## 基本配置:

原理图：



Mybatis的配置xml （类似hibernate的配置文件

Mybatis.xml:



Category.xml:

相当于将dom 完全使用配置文件表示出来，就不需要写dao了，这里是表映射类的地方，将他们关联起来



使用：这里是selectList，如果单独取出可以使用selectOne。



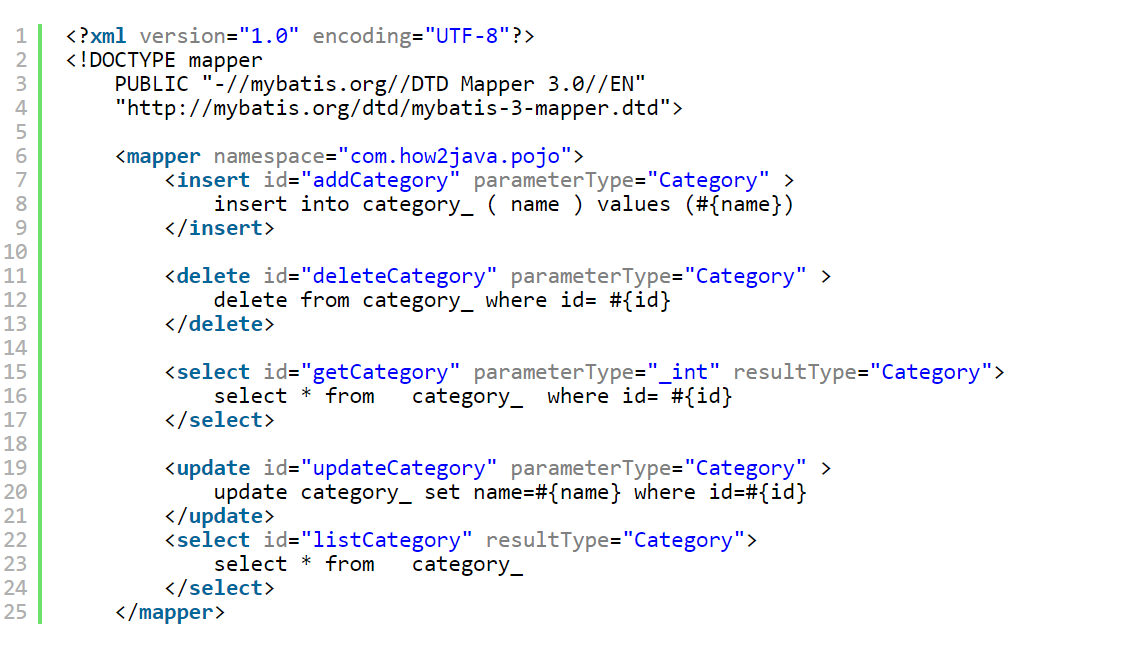
## 2.增删改查：

类似上面list取出，mybatis中对于CRUD有不同的标签对应不同的功能：

在对应类中的配置文件中修改即可。

如果返回的是单个或者集合，那么参数是resultType，

如果返回的表需要赋值给某个设计bean，那么参数是resultMap



其中parameterType是传来参数的类型，resultType是返回的类型。

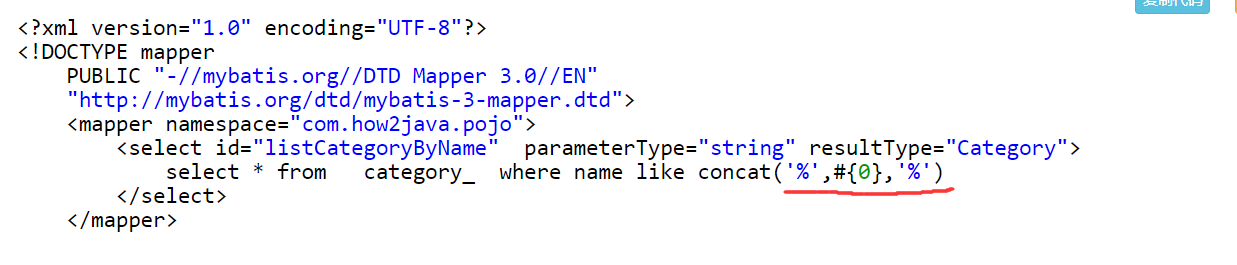
增删改查大多都是以对象为单位传入，然后根据对象的#{name},#{id},进行操作。

<https://how2j.cn/k/mybatis/mybatis-crud/1088.html#nowhere>所有例子

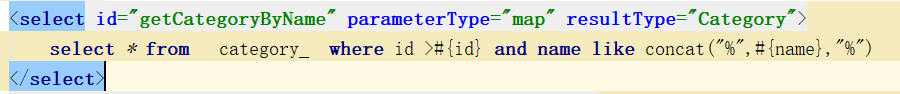
如 更新某一对象的例子：



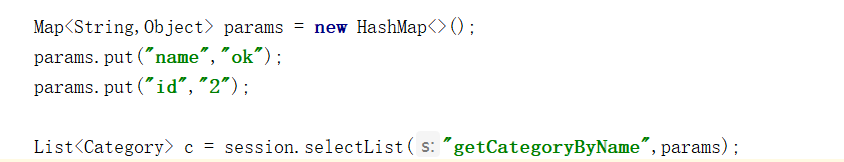
模糊查询：



复合查询:



传值需要一个map：



## 3.数据关系

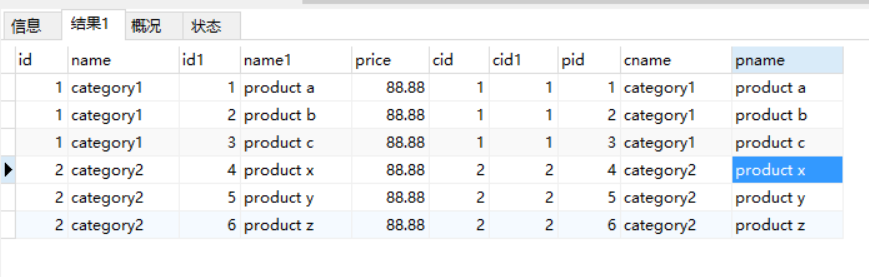
### join属性为集合，则使用collection，

### join属性为单个，则使用assciation。

### 1.一对多:

Category对Product是一对多的关系:

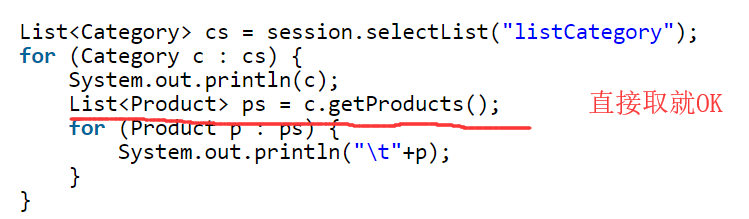
不管是多对一还是一对多都是先把他们的全表查出来:



不同点在于赋值的bean不同

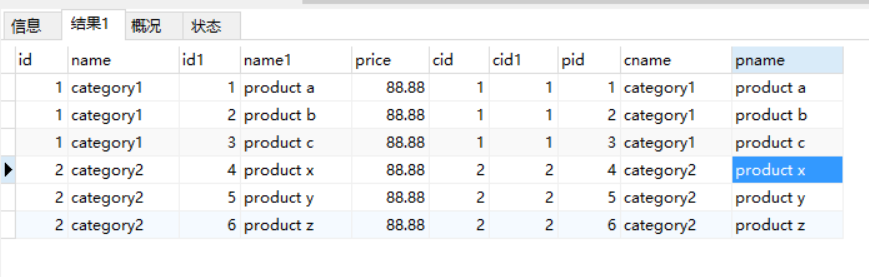
一对多，返回category的形式：





### 2.多对一:

也是先查全表，再进行赋值：



多对一是查询的product:



### 3.多对多（要依靠中间表）

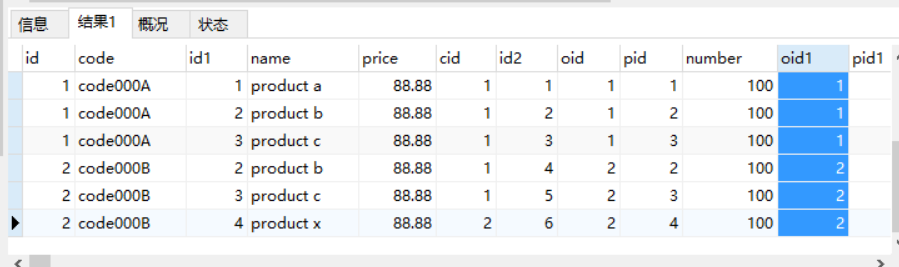
以product，order，和中间表OrderItem为例，

需要在Order中设置List<OrderItem>（中间表），来实现和Product的多对多关系

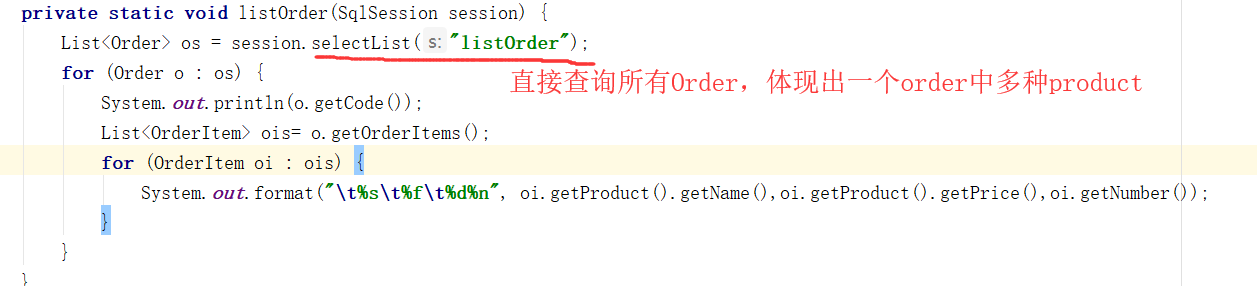
在Order的配置中，order对Order\_item是一对多的关系，使用collection联系Order\_item,再使用assciation联系product



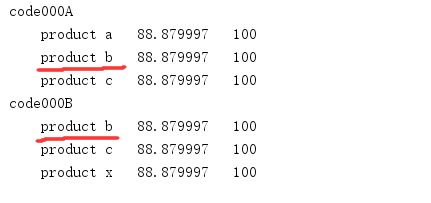
总表:



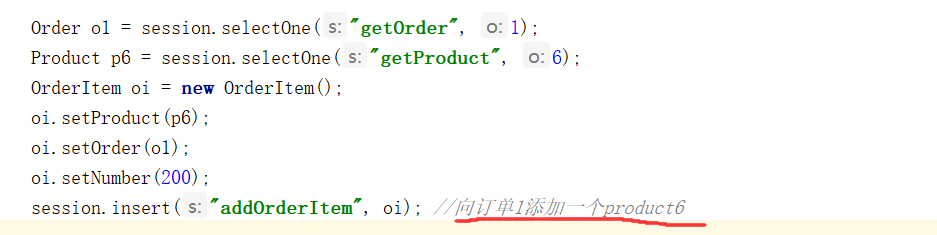
查询多对多：



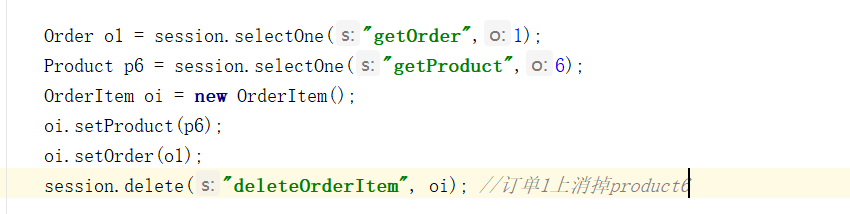
结果:



添加OrderItem内容(这里不论是增加还是删除，oi都相当于Map的作用)

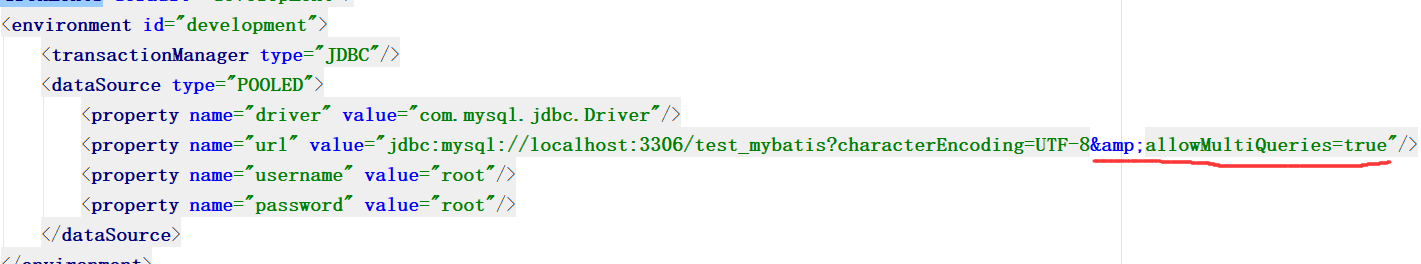


删除订单上的关系:比如在订单1上消掉product6:

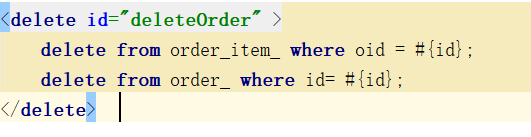


使用多条sql并用：

需要先配置:



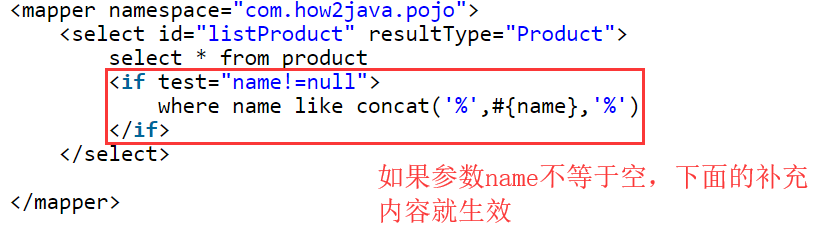
多条语句并行:



## 4.动态SQL查询:

### 1.参数判断if:

如果基于某个查询sql有了限定条件(即where后面的内容)，我们可以使用一个sql并带if判断参数的功能：



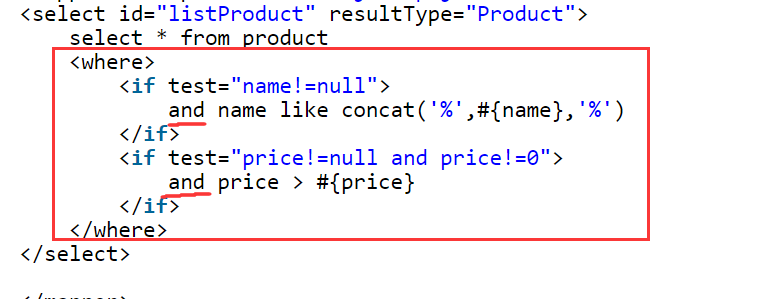
整句话就是加上了模糊查询的定义。

这个就很方便，不用再写一个sql来专门模糊查询了。

### where条件查询标签

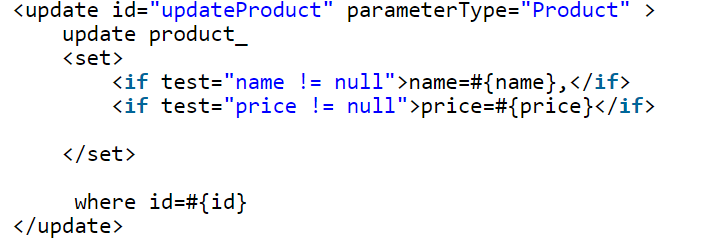
如果只有一个限制条件，比如where name =”ok” 可以就正常写。

在需要多个限制条件时，比如where name =”xxx” and age = ”xx”的时候，就需要使用where标签。



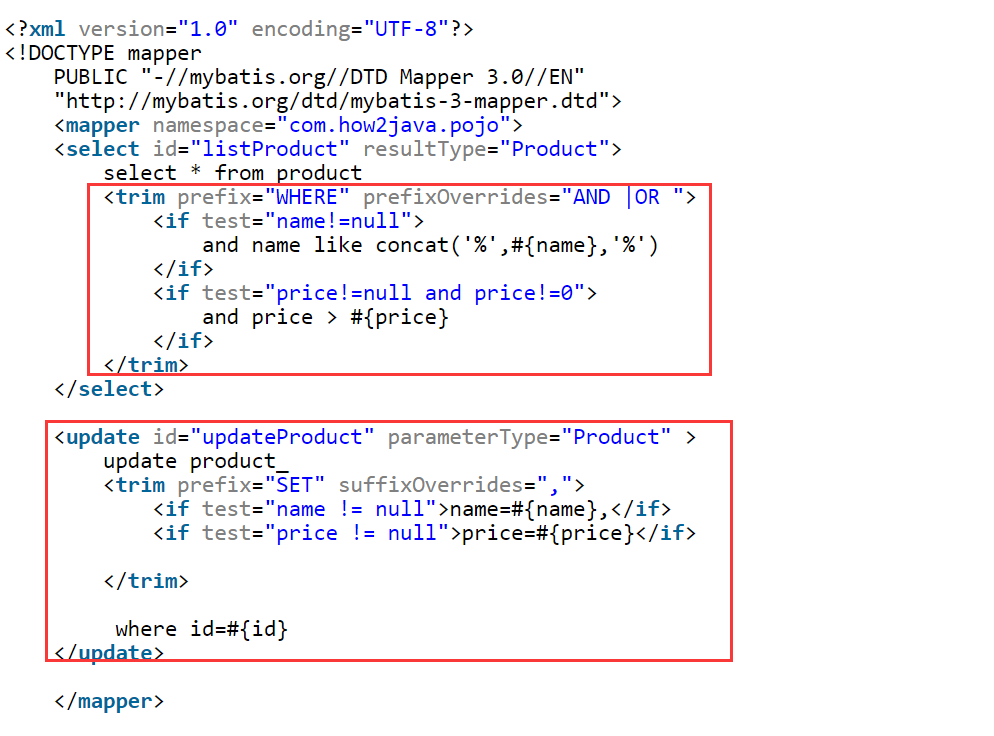
这样可以防止第一个无参数，第二个有参数的问题。框架写好了，会自己判断。

在update中set有一样的功能

:

### trim标签（自定义功能）

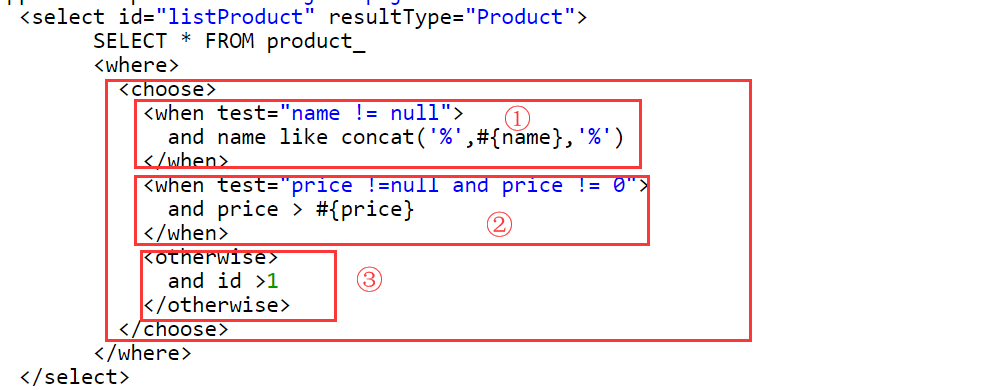
trim 用来定制想要的功能，比如where标签就可以用



效果一样

### choose (if –ifelse -else):

这个类似于if-ifesle-else的方式，



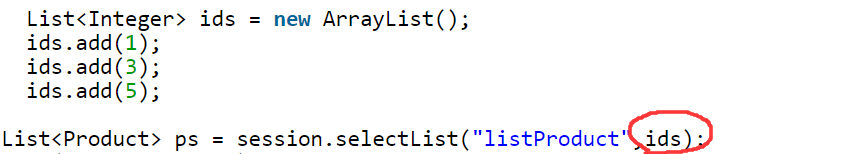
从上至下，只要能满足就执行当前，然后不会往下走了。

如果前两个都不满足，就会执行otherwise(else)里面的数据。

注意，出错了是不会进行的。比如name传了个随便的值，数据库找不到，这叫错误，而不是判断的问题。

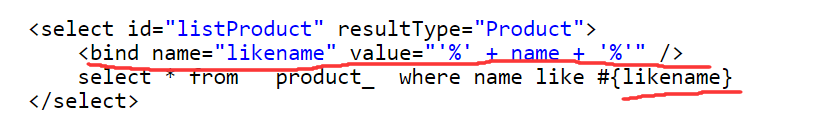
### foreach（参数迭代）/bind(参数加工)

Foreach：对传过来的参数的迭代





Bind，对传过来的参数进行二次加工:



## 5.使用注解:

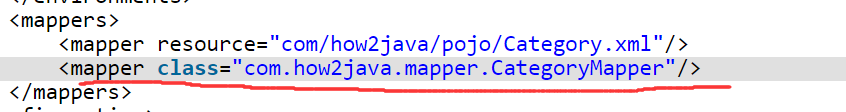
### 1.增删改查

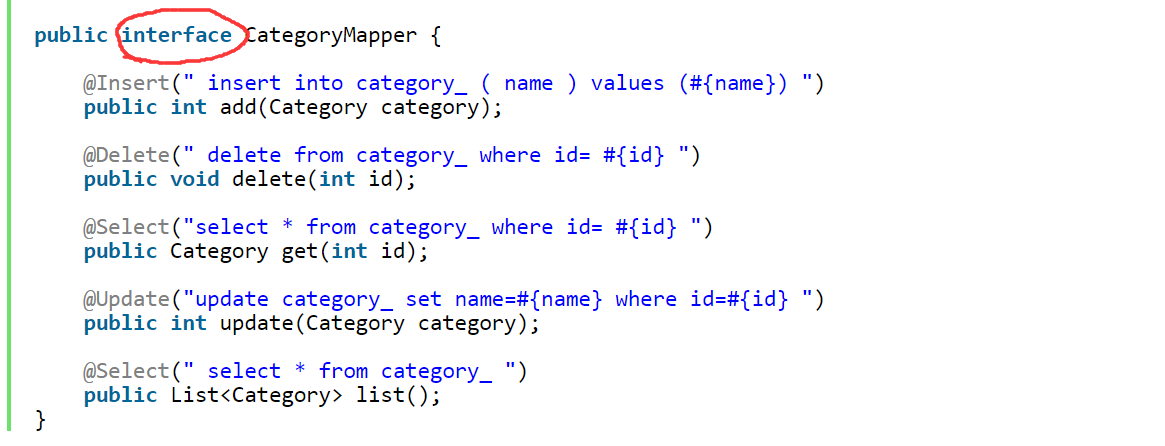
使用一个接口类来进行注解操作:

把不同CRUD方法的sql操作用注解写在上方：

很好理解。

配置文件，使用注解:





### 2.一对多

Category的配置:一个接口。



Results，是对每一个由第一个select查询出来的之后的结果的一个回调操作，类似map函数，其返回的结果还是一个category的List。

应该这么理解:

Results是属性覆盖操作。

比如查出来了一个category，再在result里面进行针对每一个属性的操作。然后覆盖原有的category。这里需要重新覆盖的属性有id和product。

//property属性：指定将关联查询的结果封装到Category对象的product属性上

//column属性指定：指定在执行@Many注解中定义的select语句时，把category表的id字段当做参数传入

//many属性：指定通过@Many注解定义关联查询的语句是ProductMapper中的listByCategory方法

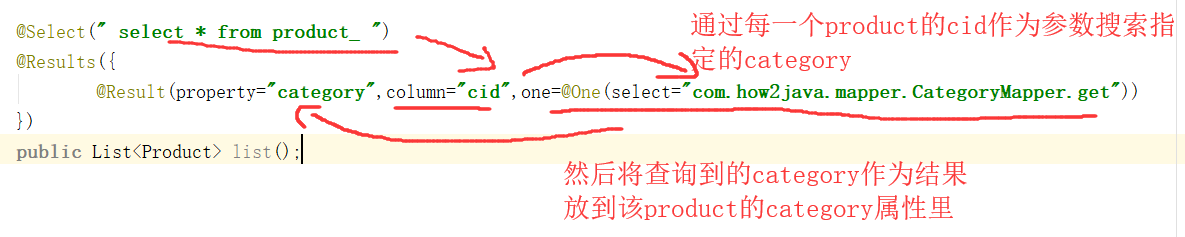
2.Product的配置：



### 3.多对一

多个product对应一个category。

这里是针对product中的category属性赋值。



Result是针对一个属性，这个属性需要另一个类的帮助才能完成补充的操作，比如这里product中的category属性就需要用cid去查询，这就变成了“再查询”之后补充齐整个product对象。

### 4.关于results的用法：

Results是针对查询之后的每一个“对象”的属性补充。

它是用来帮助实现对象，或者List的查询并赋值的。

普通的数据库在某个属性中不能放置一个对象，而是保存那个对象的id然后在取出来的时候查询。

比如category中的products，是一个List数组，所以在数据库中甚至连表示都没有，但是在category类中有List<products>这个属性存在，这个时候就需要使用result对查询之后的category进行属性“补充”。

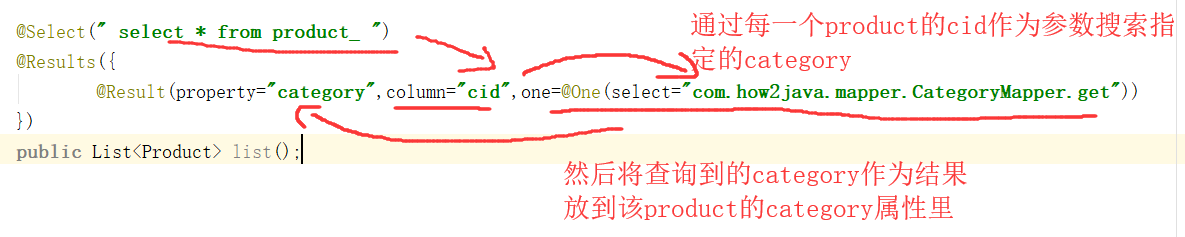
流程就是：

查询category🡪得到category的id作为参数🡪去查询这个category对应的List<products>

进行属性补充，完成一对多的关系。

再比如product中的category，数据库不能放置category对象，只能有一个category的id，但product类中有category类属性。于是查询到所有product之后，再对每一个product“对象”进行属性补充，利用每一个product的cid作为参数去查询对应的category再赋值到该product上，实现属性补充。

如果忘了就看这个流程，one代表的是查一个对象，many是查一个list。

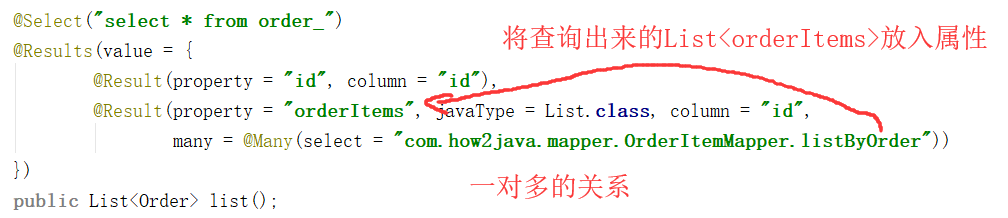


Result的针对每一条数据去查就很方便，有一种map方法的感觉。

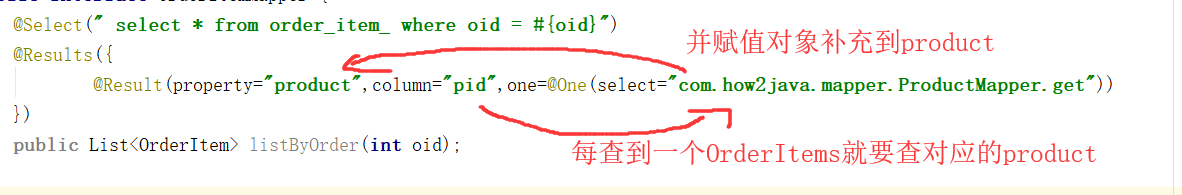
### 5.多对多：

中间表orderItem

一个order对应多个orderItem（Many），一个orderItem对应多个Product（One）

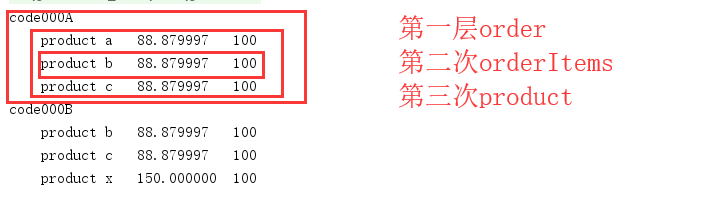


在查询List<OrderItems>的同时，没查到一个OrderItems，就要把对应的product查询出来，并补充对象到哪一个OrderItems。



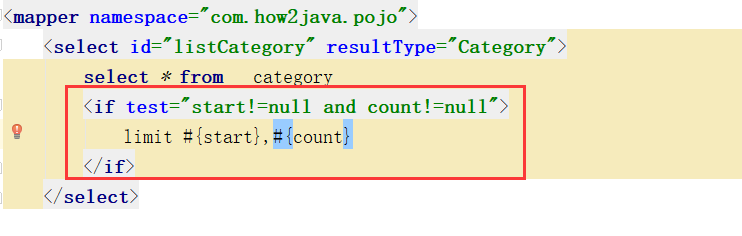
查询流程图:

OrderA 🡪 orderA的orderItems 🡪 每一个orderItems对应的product

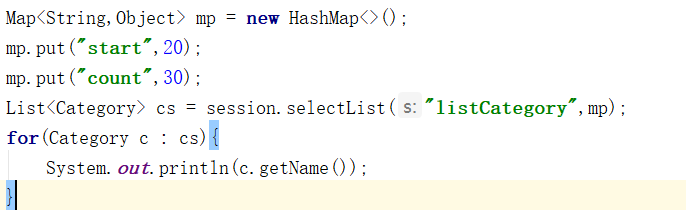


## 分页（Pagehelper）:

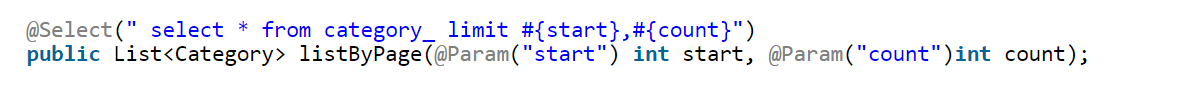
1.使用XML的分页:



传参解构。

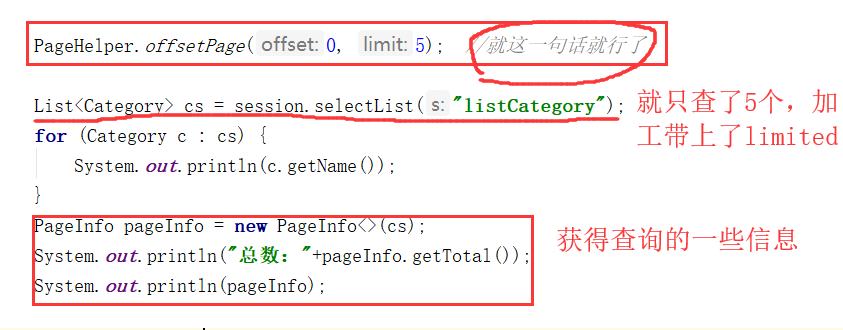


2.使用注解的翻页（也是使用Param(“xxx”)传参的一种方式,原来是传一个对象，自动解构）



3.使用pageHelper

导入两个jar包之后，注册一下插件就可以使用了，这个原理是在sql执行的时候加了语句，比如下面的查询分页，看上去查全部数据，实际上后台加工之后也是带上limited去查询的。



## 缓存(Session，SessionFactory)

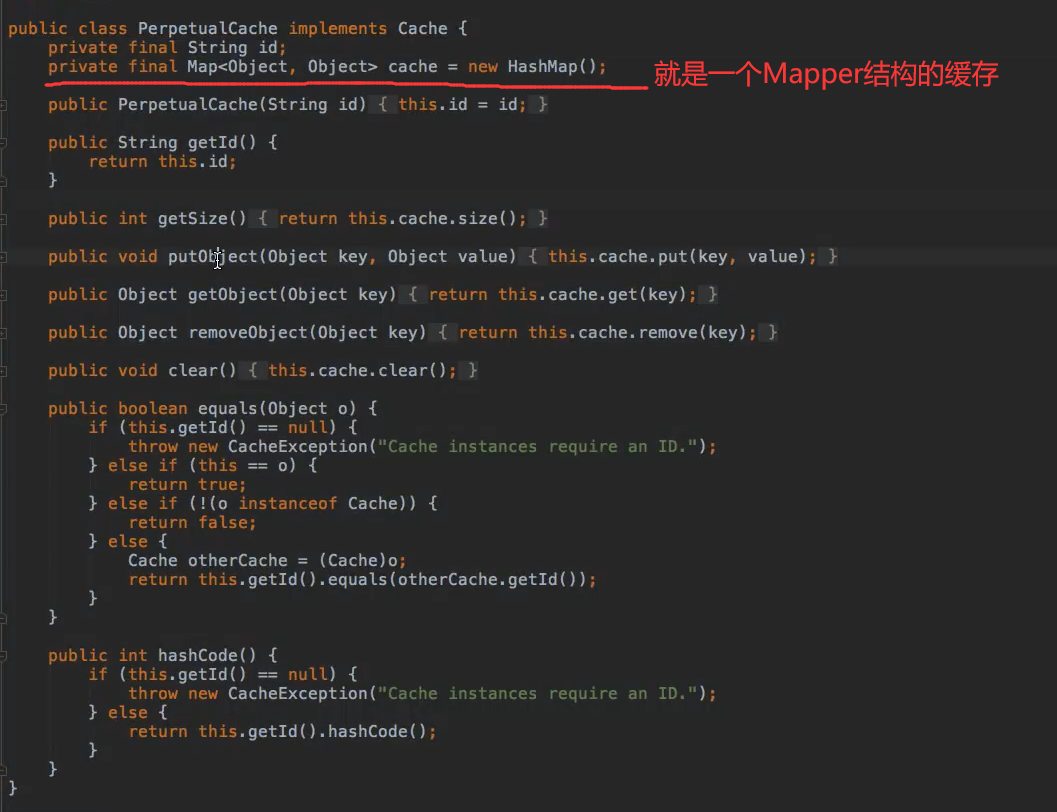
类似于Hibernate的缓存机制，从Session到上层的SessionFactory

1.一级缓存session

Mybatis的一级缓存在Mybatis的session中，即同一个查询语句，第一次查完之后结果会放在session中，第二次查就会在session中取。

而如果两个session，sessionA查语句1，sessionB也查语句1，是会从数据库中进行查询的，所以第一级的缓存在session。

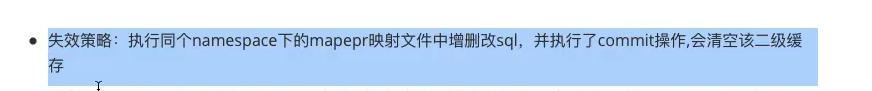
一旦session将语句commit之后，缓存就失效了。



2. 二级缓存SessionFactory

各种配置打开二级缓存，之后所有的缓存都在更上一层的Session中了，即不同的session只要来自相同的SessionFactory，执行一样的语句，后执行的都会从SF的缓存中获取了。

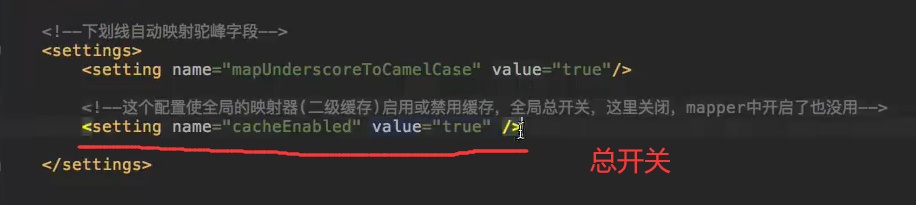
失效策略:

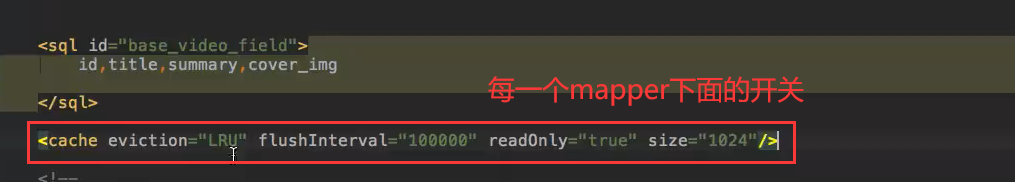


一级缓存同理



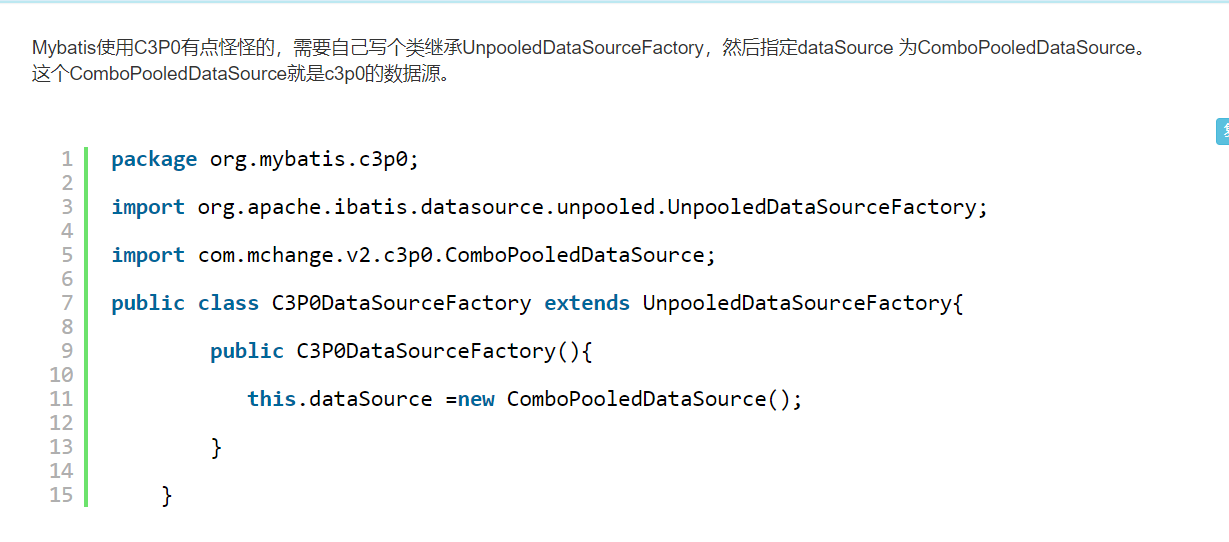
打开二级缓存需要两个地方配置:

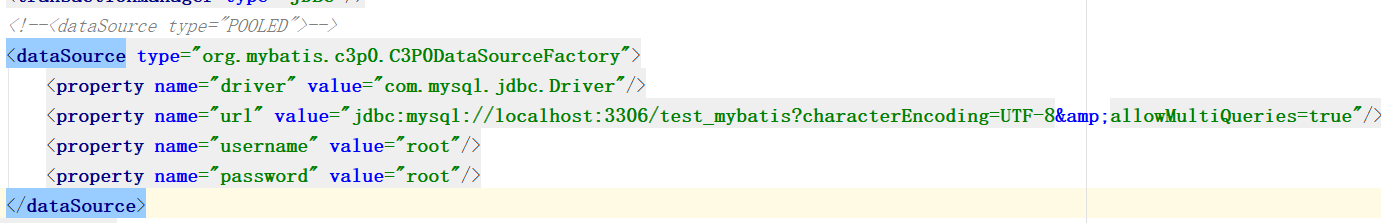




## 高并发工具，延迟加载设置（为关联表查询设计association，collection）

高并发工具c3p0配置:

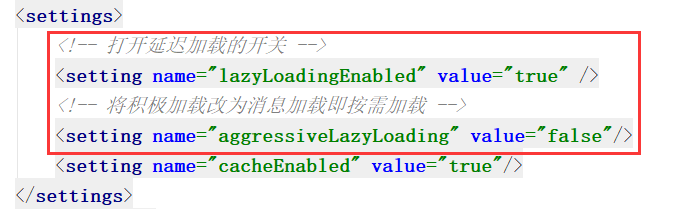




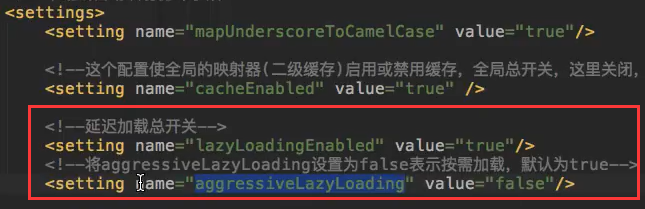
然后正常使用就行。

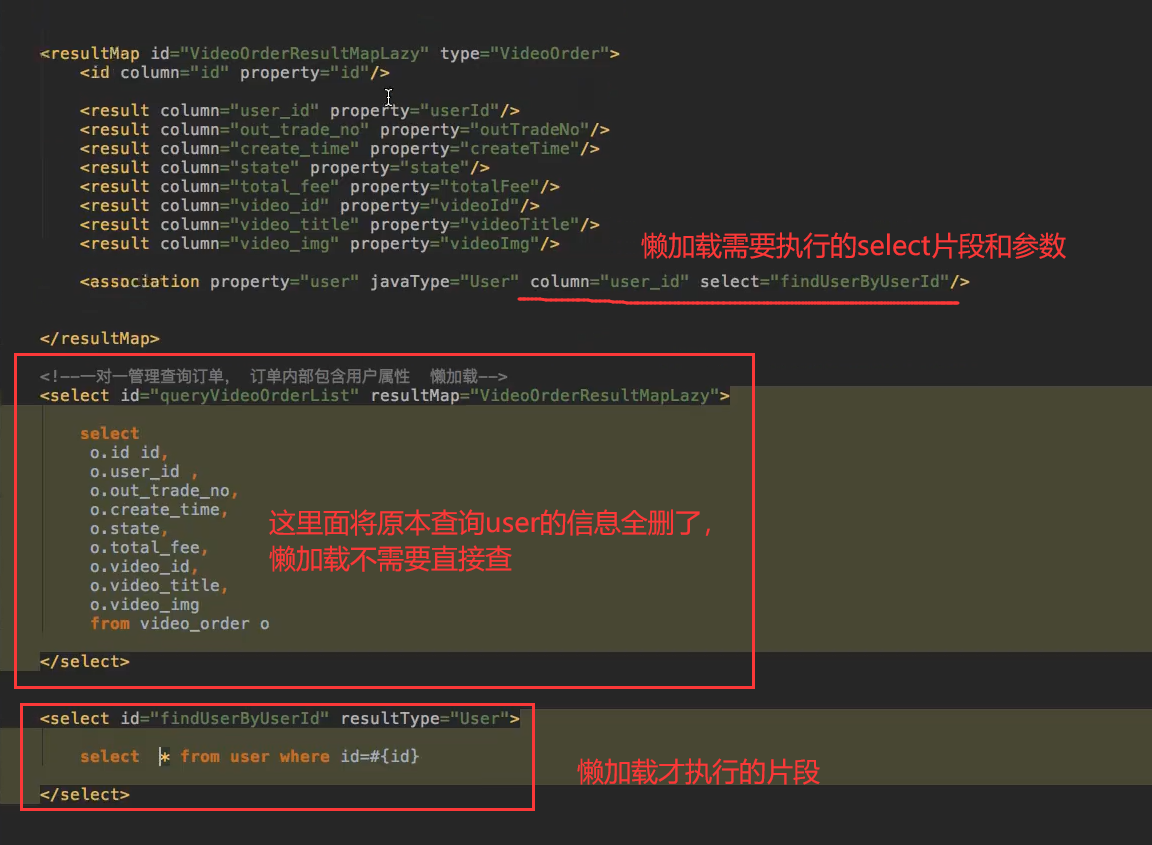
延迟加载:

只有association和collection的地方才需要配置懒加载（这不废话么）



这样在查询一对多的时候，比如查询category的时候并不会吧下面的List<Product>查出来，而是要访问的时候再查，提高查询效率，节省内存。





## 9.Mybatis一键安装:

这个可以说是很方便了，一键安装所有的配置文件。

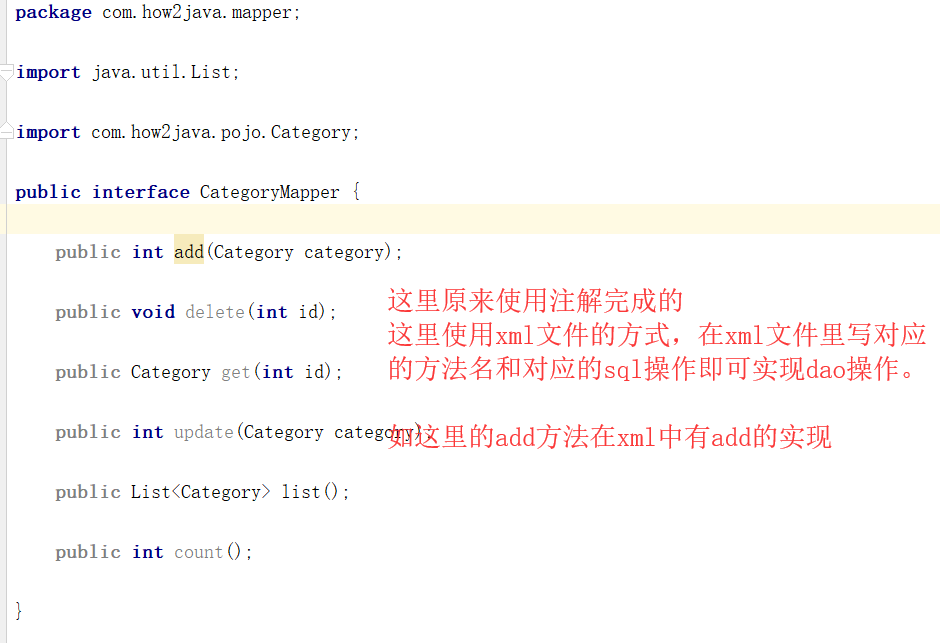
打开一个新project:

<https://how2j.cn/k/mybatis/mybatis-generator/1376.html#nowhere>

导入相应的包走一遍流程，其实按需求来做还不如自己写。反正也是复制粘贴，他提供的模板和自己想要的语句还是会有差距的。

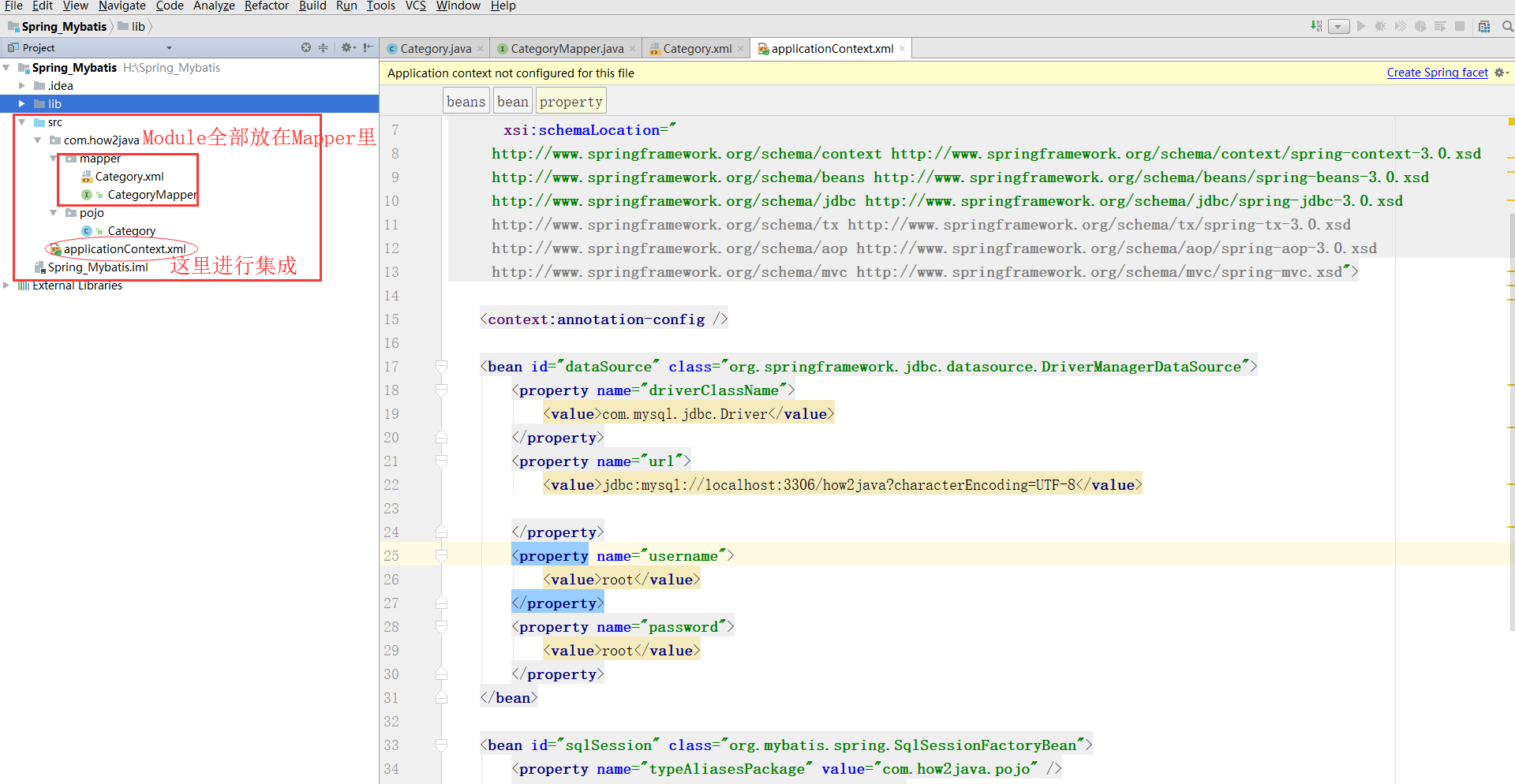
## 10.Spring—Mybatis的集成：

导入一定的包之后，进行集成，主要需要一个配置文件





配置文件：





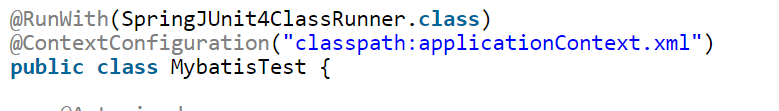
使用就直接使用mapper就行了

原文:

使用[Spring注解方式测试](https://how2j.cn/k/spring/spring-annotation-test/1133.html)，拿到注入的CategoryMapper对象，当调用add方法的时候，会自动去找Category.xml里id="add"的sql语句。

使用：

在类上要加上配置文件的目录





## 11.Mybatis逆向工程：

目前分类管理中Mybatis中相关类都是自己手动编写的，包括：Category.java, CategoryMapper.java和CategoryMapper.xml。  
  
尤其是CategoryMapper.xml里面主要是SQL语句，可以预见在接下来的开发任务中，随着业务逻辑的越来越复杂，SQL语句也会越来越复杂，进而导致开发速度降低，出错率增加，维护成本上升等问题。  
  
为了解决手动编写SQL语句效率低这个问题，我们对Mybatis部分的代码，使用逆向工程进行重构。  
  
所谓的逆向工程，就是在已经存在的数据库表结构基础上，通过工具，自动生成Category.java, CategoryMapper.java和CategoryMapper.xml。  
  
逆向工程使用了 MybatisGenerator这个工具，不熟悉的同学请先参考教程： [Mybatis Generator 逆向工程](https://how2j.cn/k/mybatis/mybatis-generator/1376.html)。

<https://how2j.cn/k/tmall_ssm/tmall_ssm-1547/1547.html#nowhere>

要了解重构和逆向工程的使用，其实就是去掉多余的操作：

这几章看完，所有的pojo，mapper，example和xml都自动生成了。

<https://how2j.cn/k/tmall_ssm/tmall_ssm-1545/1545.html>

重构的自动化构建是基于数据库表结构的。