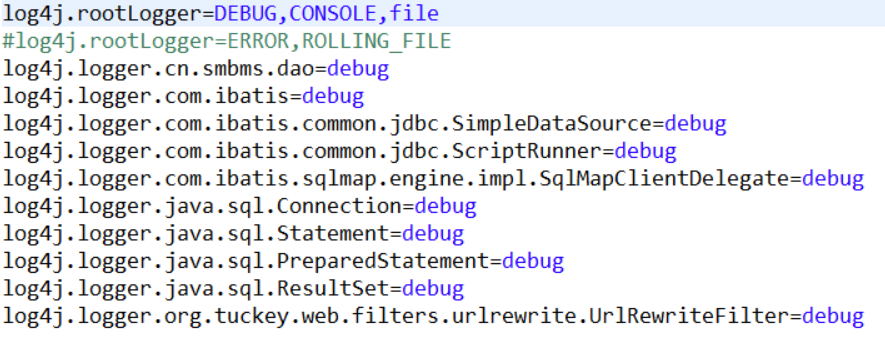
# 第一章:初始mybatis

## mybatis框架

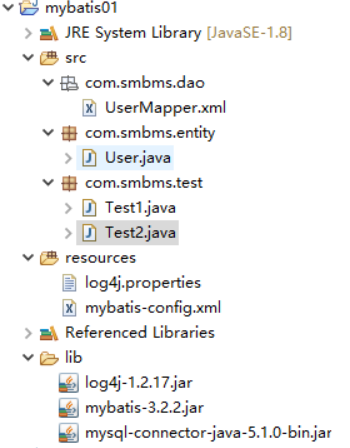
MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录

## 入门程序

### log4j.properties



### 项目结构



### mybatis-config.xml

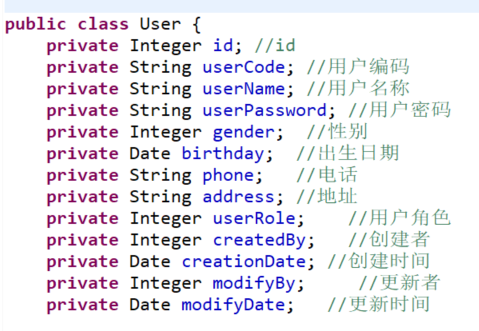
配置mybatis的运行环境，数据源、事务等。



### 编写程序

### 根据用户名模糊查询

#### 创建实体类

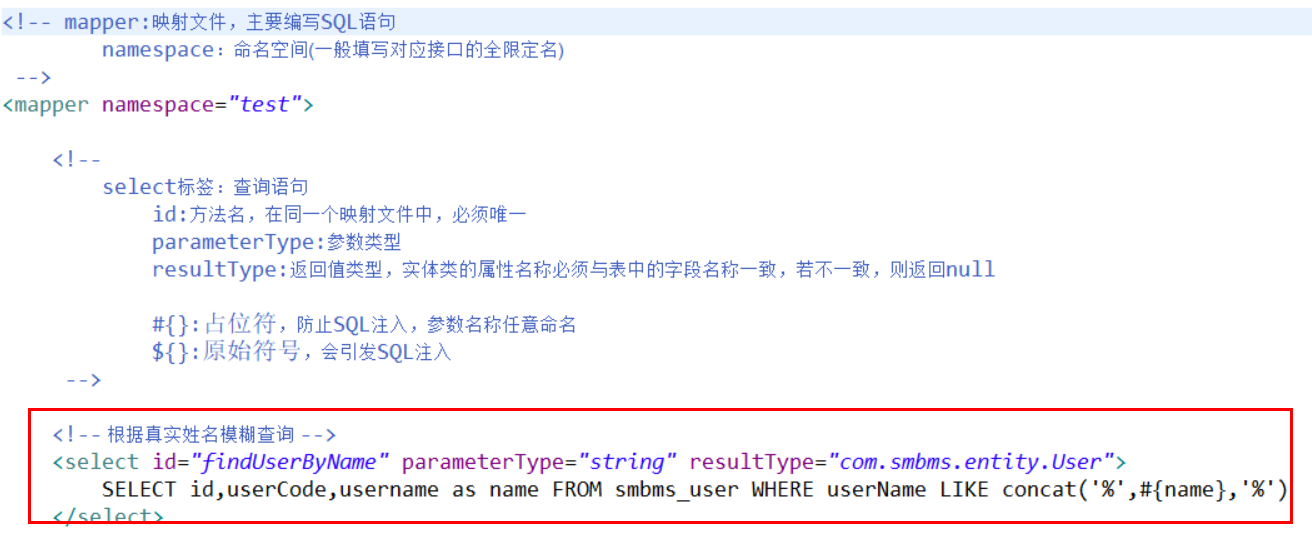


#### 映射文件

映射文件命名：

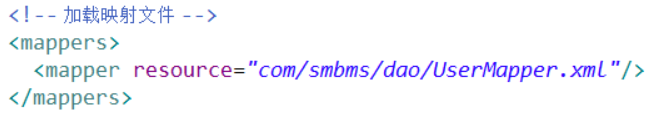
User.xml（原始ibatis命名），mapper代理开发映射文件名称叫XXXMapper.xml，比如：UserMapper.xml、ItemsMapper.xml

在映射文件中配置sql语句。



#### 在mybatis-config.xml加载映射文件

在mybatis-config.xml中加载User.xml:



### 总结

#### parameterType

在映射文件中通过parameterType指定输入参数的类型。

#### resultType

在映射文件中通过resultType指定输出结果的类型。

#### #{}和${}

#{}表示一个占位符号，#{}接收输入参数，类型可以是简单类型，pojo、hashmap。

如果接收简单类型，#{}中可以写成value或其它名称。

#{}接收pojo对象值，通过OGNL读取对象中的属性值，通过属性.属性.属性...的方式获取对象属性值。

${}表示一个拼接符号，会引用sql注入，所以不建议使用${}。

${}接收输入参数，类型可以是简单类型，pojo、hashmap。

如果接收简单类型，${}中只能写成value。

${}接收pojo对象值，通过OGNL读取对象中的属性值，通过属性.属性.属性...的方式获取对象属性值。

#### selectOne和selectList

selectOne表示查询出一条记录进行映射。如果使用selectOne可以实现使用selectList也可以实现（list中只有一个对象）。

selectList表示查询出一个列表（多条记录）进行映射。如果使用selectList查询多条记录，不能使用selectOne。

如果使用selectOne报错：

org.apache.ibatis.exceptions.TooManyResultsException: Expected one result (or null) to be returned by selectOne(), but found: 4

## mybatis开发dao的方法

### SqlSession使用范围

#### SqlSessionFactoryBuilder

通过SqlSessionFactoryBuilder创建会话工厂SqlSessionFactory

将SqlSessionFactoryBuilder**当成一个工具类使用即可**，不需要使用单例管理SqlSessionFactoryBuilder。

在需要创建SqlSessionFactory时候，只需要new一次SqlSessionFactoryBuilder即可。

#### SqlSessionFactory

通过SqlSessionFactory创建SqlSession，使用单例模式管理sqlSessionFactory（工厂一旦创建，使用一个实例）。

将来mybatis和spring整合后，使用单例模式管理sqlSessionFactory。

#### SqlSession

SqlSession是一个面向用户（程序员）的接口。

SqlSession中提供了很多操作数据库的方法：如：selectOne(返回单个对象)、selectList（返回单个或多个对象）、。

SqlSession是线程不安全的，在SqlSesion实现类中除了有接口中的方法（操作数据库的方法）还有数据域属性。

**SqlSession最佳应用场合在方法体内，定义成局部变量使用。**

## mapper代理方法（只需要mapper接口（相当 于dao接口））

### 思路（mapper代理开发规范）

程序员还需要编写mapper.xml映射文件

程序员编写mapper接口需要遵循一些开发规范，mybatis可以自动生成mapper接口实现类代理对象。

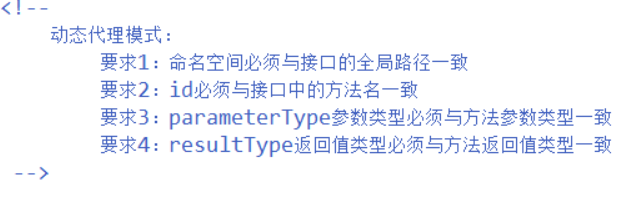
开发规范：

1、在mapper.xml中namespace等于mapper接口地址

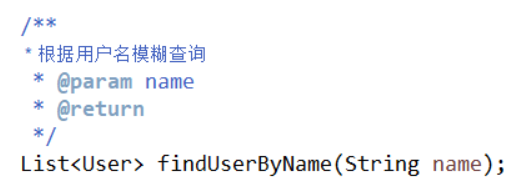
2、mapper.java接口中的方法名和mapper.xml中statement的id一致

3、mapper.java接口中的方法输入参数类型和mapper.xml中statement的parameterType指定的类型一致。

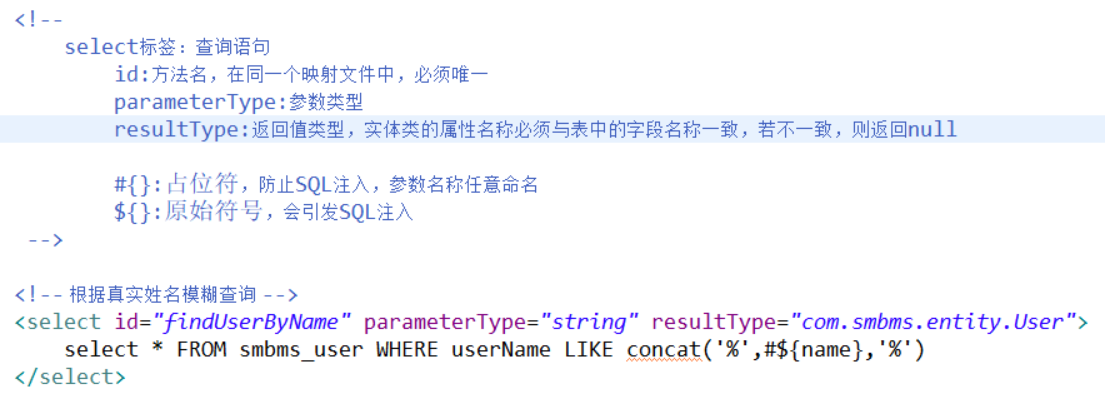
4、mapper.java接口中的方法返回值类型和mapper.xml中statement的resultType指定的类型一致。



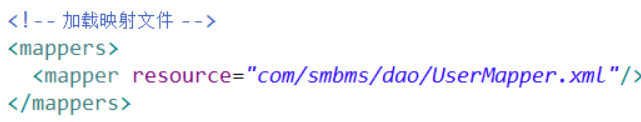
### mapper.java



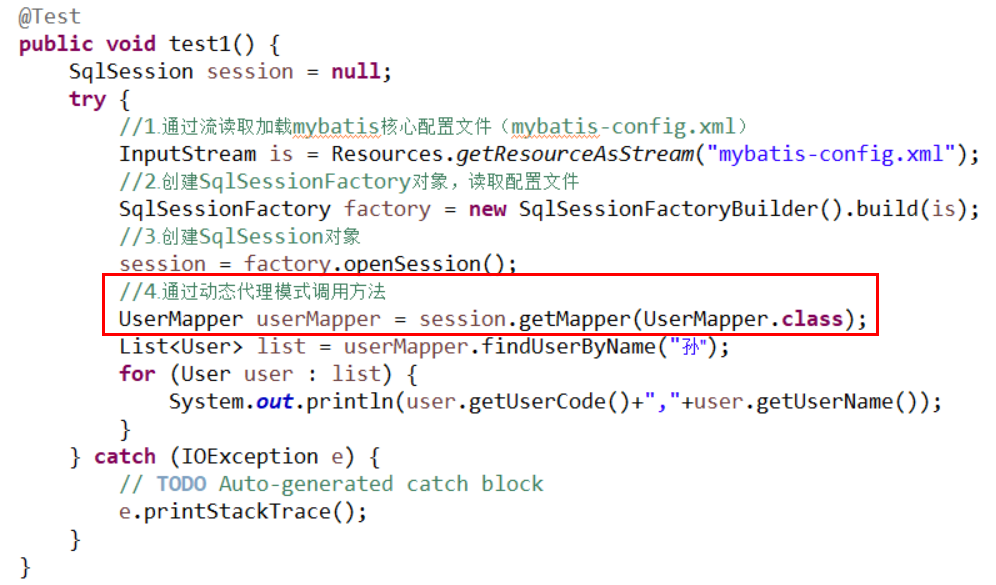
### mapper.xml



### 在mybatis.xml中加载mapper.xml



### 测试



## mybatis-config.xml

mybatis的全局配置文件SqlMapConfig.xml，配置内容如下：

properties（属性）

settings（全局配置参数）

typeAliases（类型别名）

typeHandlers（类型处理器）

objectFactory（对象工厂）

plugins（插件）

environments（环境集合属性对象）

environment（环境子属性对象）

transactionManager（事务管理）

dataSource（数据源）

mappers（映射器）

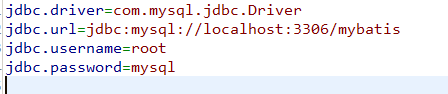
### properties属性

需求：

将数据库连接参数单独配置在db.properties中，只需要在mybatis中加载db.properties的属性值。

在mybatis-config.xml中就不需要对数据库连接参数硬编码。

将数据库连接参数只配置在db.properties中，原因：方便对参数进行统一管理，其它xml可以引用该db.properties。



在mybatis-config.xml加载属性文件：

properties特性：

注意： MyBatis 将按照下面的顺序来加载属性：

* 在 properties 元素体内定义的属性首先被读取。
* 然后会读取properties 元素中resource或 url 加载的属性，它会覆盖已读取的同名属性。
* 最后读取parameterType传递的属性，它会覆盖已读取的同名属性。

建议：

不要在properties元素体内添加任何属性值，只将属性值定义在properties文件中。

在properties文件中定义属性名要有一定的特殊性，如：XXXXX.XXXXX.XXXX

### settings全局参数配置

mybatis框架在运行时可以调整一些运行参数。

比如：开启二级缓存、开启延迟加载。。

全局参数将会影响mybatis的运行行为。

详细参见“学习资料/mybatis-settings.xlsx”文件







### typeAliases（别名）重点

### 需求

在mapper.xml中，定义很多的statement，statement需要parameterType指定输入参数的类型、需要resultType指定输出结果的映射类型。

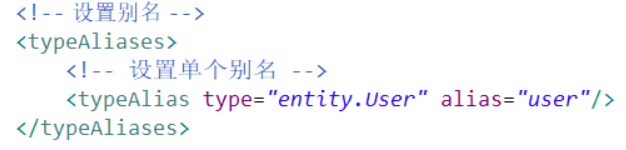
如果在指定类型时输入类型全路径，不方便进行开发，可以针对parameterType或resultType指定的类型定义一些别名，在mapper.xml中通过别名定义，方便开发。

### mybatis默认支持别名

|  |  |
| --- | --- |
| 别名 | 映射的类型 |
| \_byte | byte |
| \_long | long |
| \_short | short |
| \_int | int |
| \_integer | int |
| \_double | double |
| \_float | float |
| \_boolean | boolean |
| string | String |
| byte | Byte |
| long | Long |
| short | Short |
| int | Integer |
| integer | Integer |
| double | Double |
| float | Float |
| boolean | Boolean |
| date | Date |
| decimal | BigDecimal |
| bigdecimal | BigDecimal |

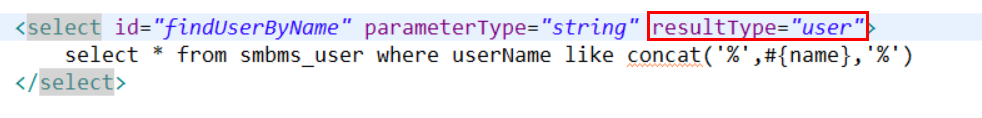
### 自定义别名

#### 单个别名定义



type属性为该类的全局路径，alias为别名

引用别名：



#### 批量定义别名（常用）



### environments



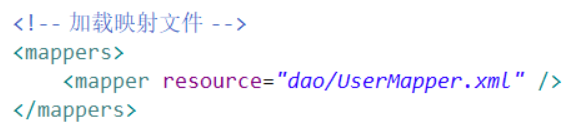
environments配置数据库环境，default属性用于指定具体配置的环境

environment配置单个的环境

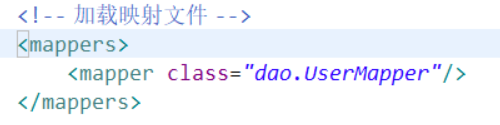
transactionManager为事务管理器

### mappers（映射配置）

#### 通过resource加载单个映射文件

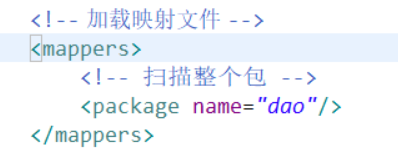


#### 通过mapper接口加载单个mapper



按照上边的规范，将mapper.java和mapper.xml放在一个目录 ，且同名。

### 批量加载mapper(推荐使用)



# 第二章：SQL映射文件

## mapper映射文件

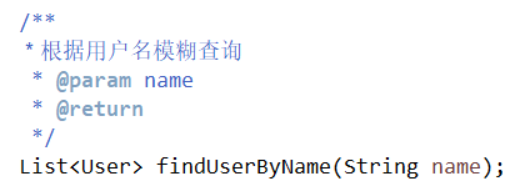
### namespace

命名空间必须指定，一般填写对应接口的全局路径，如果是动态代理模式，必须填写对应接口的全局路径

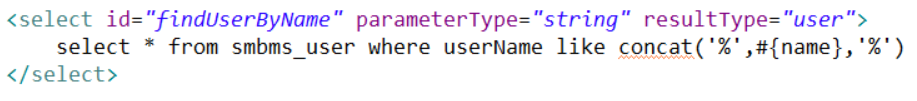
### select

#### 根据用户名模糊查询（单参）

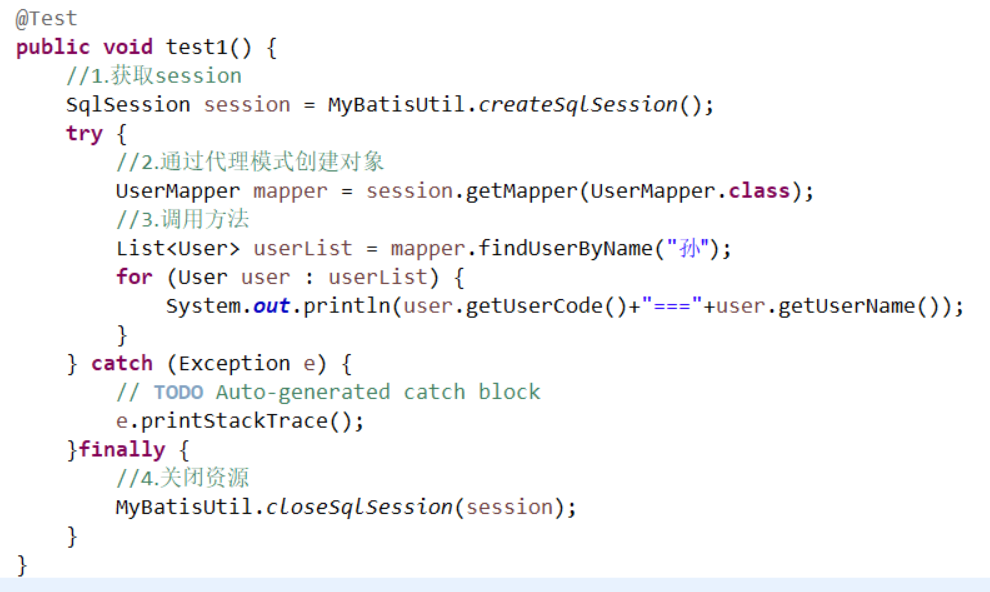
##### Mapper接口



##### 映射文件



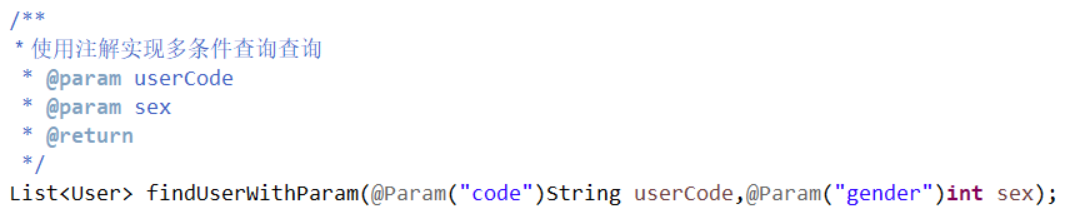
##### 测试代码



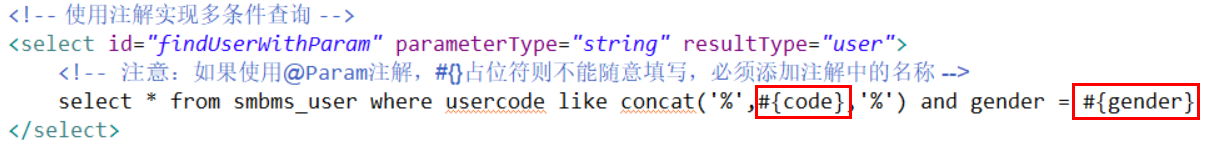
#### 多条件查询（多参）需求：根据用户编码、性别条件查询

##### 使用注解@param

###### Mapper接口

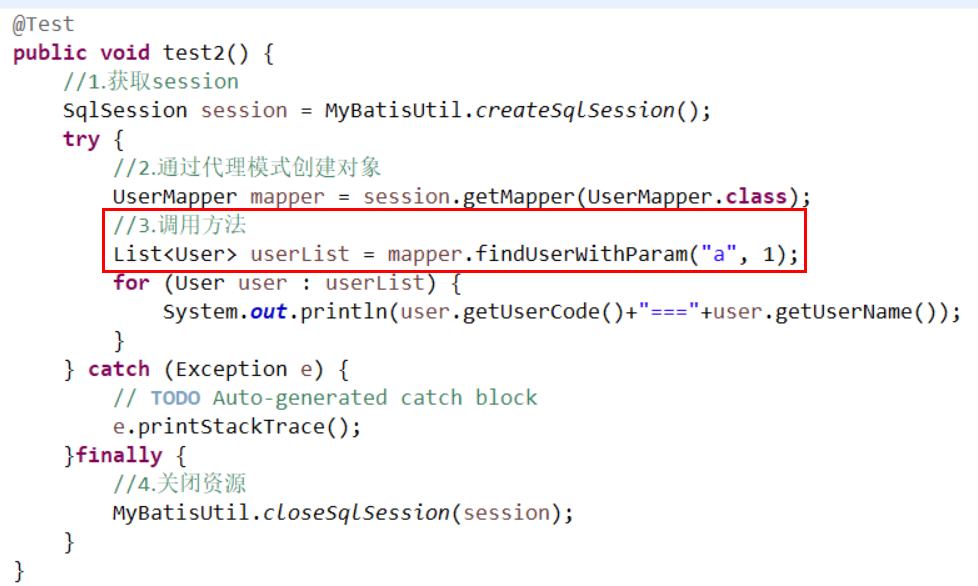


###### 映射文件



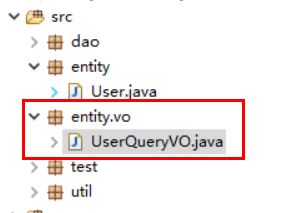
注意：如果使用@Param注解，#{}占位符则不能随意填写，必须添加注解中的名称

###### 测试代码



##### 使用实体类

###### 新建查询自定义实体类

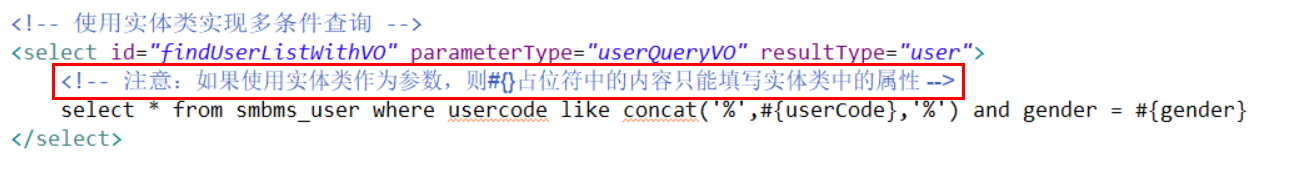




###### Mapper接口

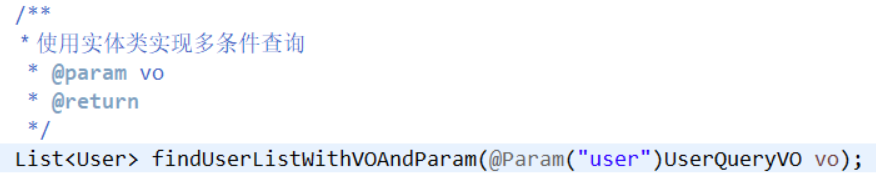


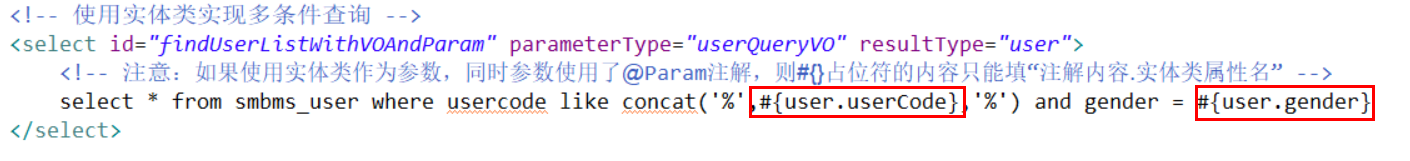
###### 映射文件



注意：如果使用实体类作为参数，则#{}占位符中的内容只能填写实体类中的属性；

如果使用实体类作为参数，同时参数使用了@Param注解，则#{}占位符的内容只能填“注解内容.实体类属性名”



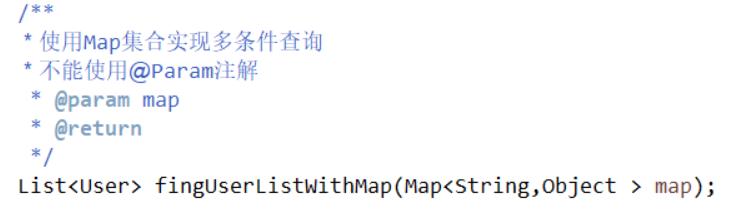


###### 测试代码

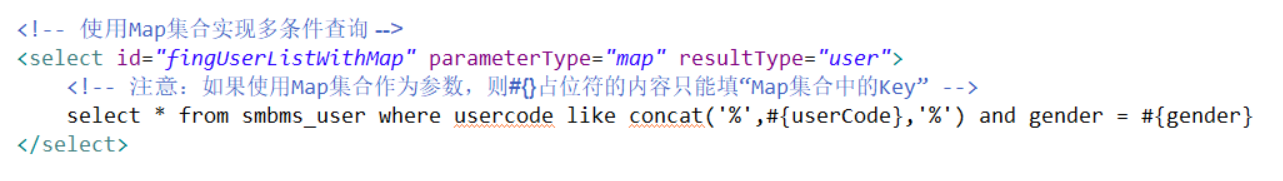


##### 使用Map集合

###### Mapper接口



###### 映射文件



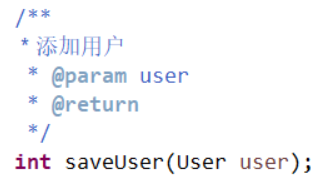
###### 测试代码



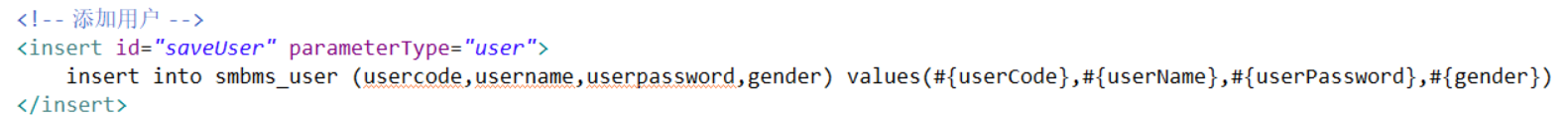
### insert

#### 添加用户

##### Mapper接口



##### 映射文件



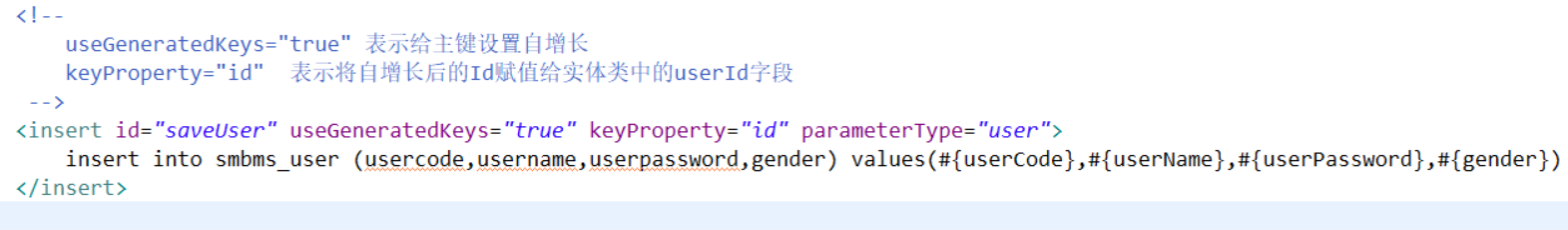
注意：增删改没有resultType属性

##### 程序代码

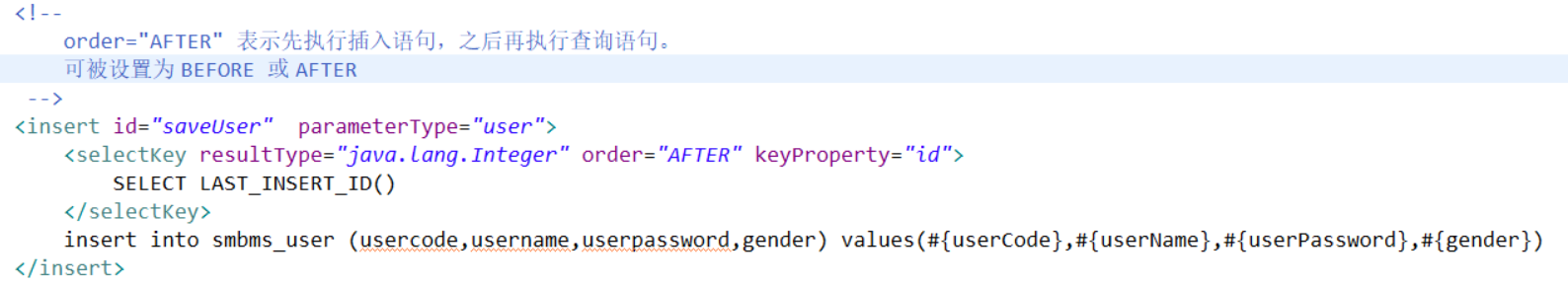


##### 主键返回

方式一



方式二



### update

#### 修改用户

##### 映射文件

##### 程序代码

##### 主键返回

### delete

#### 删除用户

##### 映射文件

##### 程序代码

##### 主键返回

## resultMap

### resultMap的使用场景

### resultMap的使用方法

#### 定义ResultMap

#### 使用resultMap代替resultType

#### mapper接口

#### 测试

### resultMap和resultType

## 复杂的关联关系映射

### 一对一

#### 需求

#### resultType

##### SQL语句

##### 创建实体类

##### 映射文件

##### mapper接口

##### 测试类

#### resultMap

##### SQL语句

##### resultMap映射思路

##### 修改实体类

##### 映射文件

##### mapper接口

##### 测试类

#### 一对一总结

### 一对多

#### 需求

#### SQL语句

#### 修改实体类

#### 映射文件

#### mapper接口

#### 测试类

### 多对多

#### 需求

#### SQL语句

#### 映射思路

#### 映射文件

#### mapper接口

#### 测试类

### 多对多总结

### resultMap自动映射

### resultMap总结

## MyBatis缓存

# 第三章：动态SQL

## 使用if + where查询

### 需求

### mapper接口

### 映射文件

### 测试类

## 使用if + trim查询

### 需求

### mapper接口

### 映射文件

### 测试类

## 使用if + set修改

### 需求

### mapper接口

### 映射文件

### 测试类

## 使用if + trim修改

### 需求

### mapper接口

### 映射文件

### 测试类

## 使用foreach完成复杂查询

### 数组array

#### 需求

#### mapper接口

#### 映射文件

#### 测试类

### 集合list

#### 需求

#### mapper接口

#### 映射文件

#### 测试类

### 集合map

#### 需求

#### mapper接口

#### 映射文件

#### 测试类

## choose(when、otherwise)

### 需求

### mapper接口

### 映射文件

### 测试类

## MyBatis分页

### 需求

### mapper接口

### 映射文件

### 测试类