# Mybatis介绍

MyBatis是一个优秀的持久层框架，它对jdbc的操作数据库的过程进行封装，使开发者只需要关注 SQL 本身，而不需要花费精力去处理例如注册驱动、创建connection、创建statement、手动设置参数、结果集检索等jdbc繁杂的过程代码。

Mybatis通过xml或注解的方式将要执行的各种statement（statement、preparedStatemnt、CallableStatement）配置起来，并通过java对象和statement中的sql进行映射生成最终执行的sql语句，最后由mybatis框架执行sql并将结果映射成java对象并返回。

# jdbc访问数据库问题总结

1. 数据库连接创建、释放频繁造成系统资源浪费，从而影响系统性能。如果使用数据库连接池可解决此问题。
2. Sql语句在代码中硬编码，造成代码不易维护，实际应用中sql变化的可能较大，sql变动需要改变java代码。
3. 使用preparedStatement向占有位符号传参数存在硬编码，因为sql语句的where条件不一定，可能多也可能少，修改sql还要修改代码，系统不易维护。
4. 对结果集解析存在硬编码（查询列名），sql变化导致解析代码变化，系统不易维护，如果能将数据库记录封装成pojo对象解析比较方便。

# mybatis架构



1. mybatis配置

SqlMapConfig.xml，此文件作为mybatis的全局配置文件，配置了mybatis的运行环境等信息。

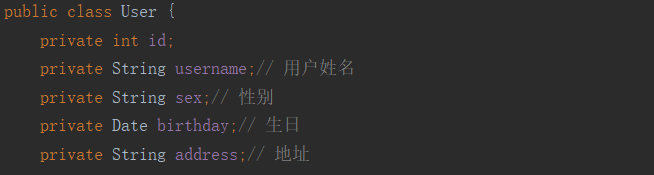
mapper.xml文件即sql映射文件，文件中配置了操作数据库的sql语句。此文件需要在SqlMapConfig.xml中加载。

1. 通过mybatis环境等配置信息构造SqlSessionFactory即会话工厂
2. 由会话工厂创建sqlSession即会话，操作数据库需要通过sqlSession进行。
3. mybatis底层自定义了Executor执行器接口操作数据库，Executor接口有两个实现，一个是基本执行器、一个是缓存执行器。
4. Mapped Statement也是mybatis一个底层封装对象，它包装了mybatis配置信息及sql映射信息等。mapper.xml文件中一个sql对应一个Mapped Statement对象，sql的id即是Mapped statement的id。
5. Mapped Statement对sql执行输入参数进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql前将输入的java对象映射至sql中，输入参数映射就是jdbc编程中对preparedStatement设置参数。
6. Mapped Statement对sql执行输出结果进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql后将输出结果映射至java对象中，输出结果映射过程相当于jdbc编程中对结果的解析处理过程。

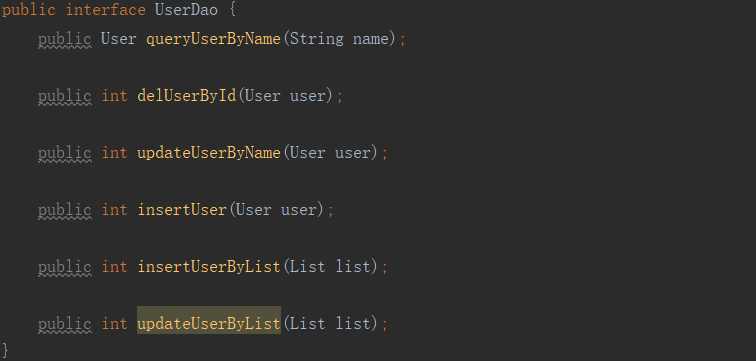
# 四、mapper动态代理开发方式

## 简单类型的crud操作

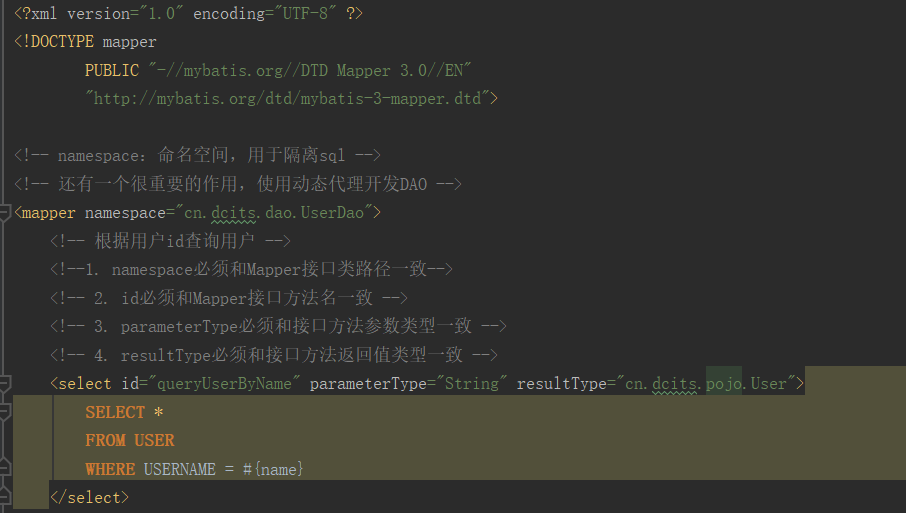
### 1. 创建pojo类和getter/setter方法



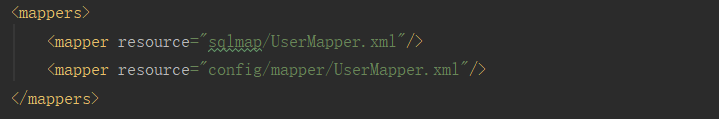
### 2．创建DAO接口



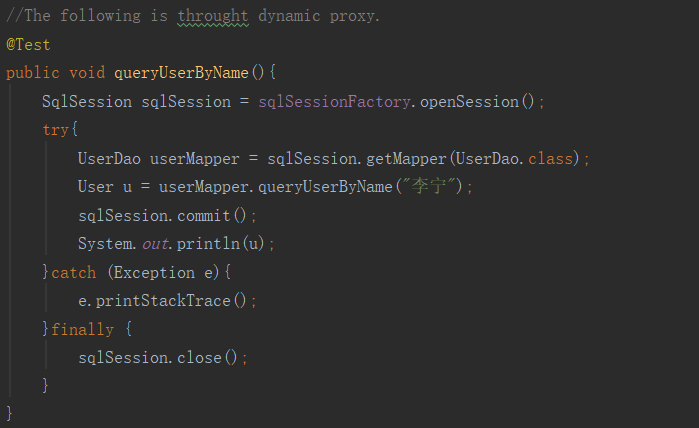
### 3. 创建UserMapper.xml文件

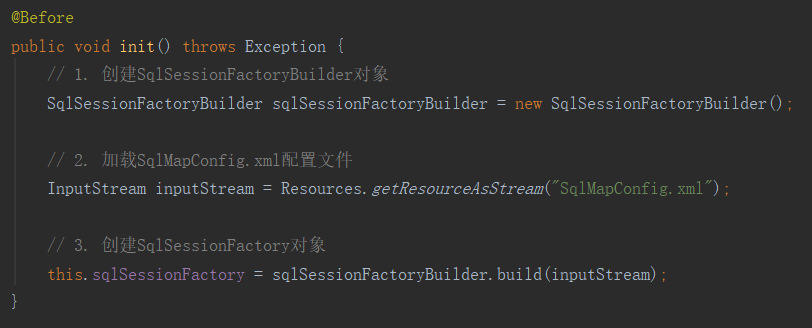


### 4. 主配置文件（SqlMapConfig.xml）引入UserMapper



### 5. 测试





### 小结

#### #{}和${}的区别

#{}表示一个占位符号，通过#{}可以实现preparedStatement向占位符中设置值，自动进行java类型和jdbc类型转换。#{}可以有效防止sql注入。 #{}可以接收简单类型值或pojo属性值。 如果parameterType传输单个简单类型值，#{}括号中可以是value或其它名称。

${}表示拼接sql串，通过${}可以将parameterType 传入的内容拼接在sql中且不进行jdbc类型转换， ${}可以接收简单类型值或pojo属性值，如果parameterType传输单个简单类型值，${}括号中只能是value。

#### ParameterType和ResultType

parameterType：指定输入参数类型，mybatis通过ognl从输入对象中获取参数值拼接在sql中。

resultType：指定输出结果类型，mybatis将sql查询结果的一行记录数据映射为resultType指定类型的对象。如果有多条数据，则分别进行映射，并把对象放到容器List中

#### selectOne和selectList

selectOne查询一条记录，如果使用selectOne查询多条记录则抛出异常。

selectList可以查询一条或多条记录。

动态代理对象调用sqlSession.selectOne()和sqlSession.selectList()是根据mapper接口方法的返回值决定，如果返回list则调用selectList方法，如果返回单个对象则调用selectOne方法。

使用mapper代理方法开发mapper接口，程序员不用编写mapper接口实现类，使用mapper代理方法时，输入参数可以使用pojo包装对象或map对象，保证dao的通用性。

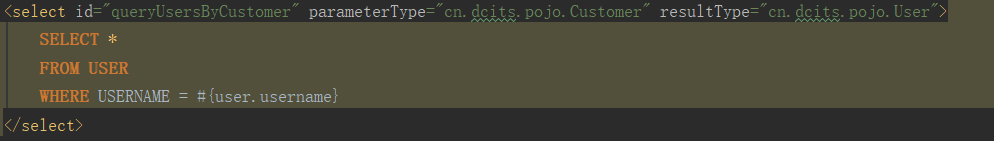
## 复杂类型的crud操作

### 传入的参数中包含pojo

#### 创建Customer类



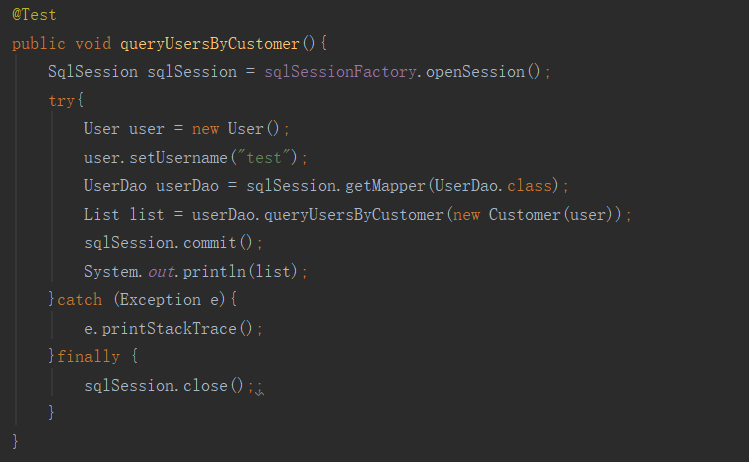
#### UserMapper.xml



#### UserDao接口

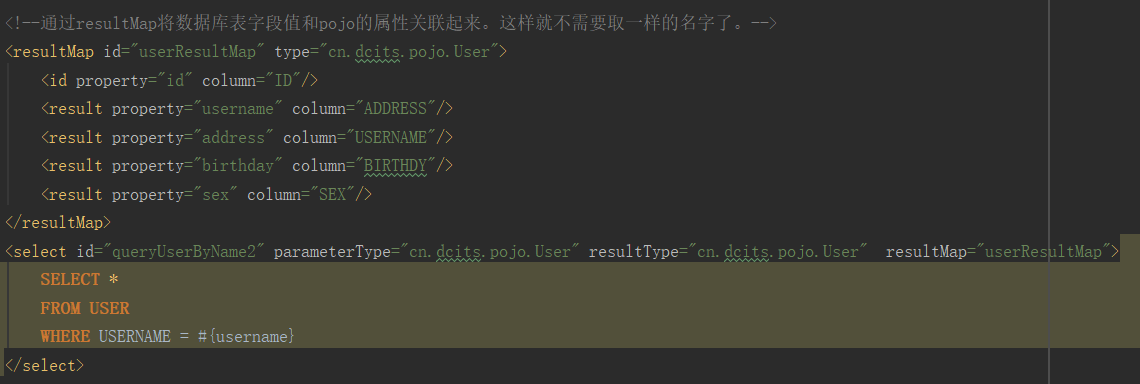


#### 测试



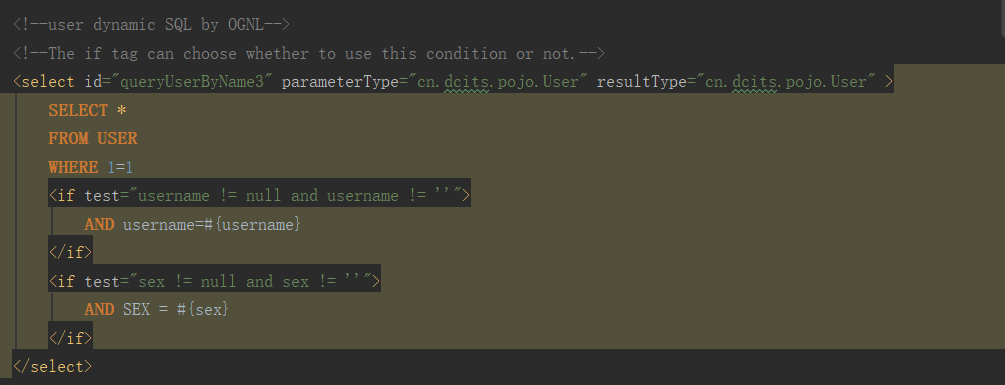
### 数据库表字段和类属性名不对应的时候

#### UserMapper.xml

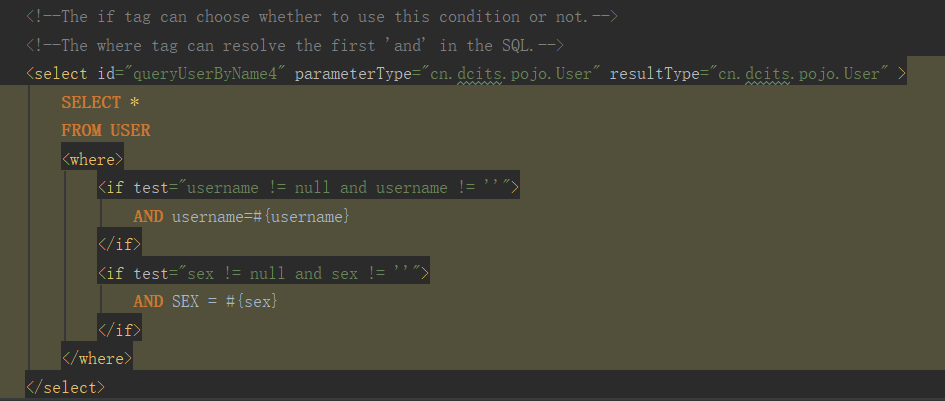


### 动态SQL

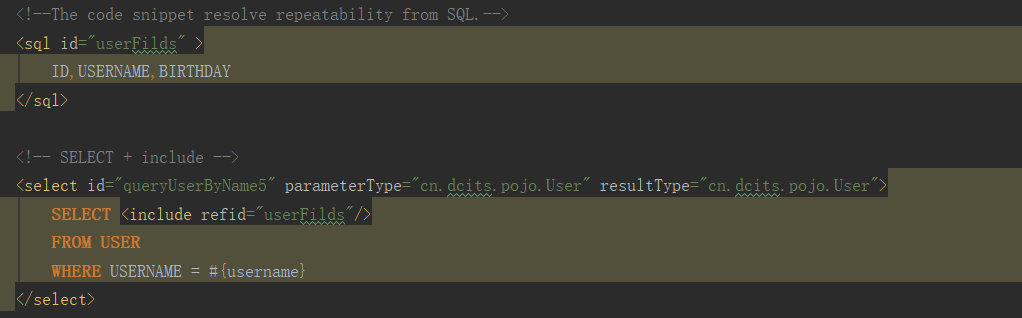
#### If标签的UserMapper.xml配置



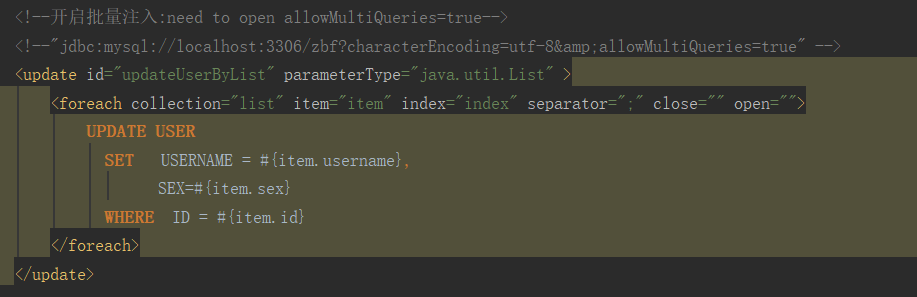
#### Where标签的UserMapper.xml配置

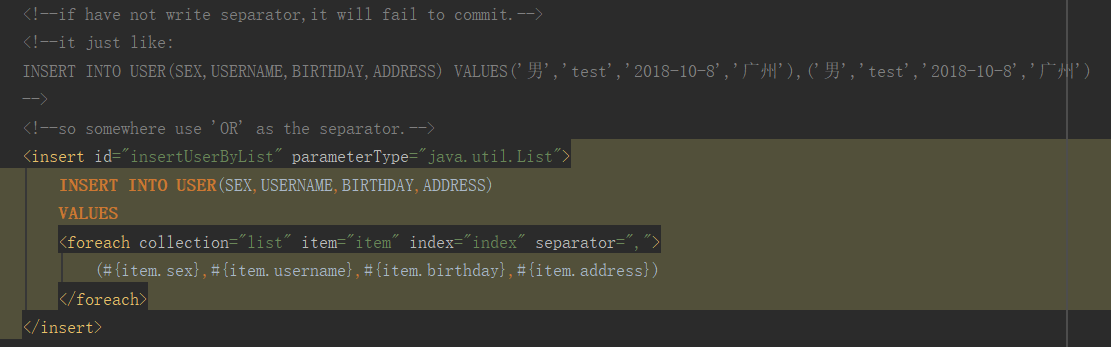


#### Sql片段



#### Foreach





### 一对一查询和一对多查询

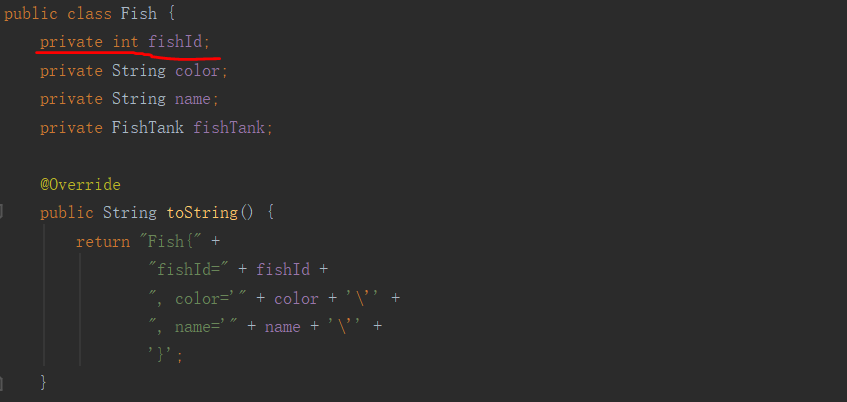
#### 表结构

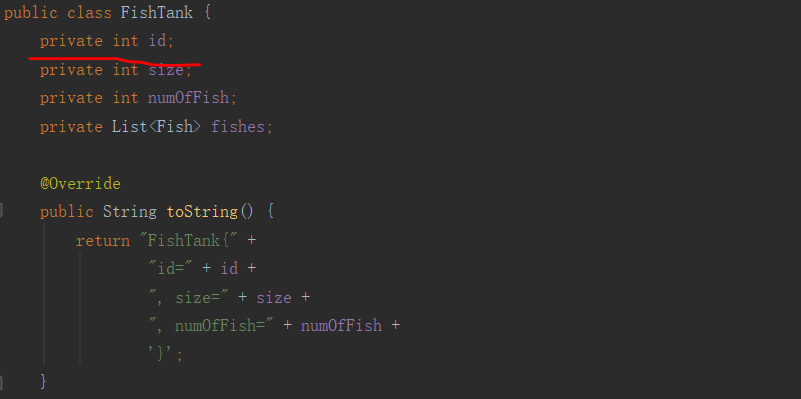




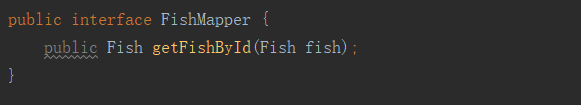
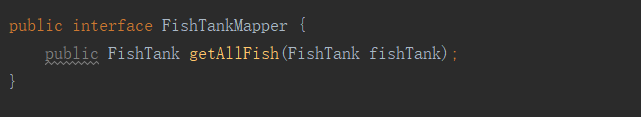
#### 创建pojo类

创建两个类，Fish和FishTank。一条鱼属于一个鱼缸（一对一），而一个鱼缸可以有多条鱼（一对多）。注意：如果两个类的id属性名一样，在List下只能注入一个记录。大概是数据库通过map保存返回数据，id同名覆盖了把。





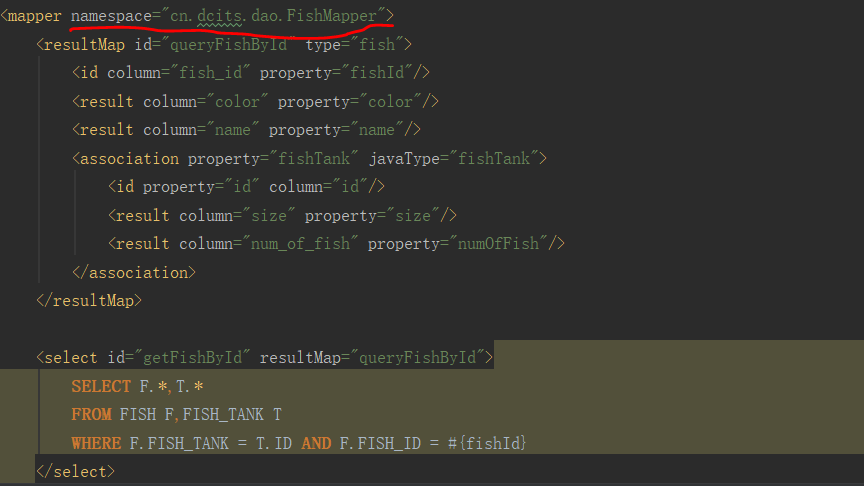
#### 编写mapper

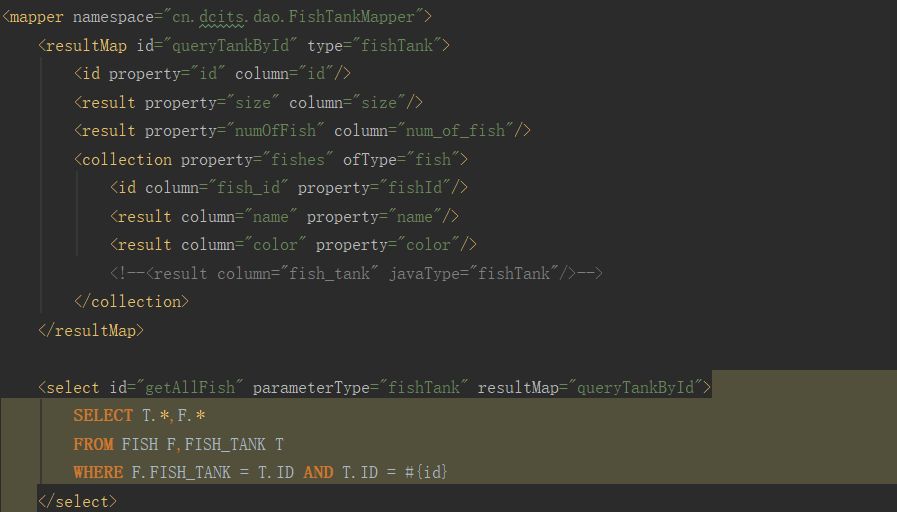
 

#### 编写mapper对应的xml文件

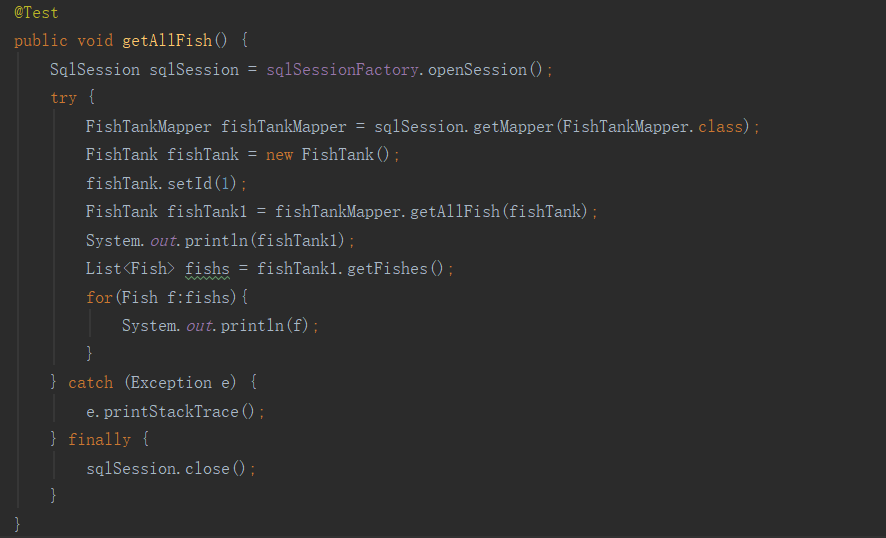
在<association>下不需要写List集合对象。

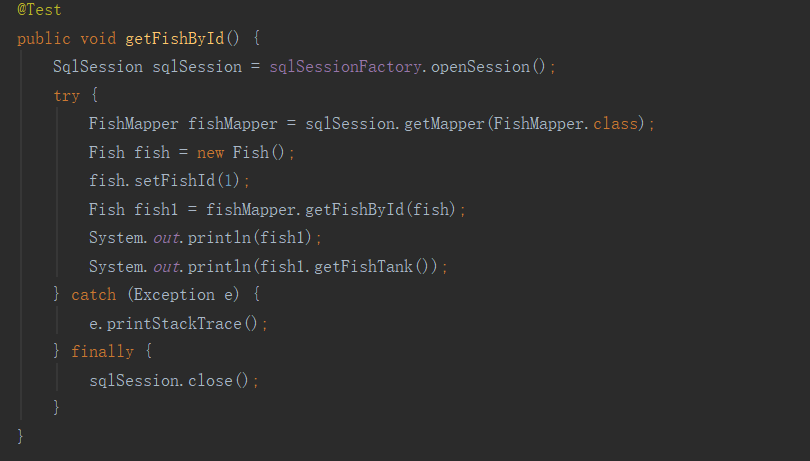
在<collection>下不需要FishTank对象。



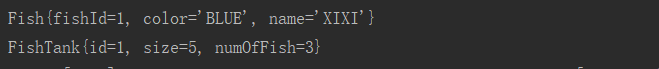


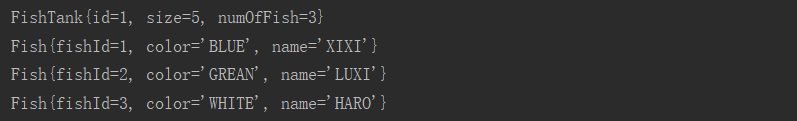
#### 编写测试方法





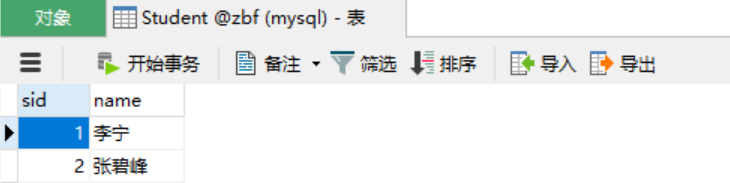
#### 测试结果

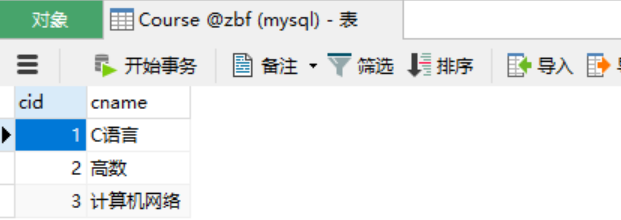




### 多对多查询

#### 表结构



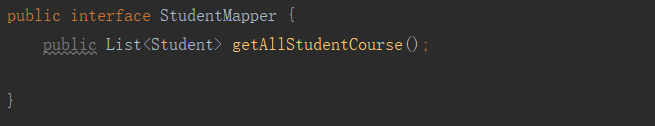


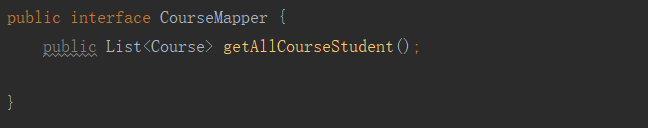
#### 创建POJO类



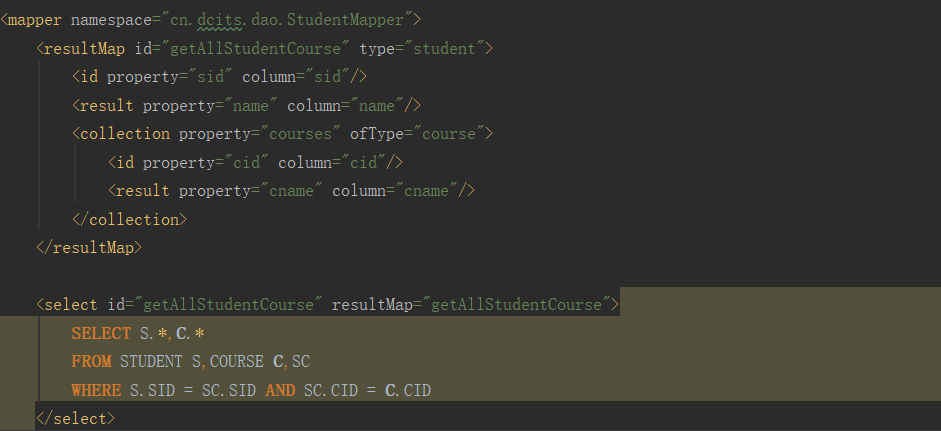


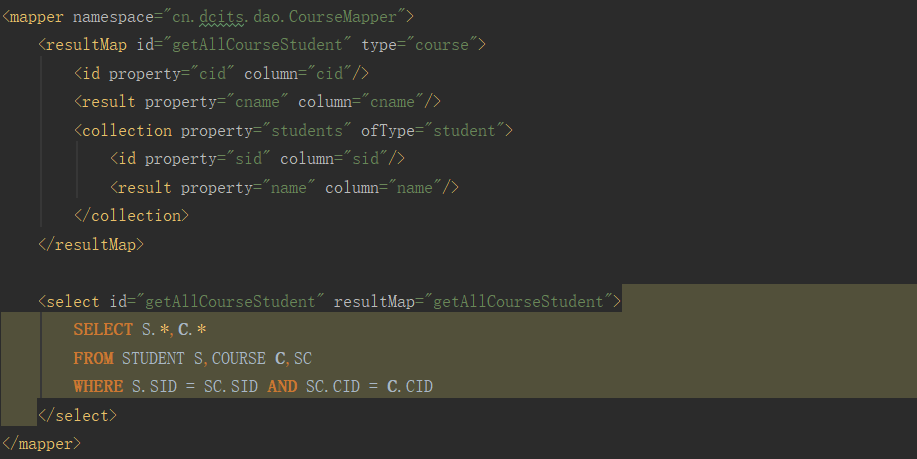
#### 编写mapper



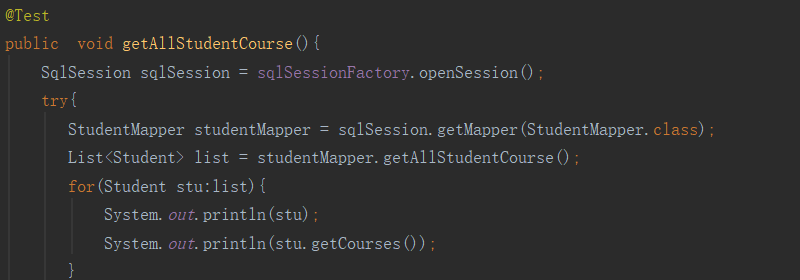


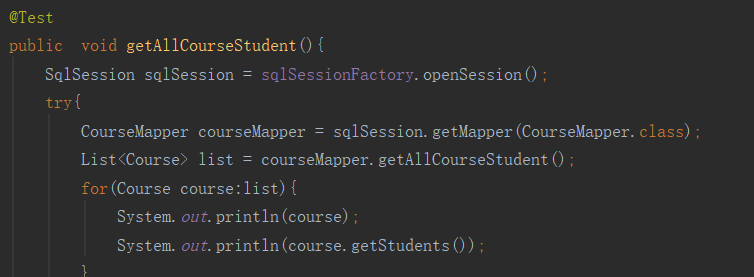
#### 编写mapper对应的xml文件



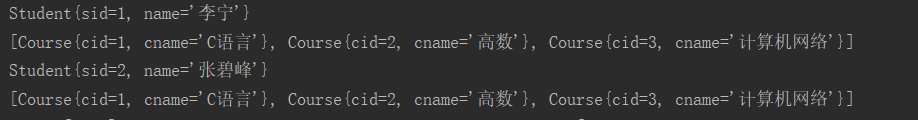


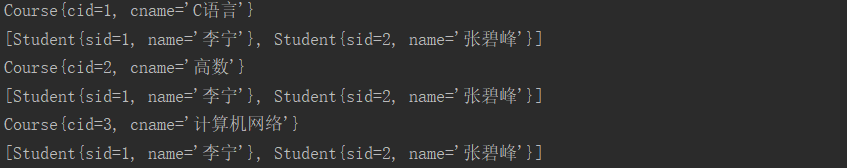
#### 编写测试方法





#### 测试结果





# 五、Mybatis整合Spring

# 六、逆向工程

# SqlMapConfig.xml配置内容

1. properties（属性）
2. settings（全局配置参数）
3. typeAliases（类型别名）
4. typeHandlers（类型处理器）
5. objectFactory（对象工厂）
6. plugins（插件）
7. environments（环境集合属性对象）
8. environment（环境子属性对象）
9. transactionManager（事务管理）
10. dataSource（数据源）
11. mappers（映射器）