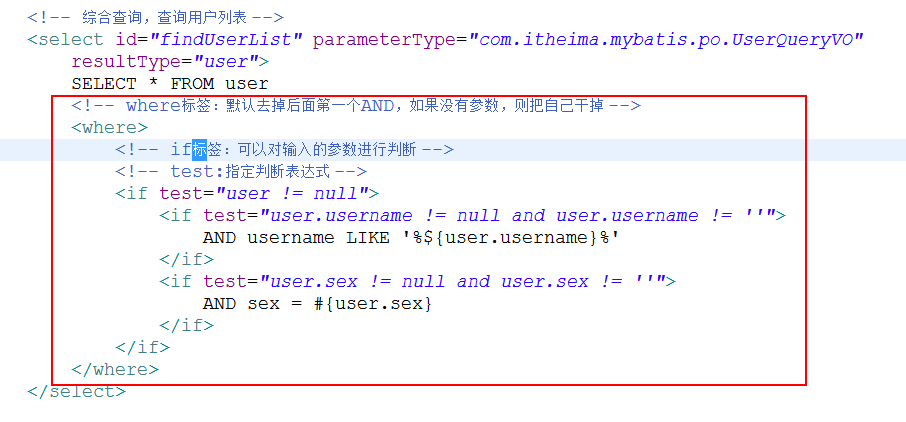
mybatis重点是对sql的灵活解析和处理。

# if和where

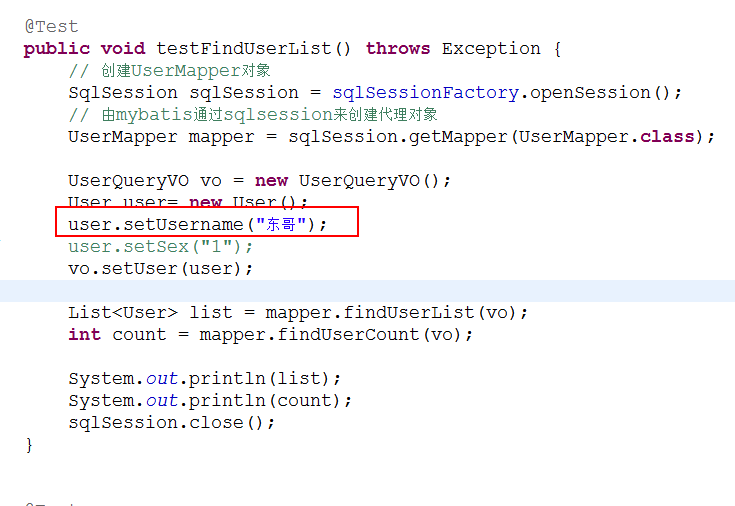
## 需求

将自定义查询条件查询用户列表和查询用户列表总记录数改为动态sql

## Mapper.xml



### 测试代码



# sql片段—拼接

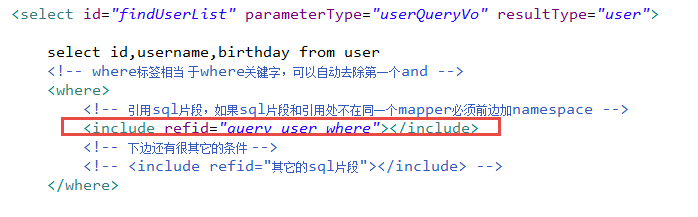
通过sql片段可以将通用的sql语句抽取出来，单独定义，在其它的statement中可以引用sql片段。提高代码重用性

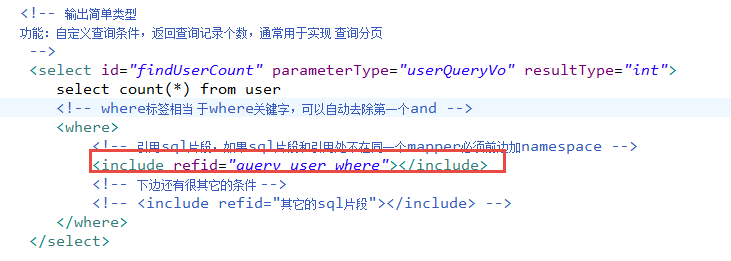
通用的sql语句，常用：where条件、查询列

## sql片段的定义



## 引入sql片段





# foreach

## 传递pojo对象中的List集合

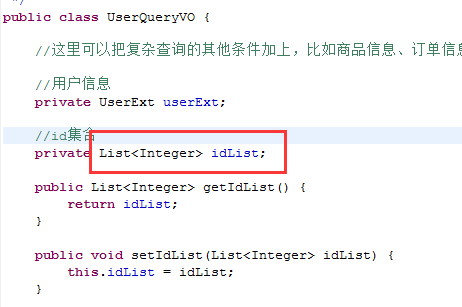
### 需求

在用户查询列表和查询总数的statement中增加多个id输入查询。

### SQL

SELECT \* FROM user WHERE **id IN (1,10,16)**

### 定义pojo中的List属性



### 映射文件

<!-- [foreach标签]：表示一个foreach循环 -->

<!-- [collection]：集合参数的名称，如果是直接传入集合参数，则该处的参数名称只能填写[list]。 -->

<!-- [item]：每次遍历出来的对象 -->

<!-- [open]：开始遍历时拼接的串 -->

<!-- [close]：结束遍历时拼接的串 -->

<!-- [separator]：遍历出的每个对象之间需要拼接的字符 -->

<if test=*"idList != null and idList.size > 0"*>

<foreach collection=*"idList"* item=*"id"* open=*"AND id IN ("* close=*")"* separator=*","*>

#{id}

</foreach>

</if>

### Mapper接口

//根据用户ID的集合查询用户列表（学习foreach标签之通过POJO对象传ID集合）

**public** List<UserExt> findUserList(UserQueryVO vo);

### 测试代码

@Test

**public** **void** testFindUserList() {

// 创建SqlSession

SqlSession sqlSession = sqlSessionFactory.openSession();

// 通过SqlSession，获取mapper接口的动态代理对象

UserMapper mapper = sqlSession.getMapper(UserMapper.**class**);

// 构造QueryUserVO对象

QueryUserVO vo = **new** QueryUserVO();

// UserExt ext = new UserExt();

// ext.setUsername("小明");

// ext.setSex("1");

// vo.setUserExt(ext);

// 创建用户ID集合，然后设置到QueryUserVO对象中

List<Integer> idList = **new** ArrayList<Integer>();

idList.add(1);

idList.add(10);

idList.add(16);

vo.setIdList(idList);

// 调用mapper代理对象的方法

List<UserExt> list = mapper.findUserList(vo);

System.*out*.println(list);

// 关闭SqlSession

sqlSession.close();

}

## 直接传递List集合（自学）

### 需求

根据用户ID的集合查询用户列表

### SQL

SELECT \* FROM user WHERE id IN (1,10,16)

### 映射文件

<!-- 根据用户ID的集合查询用户列表（学习foreach标签之直接传ID集合） -->

<!--

[foreach标签]：表示一个foreach循环

[collection]：集合参数的名称，如果是直接传入集合参数，则该处的参数名称只能填写[list]。

[item]：定义遍历集合之后的参数名称

[open]：开始遍历之前需要拼接的SQL串

[close]：结束遍历之后需要拼接的SQL串

[separator]：遍历出的每个对象之间需要拼接的字符

-->

<select id=*"findUsersByIdList"* parameterType=*"****java.util.List****"* resultType=*"user"*>

SELECT \* FROM USER

<where>

<if test=*"****list*** *!= null and* ***list****.size > 0"*>

<foreach collection=*"****list****"* item=*"id"* open=*"AND id IN ("* close=*")"* separator=*","*>

#{id}

</foreach>

</if>

</where>

</select>

### Mapper接口

//根据用户ID的集合查询用户列表（学习foreach标签之直接传ID集合）

**public** List<User> findUsersByIdList (List<Integer> idList);

### 测试代码

@Test

**public** **void** findUsersByIdListTest() {

// 创建SqlSession

SqlSession sqlSession = sqlSessionFactory.openSession();

// 通过SqlSession，获取mapper接口的动态代理对象

UserMapper userMapper = sqlSession.getMapper(UserMapper.**class**);

// 构造List<Integer>集合

List<Integer> idList = **new** ArrayList<Integer>();

idList.add(1);

idList.add(10);

idList.add(16);

// 调用mapper对象的方法

List<User> list = userMapper.findUsersByIdList (idList);

System.*out*.println(list);

// 关闭SqlSession

sqlSession.close();

}