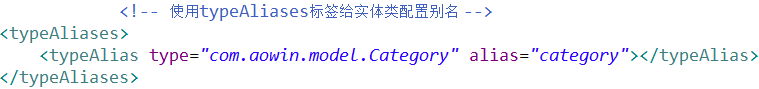


#### ⑴采用<typeAlias>起别名

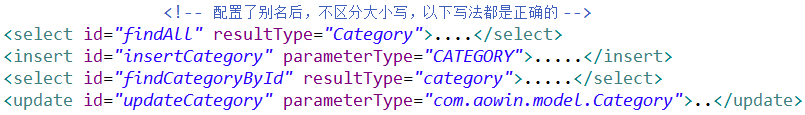
**type属性**：为实体类的全限定类名，表示是给这个实体类起别名。

**alias属性**：这个是给实体类起的别名。**注意当指定了别名，则不再区分大小写**。

在主配置文件SqlMapConfig.xml中配置如下：



在映射配置文件CategoryDao.xml中的使用如下：



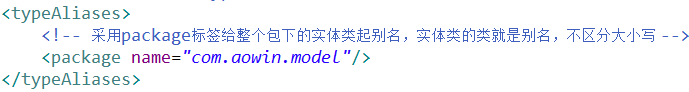
#### ⑵采用<package>起别名

当要给多个实体类起别名，则采用上面<typeAlias>标签的方式则有点麻烦，也可以采用<package>标签起别名。

<package>用于指定要配置别名的包，当指定之后，该包下的所有实体类都会注册别名，并且对应的类名就是别名，不区分大小写。

**name属性**：值为指定要配置别名的包。

在主配置文件SqlMapConfig.xml中配置如下：



则可以在映射配置文件中使用别名。

### 4.<dataSource>标签

该标签用来配置数据源(链接池)。

**type属性**：采用何种连接池的方式。用来配置mybatis中的数据源类型为：POOLED、UNPOOLED、JNDI。

**type属性的取值如下**：

**POOLED** ：采用传统的javax.sql.DataSource规范中的连接池，mybatis中有针对规范的实现。POOLED则是从池中获取一个连接对象，使用完这个对象之后，则又放回连接池中。

**UNPOOLED**：采用传统的获取连接的方式，虽然也实现了javax.sql.DataSource接口，但是没有使用池的思想。UNPOOLED中每次都是使用DriverManager.getConnection()获取新的Connection对象。

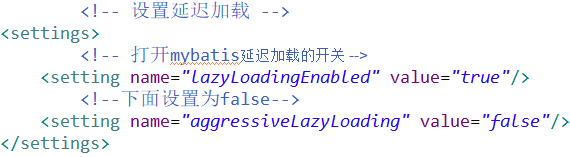
**JNDI**：采用服务器提供的JNDI技术实现，来获取DataSource对象，不同的服务器所能拿到的DataSource是不同的。**注意，若不是web或者maven的war工程，是不能使用的**。tomcat服务器采用的连接池就是dbcp连接池。

### 5.<settings>标签

**下面两个是用来设置延迟加载的。**

**lazyLoadingEnabled**：延迟加载的全局开关。当开启时，所有关联对象都会延迟加载。 **特定关联关系中可通过设置 fetchType 属性来覆盖该项的开关状态**。默认值为false。

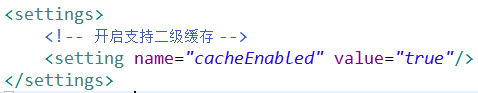
**aggressiveLazyLoading**：开启时，任一方法的调用都会加载该对象的所有延迟加载属性。否则，每个延迟加载属性会按需加载。默认值为false(在 3.4.1 及之前的版本中默认为 true)。



**localCacheScope**：用来设置是否使用一级缓存。 默认值为 SESSION，会缓存一个会话中执行的所有查询。 若设置值为 STATEMENT，本地缓存将仅用于执行语句，对相同 SqlSession 的不同查询将不会进行缓存。



**cacheEnabled**：全局性地开启或关闭所有映射器配置文件中已配置的任何缓存。默认值为true。具体使用：[二级缓存的使用](#_⑵二级缓存)



## 十三、dao接口和mapper文件的存放位置

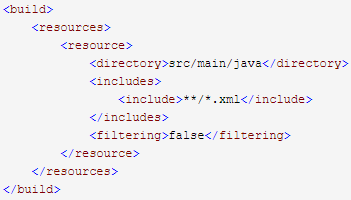
[3.最好遵循如下约束](#_3.最好遵循如下约束)

### 1.接口和映射文件在同一个包下

**在默认的情况下maven打包的时候，对于**src/main/java**目录只打包源代码，而不会打包其他文件**。所以若把对应的mapper文件(即映射配置文件)放到src/main/java目录下时，不会打包到最终的jar文件夹中，也不会输出到target文件夹中，因为在进行单元测试的时候执行的是/target目录下/test-classes下的代码，所以在测试的时候也不会成功。

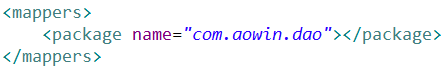
**⑴为了实现在maven默认环境下打包时，Mybatis的接口和mapper文件在同一包中，可以通过将接口文件放在**src/main/java**某个包中，而在**src/main/resources**目录中建立同样的包**，这是一种约定优于配置的方式，这样在maven打包的时候就会将src/main/java和src/main/resources相同包下的文件合并到同一包中。

**⑵**若不想将接口和mapper映射文件分别放在src/main/java和src/main/resources目录下，而是全部放到src/main/java目录下，在构建的时候需要指定maven打包需要包括xml文件，具体配置如下：



这样在打包的时候也会将mapper文件打包到target文件夹下。

当接口和映射文件在同一包下，且名称一致，则能在主配置文件中使用<package>标签中进行自动映射，而不用使用<mapper>标签的resource属性或者class属性进行手动映射配置。[子标签<package>](#_⑵子标签<package>)



### 2.接口和映射文件不在同一个包下

若接口和mapper文件不在同一个包下，就不能使用<package>进行自动映射，而需要在主配置文件中手动使用<mapper>标签的resource属性或者class属性进行映射配置。[子标签<mapper>](#_⑴子标签<mapper>)



## 十四、连接池和事物控制

连接池就是用于存储连接的一个容器，容器其实就是一个集合对象，该集合必须是线程安全的，不能多个线程同时拿到同一个连接。该集合还必须实现队列的特性：先进先出。

**mybatis中的连接池**

主配置文件SqlMapConfig.xml中的<dataSource>标签中进行配置。type属性就是采用何种连接池的方式。[<dataSource>标签](#_4.<dataSource>标签)

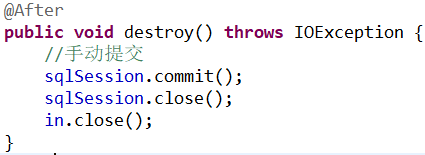
type属性值可以为POOLED、UNPOOLED、JNDI三种。这三种方式在mybatis中的具体运行过程有空再学习，黑马视频将了。

**mybatis中的事物**

在mybatis中，它是通过SqlSession对象的commit()方法和rollback()方法实现事务的提交和回滚。

mybatis中的SqlSession默认不是自动提交事物的。在执行增、删、改操作时，有两种方式让SqlSession提交事物：

**法一**：**sqlSession.commit()手动提交事物**



**法二**：在创建SqlSession对象时，也可在SqlSessionFactory工厂的openSession()方法中设置为true，则表示自动提交事物，则不用再写sqlSesson.commit()。

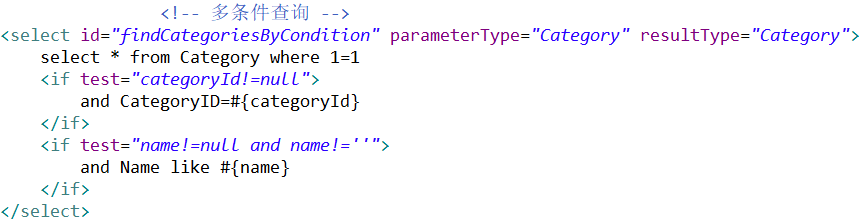


## 十五、基于xml配置的动态SQL语句

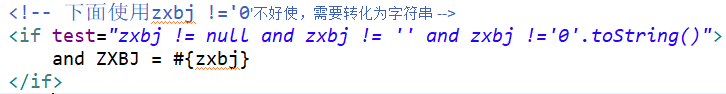
### 1. <if>标签

在查询时，当有未知个查询条件时，可以使用<if>标签进行条件判断，然后再动态拼接sql语句。

用<if>标签的test属性进行条件判断，**注意多个条件之间用and连接，而不能用&&**。



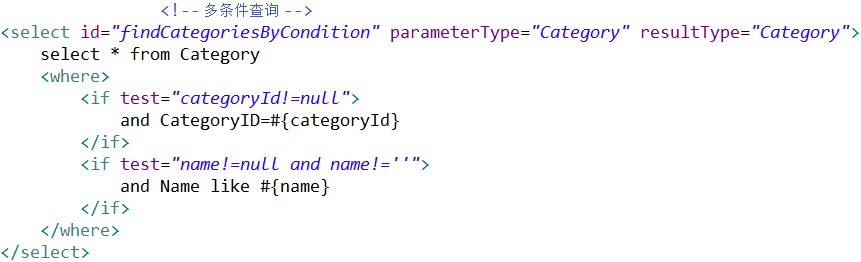
注入当if标签中要判断参数的内容等于或不等于某个数字时，需要将该数字转换成字符串再比较，即使用toString()方法。



### 2.<where>标签

使用<where>标签，mybatis会根据是否有条件自动拼接where关键字，并且检查后面是否有多余的and，如果有则去掉and，然后再拼接后面的条件。

上面<if>标签中的多条件查询也可以修改成下面这种形式。



### 3.<foreach>标签

当sql语句类似select \* from user where id in(?)，问号中的内容是在一个集合中时，可使用<foreach>标签遍历这个集合，并将集合中的数据存放到in()的括号内。

**<foreach>标签的属性如下**：

**collection属性**：代表要遍历的集合元素，**注意在编写时不要写#{}，而是直接写这个集合**。

**open属性**：代表语句的开始部分。

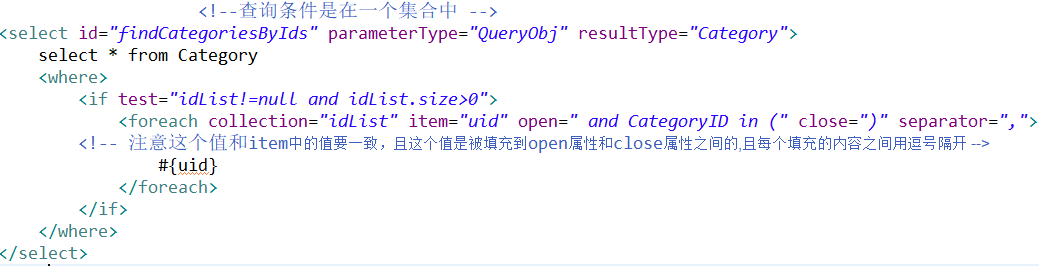
**close属性**：代表语句的结束部分。

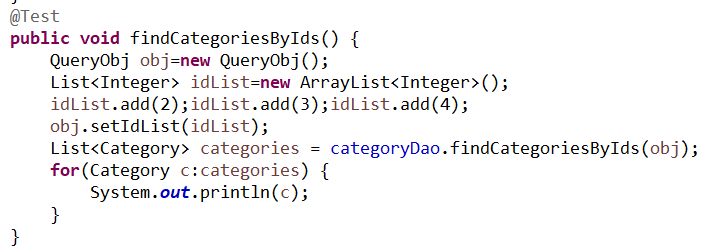
**item属性**：代表遍历的集合中的每个元素。注意要填充到open属性和close属性之间的#{}中的内容要和item属性的值一致。

**separator属性**：代表分隔符。填充到open和close属性之间的内容之间用这个分隔符分隔。

注意各属性之间用空格隔开，而不是逗号。

**例1：**



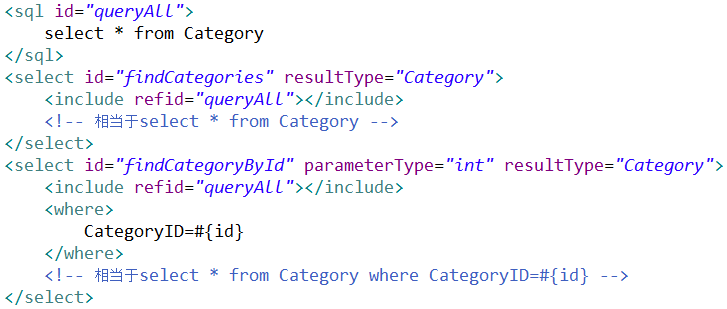




**例2**：[参数类型为List](#_4.参数类型为List)

### 4.<sql>和<include>标签

<sql>标签可以用来抽取重复的sql语句，<sql>中的id属性用来标识这个sql语句。当其它sql语句中需要使用到这个重复的sql语句时，需要使用<include>标签的refid属性引入这个sql语句。

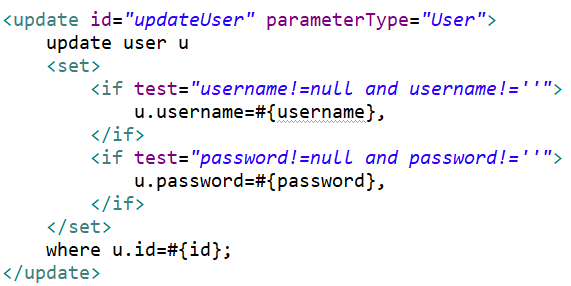


### 5.<set>标签

在mybatis中使用update语句时，若需要更新多个内容，同时这多个内容用<if>标签进行判断，如果某个<if>条件不满足，则会产生多余的逗号，从而导致错误。使用<set>标签可以动态的配置update语句中的set关键字，同时剔除追加到条件末尾的任何不相关的逗号。

**注意在<set>标签中的每个<if>标签内的内容一定不要忘记加逗号**。





### 6.<trim>标签

可用于去除sql语句中多余的and关键字、逗号、或者给sql语句拼接 “where”、“set”以及“values(” 等前缀，或者添加|“)”等后缀，可用于选择性插入、更新、删除或者条件查询等操作。

有空再补充。

## 十六、mybatis中的多表操作

示例：用户表(client)和账户表(account)

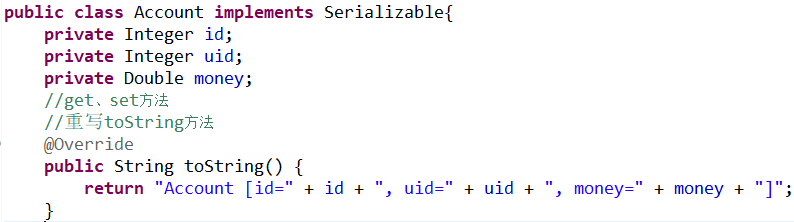
一个用户可以有多个账户，一个账户只能属于一个用户，多个账户可以同属于一个用户。account表的外键uid对应的是client表的id，主表是client表，从表是account表。

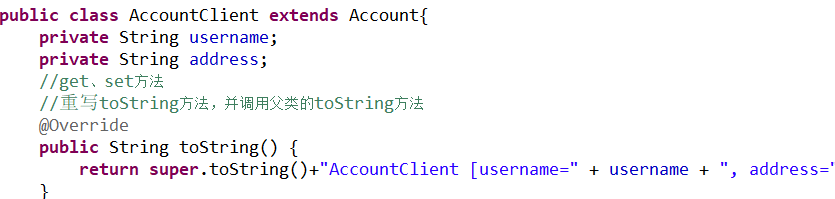
### 1.mybatis中一对一的查询

**例子：**查询所有的账户表中的信息，同时还包含客户名称和地址。

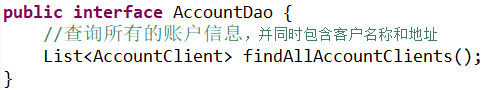
**方法一**：新建一个实体类AccountClient，它继承自Account实体类，子类中声明的属性有username和address，重写toString方法，并调用父类的toString方法。或者在Account实体类中添加属性username和address属性(这种方式感觉还没新建实体类好)。

**实体类**

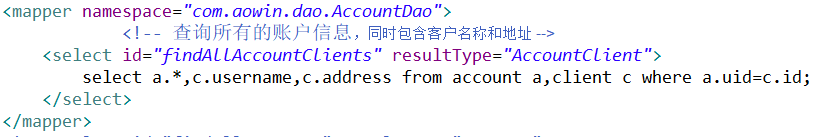




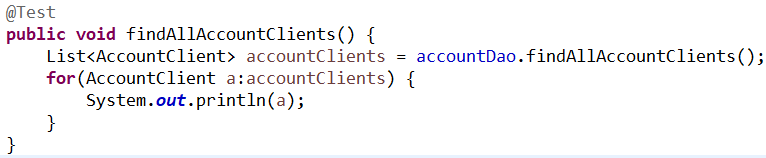
**接口**



**映射配置文件**

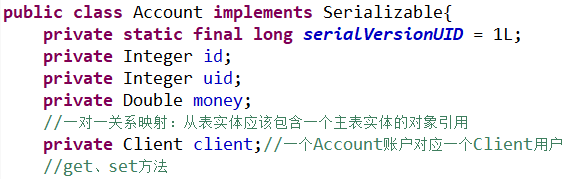


**测试方法**



**方法二**：**在一对一的关系中，将主表实体类封装到从表实体类中**。

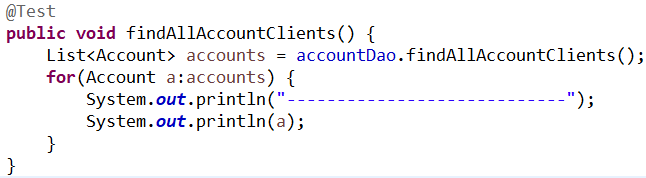
**实体类**

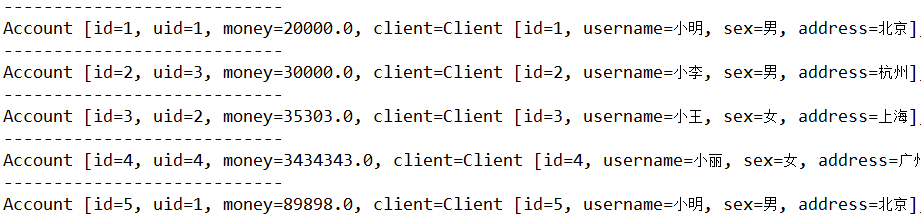


**映射配置文件**



**测试类**



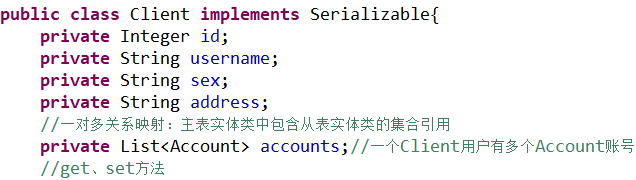


### 2.mybatis中一对多的查询

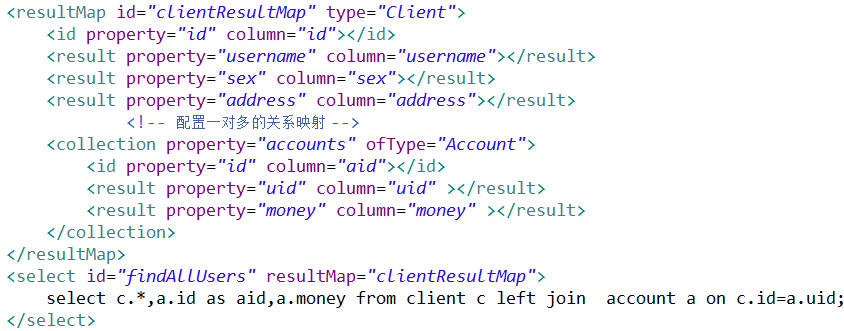
**在一对多的关系中，主表实体类中包含从表实体类的集合引用**。

**例子**：查询所有的用户信息，同时获取到用户下所有的账户信息。

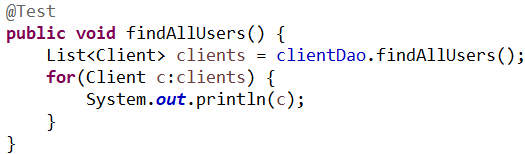
**实体类**



**映射配置文件**



**测试方法**



### 3.mybatis中多对多的查询

**多对多的查询就相当于两个多对一的组合**。

示例：用户表(user)和权限表(systemmodel)

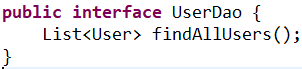
一个用户可以有多个权限，同时每个权限可对应多个用户。

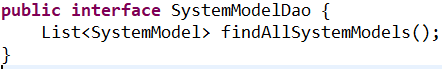
**例子**：**①**当查询用户信息时，同时显示每个用户对应的权限信息。**②**当查询权限信息时，同时显示每个权限对应的用户信息。

**①**select u.\*,sm.\* from user u left join usermodel um on u.id= um.user\_id left join systemmodel sm on um.ModelCode=sm.ModelCode;

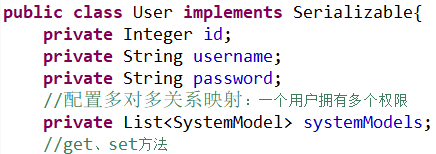
**②**select sm.\*, u.\*from systemmodel sm left join usermodel um on um.ModelCode=sm.ModelCode left join user u on u.id=um.user\_id;

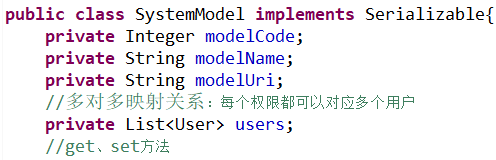
**接口**





**实体类**



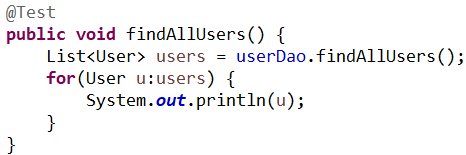


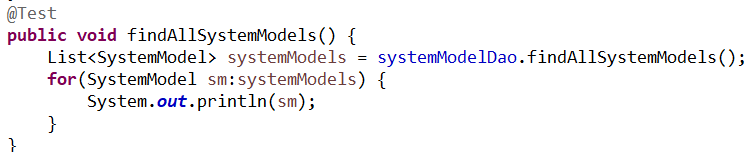
**映射配置文件**





**测试方法**





## 十七、JNDI

JNDI(Java Naming and Directory Interface)：java命名和目录接口，是[SUN公司](https://baike.baidu.com/item/SUN%E5%85%AC%E5%8F%B8)提供的一种标准的Java命名系统接口。目的是模仿windows系统中的注册表，在服务器中注册数据源。

### 1.mybatis中使用采用JNDI配置数据源

**步骤一**：创建一个war类型的项目，然后在webapp目录下新建一个**META-INF**目录，然后新建一个context.xml文件，然后在这个context.xml文件中配置如下信息：**详细看数据源的笔记**：[..\4.javaEE基础\15.数据源.docx](file:///F:\1.java笔记（和盈）\4.javaEE基础\15.数据源.docx)





**步骤二**：在主配置文件中将配置数据源<dataSource>的type属性值改为JNDI。然后在<property>标签中填写相应的内容。

**主配置文件中使用JNDI的方式**

**形式一**：**直接写在内部**



**形式二：写在外部，然后引入**



**主配置文件中使用POOLED的方式**



上面的内容可能还不太完整。

## 十八、mybatis中的延迟加载和缓存

### 1.延迟加载

**延迟加载**：在执行查询操作时，在真正需要使用到该数据时才进行查询，不使用的时候不查询。也称按需加载、懒加载。

**立即加载**：在执行查询操作时，不管是否需要使用到关联表格的数据，都会马上查询出来。

在对应的四种关系种：

**一对多、多对多：通常采用延迟加载**。

**多对一、一对一：通常采用立即加载**。

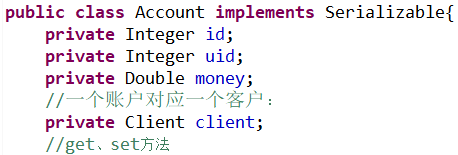
**默认情况下是立即加载**。

在mybatis的主配置文件SqlMapConfig.xml文件中使用<settings>标签设置延迟加载。[<settings>标签](#_5.<settings>标签)

#### ⑴一对一中采用延迟加载的写法

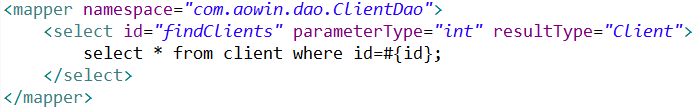
示例：一个账户对应一个客户

**实体类：**

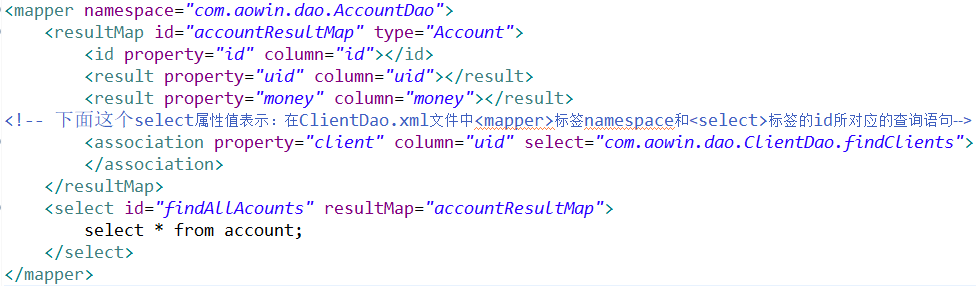


**映射配置文件**

**ClientDao.xml**



**AccountDao.xml**



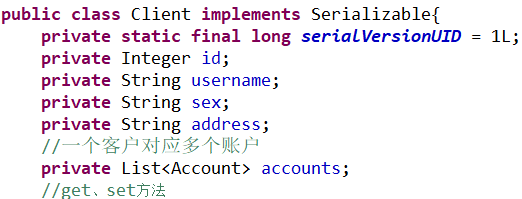
**主配置文件中设置延迟加载的内容**



#### ⑵一对多中采用延迟加载

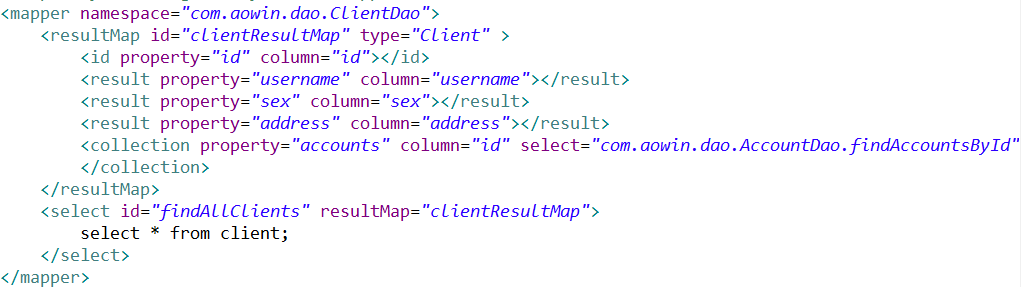
示例：一个客户对应多个账户

**实体类**

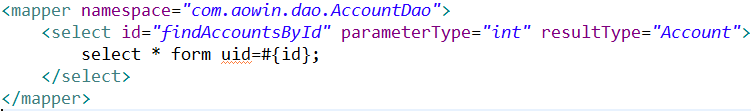


**映射配置文件**

**ClientDao.xml**



**AccountDao.xml**



### 2.缓存

**缓存**：存在于内存中的临时数据。

**使用缓存的好处**：减少和数据库的交互次数，提高执行效率。

**适合使用缓存的数据**：经常查询并且不经常改变的数据，且数据的正确与否对最终结果影响不大。

不适用缓存的数据：经常改变的数据，且数据的正确对最终的结果影响很大。如银行的汇率、商品的库存等。

#### ⑴一级缓存

**mybatis中的一级缓存指的是SqlSession对象的缓存**，在操作数据库时需要构造SqlSession对象，在对象中有一个HashMap用于存储缓存数据。不同的SqlSession之间的缓存数据区域是互相不影响的。

**mybatis中一级缓存默认是开启的**。**一级缓存的作用域是SqlSession范围的，在同一个SqlSession中执行相同的查询SQL，第一次会去数据库进行查询，并写到缓存中，第二次以后是直接去缓存中获取数据，不再去底层数据库查询，从而提高查询效率**。

当SqlSession执行SQL时发生了增删改操作，并且提交到数据库，MyBatis则会清空SqlSession中的一级缓存，这样做的目的是为了保证缓存中存储的是最新的信息，避免出现脏读现象。

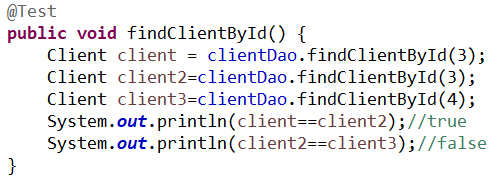
当SqlSession对象执行增删改操作或者调用commit()、close()等方法时，会清空一级缓存。

关闭一级缓存后，再次访问，需要再次获取一级缓存，然后才能查找数据，否则会抛出异常。

一级缓存的范围有**SESSION**和**STATEMENT**。**默认是SESSION**，建议不要修改。当不想使用一级缓存，可以把一级缓存的范围指定为STATEMENT，这样每次执行完一个映射配置文件中的语句后都会将一级缓存清除。在主配置文件中进行如下设置：



**例1：**



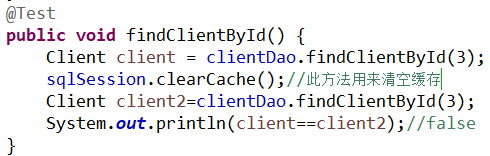
分析上面的结果知道：在默认情况下，即一级缓存开启时，在不关闭SqlSession对象之前，使用同一个SqlSession对象执行相同的查询SQL语句操作且参数也相同的情况下，获得了相同的对象，说明第二次是从缓存中获取数据。

**例2：**



当执行完查询SQL之后，关闭了SqlSession对象，创建新的SqlSession对象，并执行相同的查询SQL语句，且在参数相同的情况下，获得了不同的对象，说明第二次获取的数据不是从缓存中获取的。

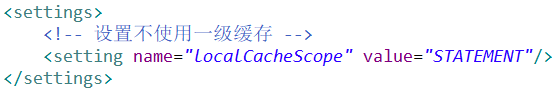
**例3：**

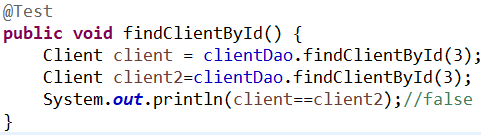


当执行完查询SQL之后，清空缓存，在不关闭SqlSession对象的情况下，执行相同的查询SQL语句，且在参数相同的情况下，获得了不同的对象，说明第二次获取的数据不是从缓存中获取的。

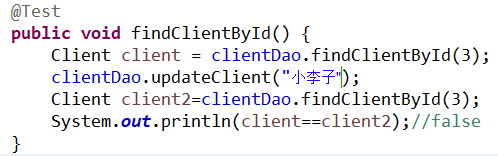
**例4：**

在主配置文件中关闭一级缓存





**例5：**



在同一SqlSession对象下，当第一执行完查询SQL操作之后，若又执行了增删改操作，则原来SqlSession对象存放的缓存会被清空。

#### ⑵二级缓存

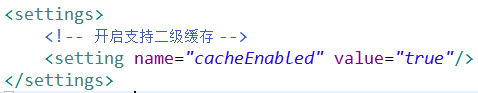
**mybatis中的二级缓存指的是SqlSessionFactory对象的缓存**(或者mapper级别的缓存)。**由同一个SqlSessionFactory对象创建的SqlSession共享其缓存**。

使用二级缓存时，多个SqlSession使用同一个Mapper的sql语句去操作数据库，得到的数据会存在二级缓存区域，它同样是使用HashMap进行数据存储。相比一级缓存SqlSession，二级缓存的范围更大，多个Sqlsession可以共用二级缓存，二级缓存是跨SqlSession的。

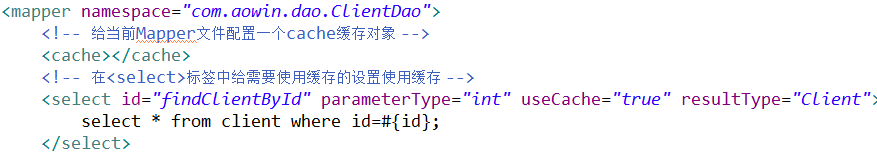
**二级缓存的作用域是同一个SqlSessionFactory对象**(或者 mapper的同一个namespace)。不同的sqlSession多次执行相同的namespace下的查询SQL语句，且向SQL中传递的参数也相同，则第一次执行会将数据库中查询的数据写到缓存，第二次查询会从缓存中获取数据，不再去底层数据库查询，从而提高效率。

**二级缓存的使用步骤**：

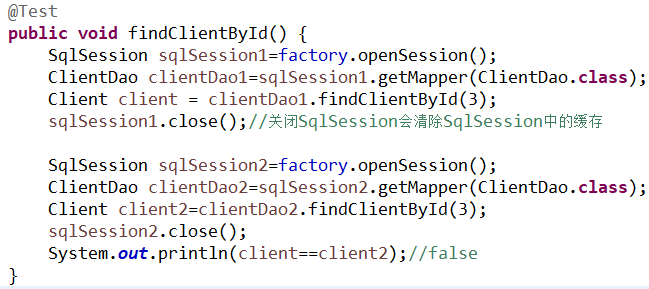
**步骤一**：在Mybatis的主配置文件SqlMapConfig.xml中设置支持二级缓存。

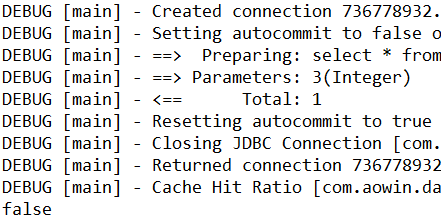


**步骤二**：在映射配置文件(Mapper文件)中使用<cache>标签设置支持二级缓存，同时在<select>标签中使用useCache=”true”表示使用缓存。



**测试：**



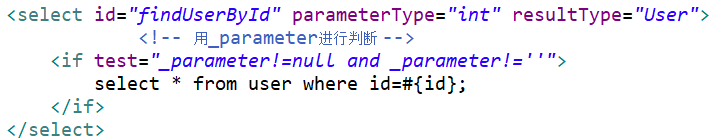


从结果中可以看出，上面只执行了一次查询操作。配置了二级缓存，在同一个SqlSessionFactory对象中，使用不同的SqlSession对象执行相同的查询SQL操作，在参数相同的情况下，第二次是从缓存中获取数据。结果是false是因为二级缓存中存放的内容是数据，而不是对象。

## 十九、mybatis中的传参

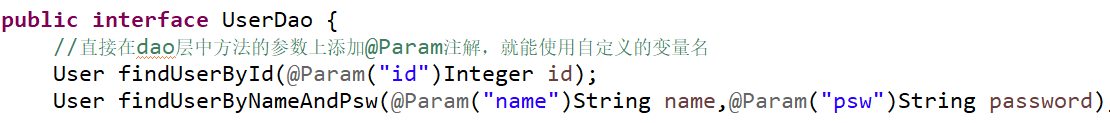
### 1.\_parameter

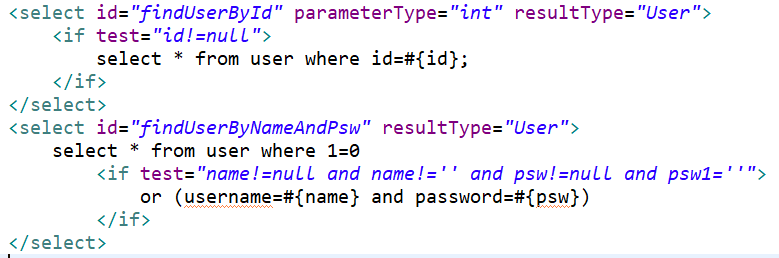
在执行SQL语句之前，若要先对获取的参数进行判断，则不能直接在<if>标签中直接用变量名进行判断，因为mybatis中默认变量名为\_parameter，因此需要**用\_parameter进行判断**，否则会报no getter/setter方法。



### 2.@Param

若不想在判断时使用mybatis默认的变量名，则要在dao层中传入的参数上**添加@Param注解**，则可以使用自己定义的变量名。

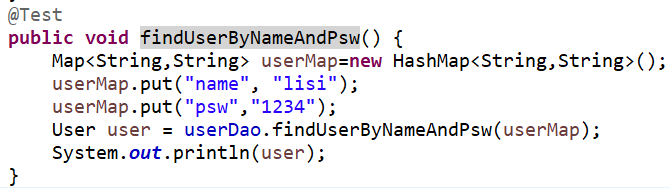


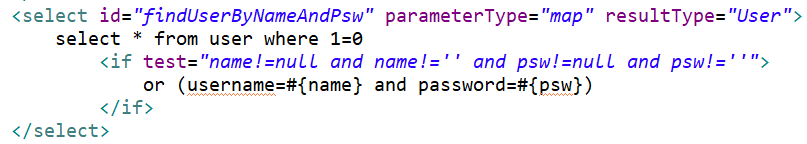


### 3.参数类型为Map(重要)

当传入的参数类型是Map时，则parameterType=‘map’，在sql语句中直接**取key作为传入参数的变量名**，不需要使用@Param或者\_parameter定义的参数变量名。当参数比较多时，可以采用这种方式，从而避免写多个@Param注解。一般多个String类型的参数，可以放在一个Map中作为参数传入mybatis中。**#{key对应的变量名}**获得值。



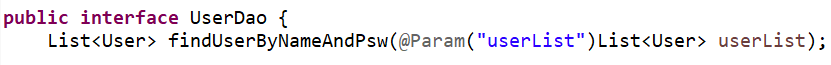


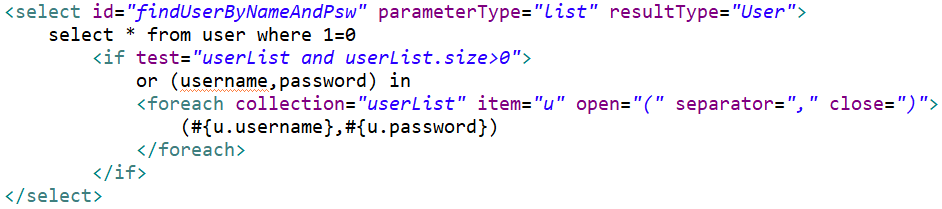


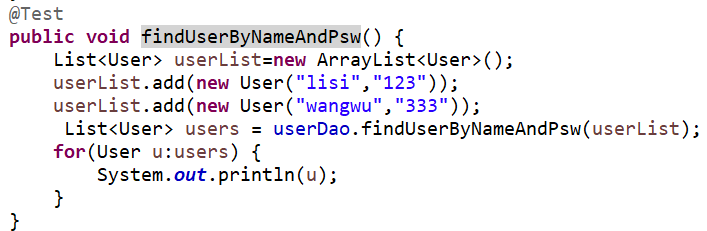
### 4.参数类型为List

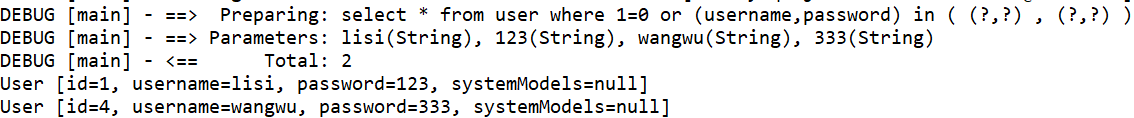
当传入的参数类List集合时，parameterType=”list”。可以使用@Param或者\_parameter定义的参数变量名，也可以直接使用list。然后使用<foreach>标签遍历集合。当List集合中存放的是对象时，要使用对象的属性作为条件时，使用item的值.属性名即可。当List集合中存放的是Map时，也可用item的值.Map中的key值。

**例1(使用@Param)**：



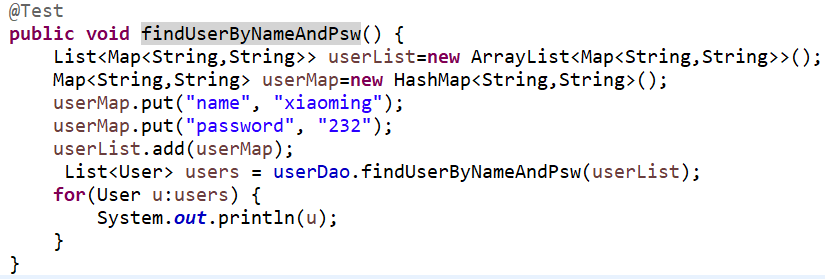


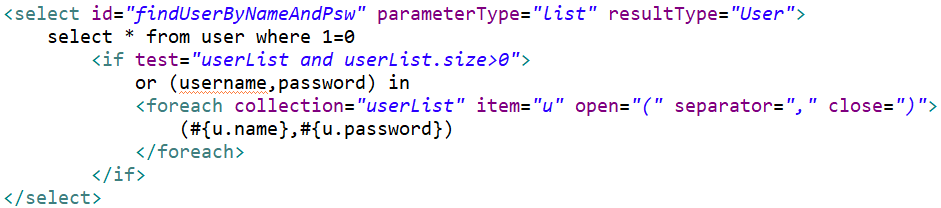




**例2(List中存放的Map)**：

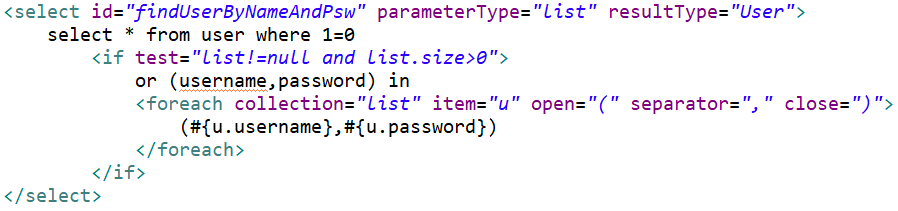






**例3(直接使用list)**：

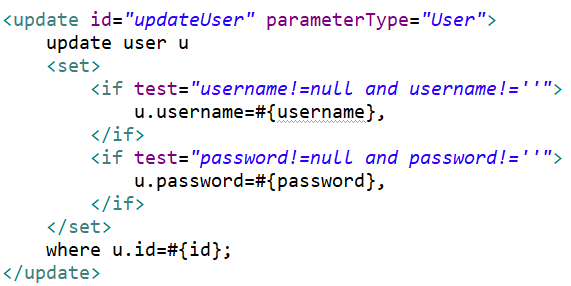


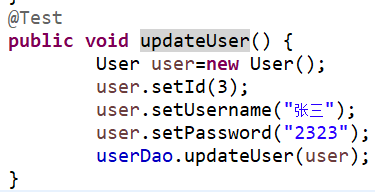


### 5.参数类型为实体类

当参数类型为实体类时，在<if>标签中可以直接使用实体类的属性名进行判断，在#{}内也是填实体类的属性名。注意这个实体类的属性必须有get、set方法。

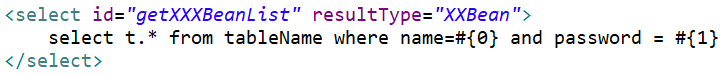






### 6.参数有多个，且为不同类型

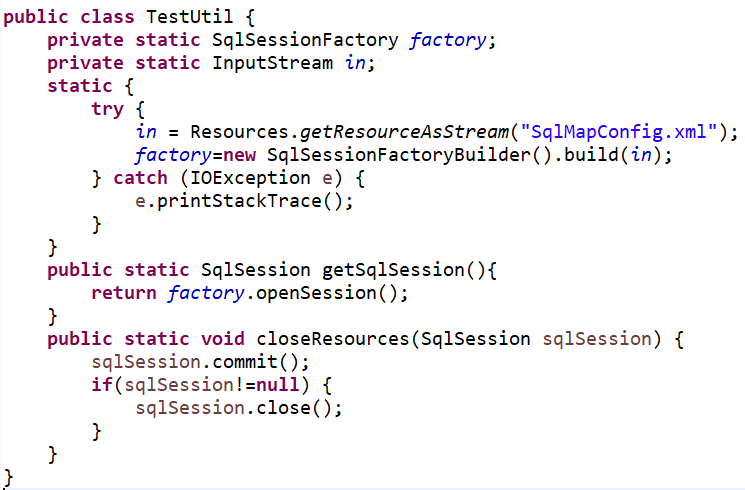




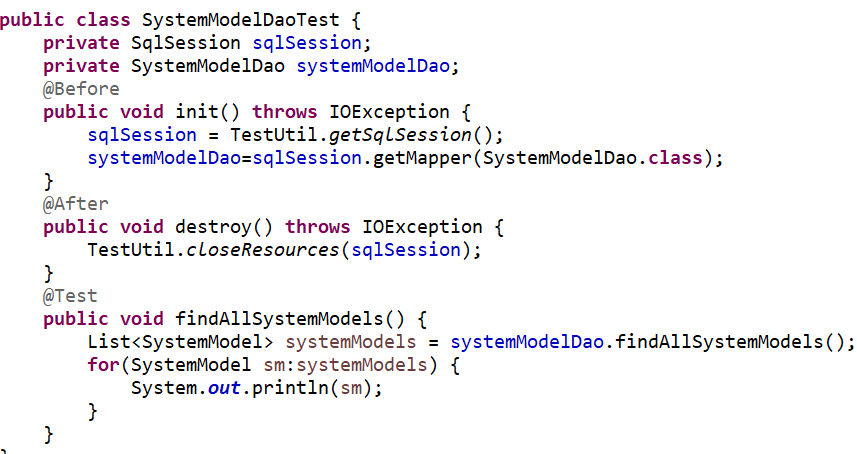
对于多个参数，且参数类型不同，则可以不使用ParameterType，同时使用#{index}索引的方式获取第几个参数。若有多个参数，最好使用Map。

## 二十、mybatis中的测试代码优化

**测试工具类**



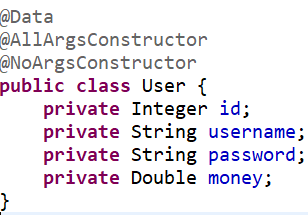
**测试类**



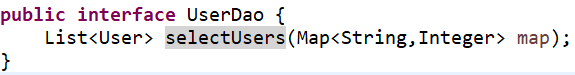
## 二十一、mybatis中的分页查询

### 1. 自己编写代码实现分页

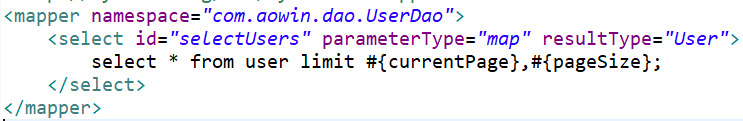
**实体类**：



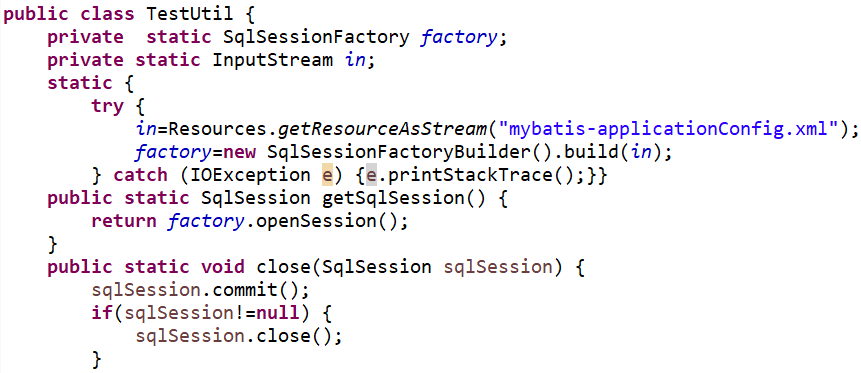
**dao接口**：



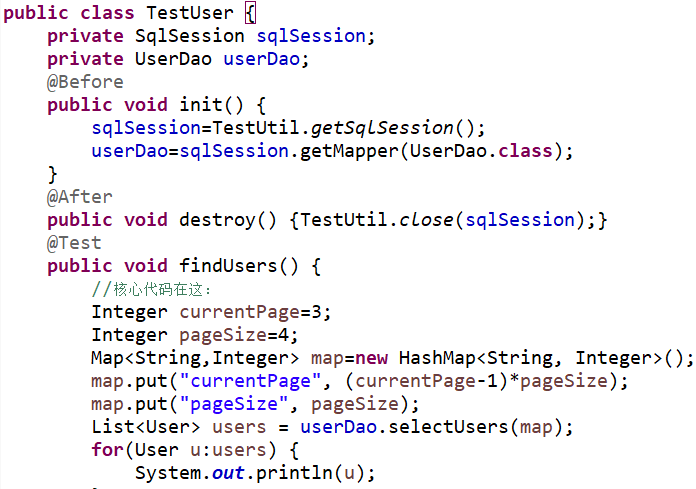
**映射配置文件**：



**测试工具类**：



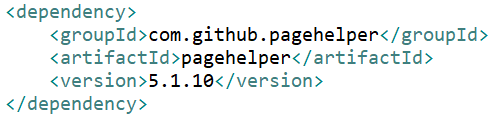
**测试类**：



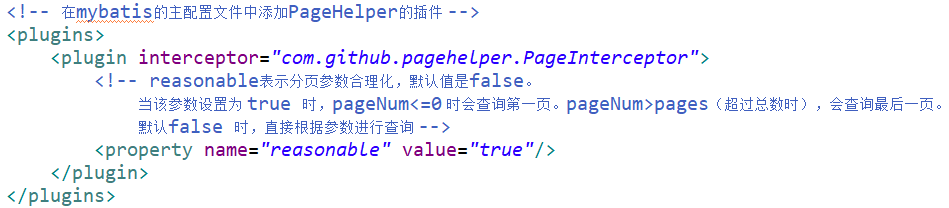
### 2.使用分页插件

此处使用的是PageHelper插件。

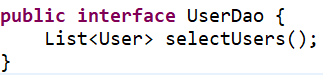
**添加PageHelper的依赖**：

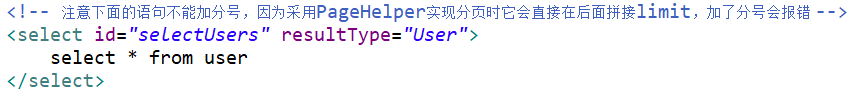


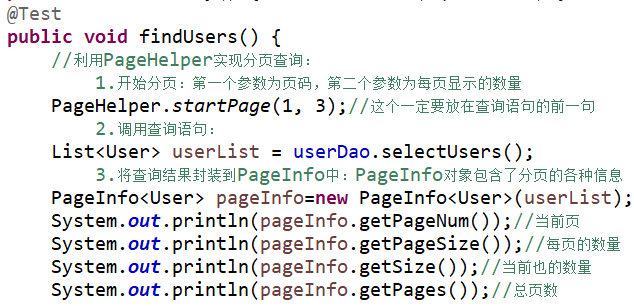
**在mybatis的主配置文件中配置分页插件**：



使用了分页插件之后，sql中就不用再写limit，PageHelper会自己处理分页。









## 二十二、其他内容

在连接数据库的url后面可以拼接一些参数。

如拼接useUnicode=true&characterEncoding=utf8表示按指定字符的编码、解码格式。当没有在mysql的配置文件中修改数据库的编码格式时，默认使用的是jdk编码。mysql使用的是jdk编码，项目中使用utf-8编码，因此存取数据的过程可能乱码。可以拼接useUnicode=true&characterEncoding=utf8到url后面，这样在存数据时，数据库在存放项目数据的时候会先用UTF-8格式将数据解码成字节码，然后再将解码后的字节码重新使用GBK编码存放到数据库中。取数据时在从数据库中取数据的时候，数据库会先将数据库中的数据按GBK格式解码成字节码，然后再将解码后的字节码重新按UTF-8格式编码数据，最后再将数据返回给客户端。

在url后面拼接useSSL=true，表示当使用JDBC和MySQL相连时，由于MySQL的版本更高一些，所以可能导致JDBC和MySQL不兼容，在url的后面添加useSSL=true，则可以连接更高版本的MySQL。

如果使用的是MySQL8.0+，则还要在路径后面添加一个时区的配置，如serverTimezone=Asia/Shanghai。

