# Mybatis3 学习

## 一 Mybatis 简介

* MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架。
* 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或注解用于配置和原始映射，将接口和Java的POJO（Plain Old JavaObjects，普通的Java对象）映射成数据库中的记录。

## 二 Mybatis 搭建

Mybatis有两个重要的配置文件

全局配置文件mybatis-config.xml和映射文件XxxMapper.xml(如：EmployeeMapper.xml)数据库配置文件

### 1.配置mybatis-config.xml文件

XML 配置文件中包含了对 MyBatis 系统的核心设置，包括获取数据库连接实例的数据源（DataSource）以及决定事务作用域和控制方式的事务管理器（TransactionManager）

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<environments default="development">

<environment id="development">

<transactionManager type="JDBC"/>

<dataSource type="POOLED">

<property name="driver" value="${driver}"/>

<property name="url" value="${url}"/>

<property name="username" value="${username}"/>

<property name="password" value="${password}"/>

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource="org/mybatis/example/BlogMapper.xml"/>

</mappers>

</configuration>

### 2. XxxMapper.xml（映射器）

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper  
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.test.mybatis.dao.EmployeeMapper">  
   
 <select id="getEmpById" resultType="employee">  
 select \* from student where id = #{id}  
 </select>  
</mapper>

### 3. log4j配置

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

<appender name="STDOUT" class="org.apache.log4j.ConsoleAppender">

<param name="Encoding" value="UTF-8" />

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%-5p %d{MM-dd HH:mm:ss,SSS} %m (%F:%L) \n" />

</layout>

</appender>

<logger name="java.sql">

<level value="debug" />

</logger>

<logger name="org.apache.ibatis">

<level value="info" />

</logger>

<root>

<level value="debug" />

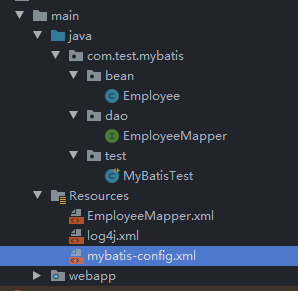
<appender-ref ref="STDOUT" />

</root>

</log4j:configuration>

### 4.测试

结构目录



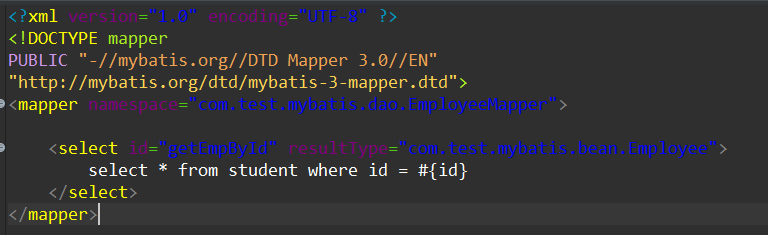
全局配置文件 Mybatis-config.xml配置



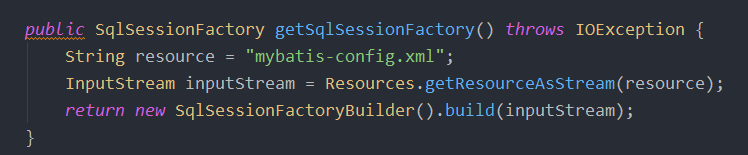
映射文件EmployeeMapper.xml

其中namespace:名称空间,指定为接口的全类名

id:唯一标识  
resultType：返回值类型  
#{id}:从传出过了的参数中取出id值



根据全局配置文件，利用SqlSessionFactoryBuilder 创建SqlSessionFactory



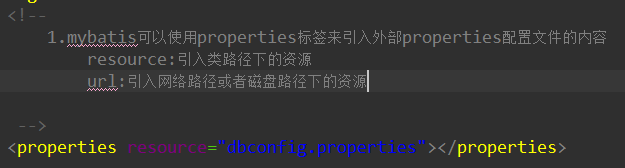


### 4. SqlSession

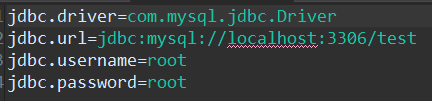
1. sqlSession代表和数据库的一次会话；用完必须关闭释放资源
2. SqlSession他和connection一样都是非线程安全的，不能用成员变量定义
3. SqlSession可以直接调用方法的id进行数据库操作，但是我们一般还是推荐使用SqlSession获取到Dao接口的代理类，执行代理对象的方法，可以更安全的进行类型检查操作。

### 5. Mybatis-全局配置文件

#### 1.properties属性



dbconfig.properties文件



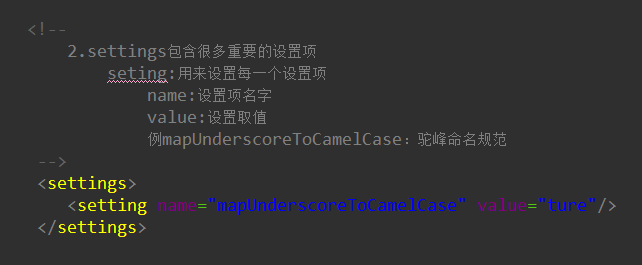
在全局配置文件中通过${}使用



#### 2. settings 设置

这是 MyBatis 中极为重要的调整设置，它们会改变MyBatis 的运行时行为。

常用设置（自动映射、驼峰命名映射、级联规则、是否启动缓存、执行器（Executor）类型等。）



#### typeAliases 类型命名

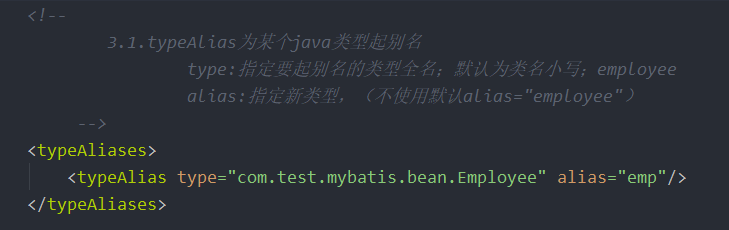
类型别名是为 Java 类型设置一个短的名字，可以方便我们引用某个类。

使用typeAliases:可以为我们的java类型起别名

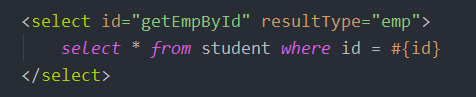
别名不区分大小写

推荐不使用起别名的方式，使用全类名的方式

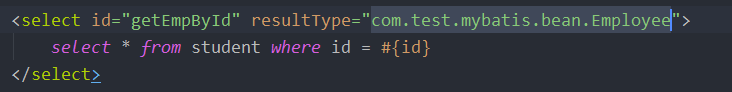
##### 3.1使用typeAlias



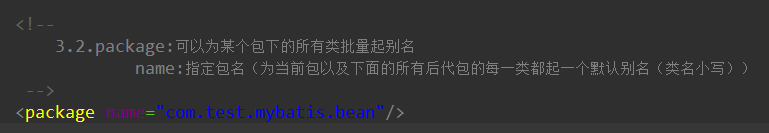
使用typeAliases起别名后映内容为射文件中的



没使用之前，使用全类名

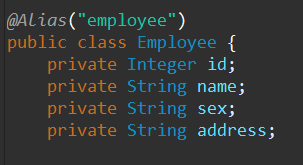


##### 3.2使用package



##### 3.3使用@Alias注解

批量起别名出现类名重复的情况下，使用@Alias注解为某个类型指定新的别名。



### 6. environments 属性

• MyBatis可以配置多种环境，比如开发、测试和生产环境需要有不同的配置。

• 每种环境使用一个environment标签进行配置并指定唯一标识符

• 可以通过environments标签中的default属性指定一个环境的标识符来快速的切换环境（通过default去匹配environment标签中的id属性值）。

Id：指定当前环境的唯一标识符

environment：配置一个具体的环境信息；其中必须包含两个标签

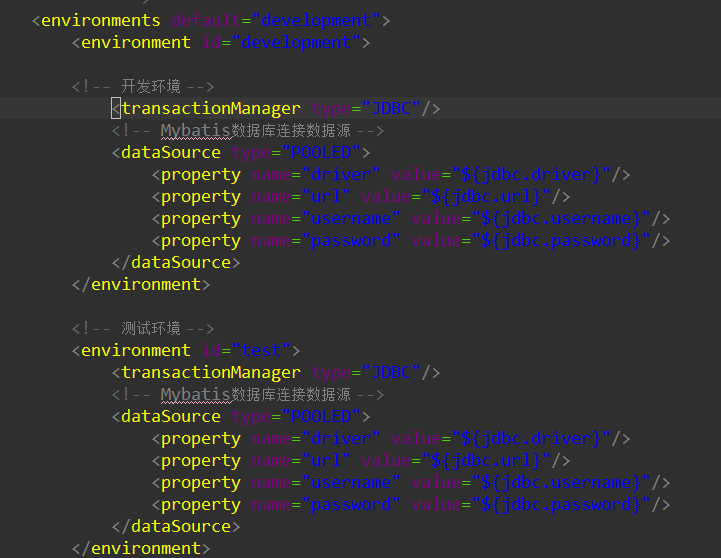
transactionManager：事务管理器

type：事务管理器类型 JDBC|MANAGED

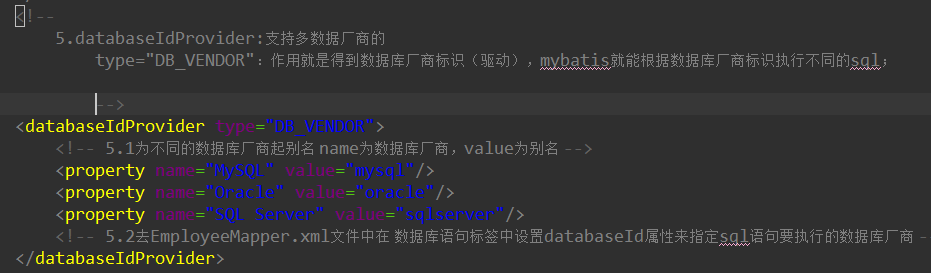
JDBC使用了 JDBC 的提交和回滚设置，依赖于从数据源得到的连接来管理事务范围

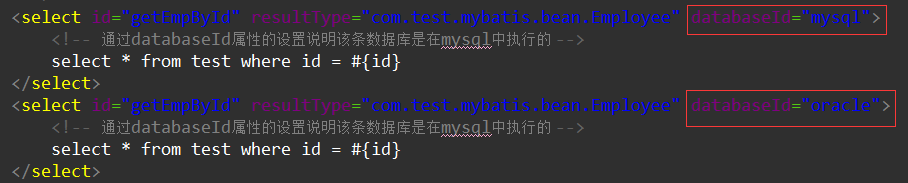
dataSource:数据源类型POOLED(使用连接池PooledDataSourceFactory)

自定义数据源 ：实现DataSourceFactory接口



### 7. databaseIdProvider 数据库厂商标识





### 8. mappers 映射器

将我们写好的sql映射文件（EmployeeMapper.xml）一定要注册到全局配置文件（mybatis-config.xml）中。

使用mappers :将sql映射注册到全局配置中。

mapper:注册一个sql映射

注册配置文件：

resource:引用类路径下的sql映射文件

url:引用网络路径或者磁盘路径下的sql映射文件

注册接口：

class:引用（注册）接口

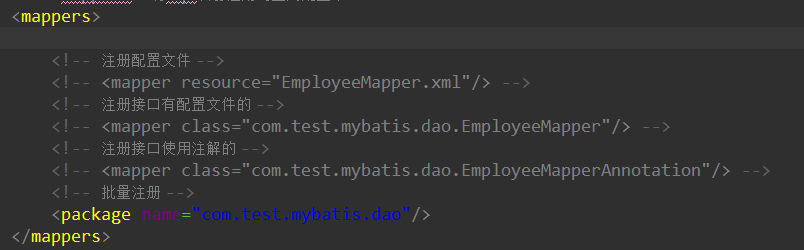
1.有sql映射文件，映射文件必须与接口同名，并且放在与接口同一目录下

2.没有sql映射文件，所有的sql都是利用注解写在接口上

推荐：

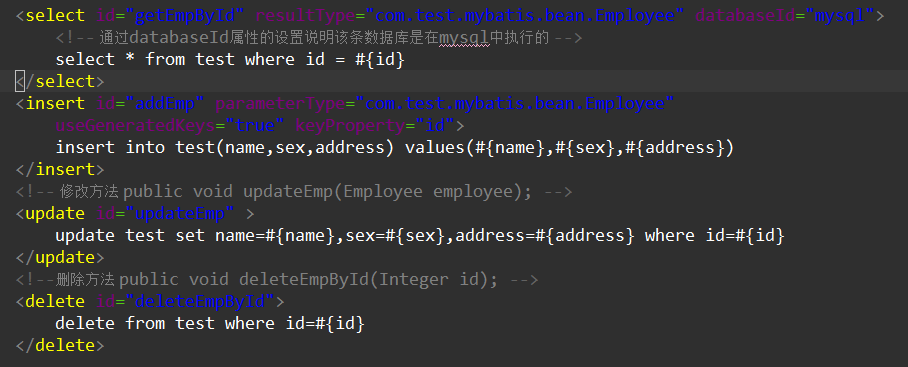
比较重要的，复杂的Dao接口我们来写sql映射文件

不重要的，简单的Dao接口为了开发快速可以使用注解



## 四 MyBatis-映射文件

### 1.对数据库操作的方法



其中：

id:唯一标识

resultType：返回值类型

#{id}:从传出过了的参数中取出id值

parameterType:主要针对将数据存储到数据库中(可以省略不写)

resultType：主要针对从数据库中提取数据（select标签必须有，insert，update，delete可以省略）。

测试test类

查询测试



添加测试addRmp

添加方法比查询方法多一步，手动提交



修改，删除同添加方法，都需要有一个手动提交步骤

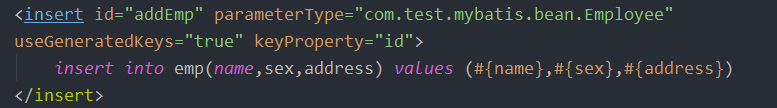
### 2.获取自增主键的值

mysql支持自增主键，自增主键的获取，mybatis也是利用statement.getGenreatedKeys();。

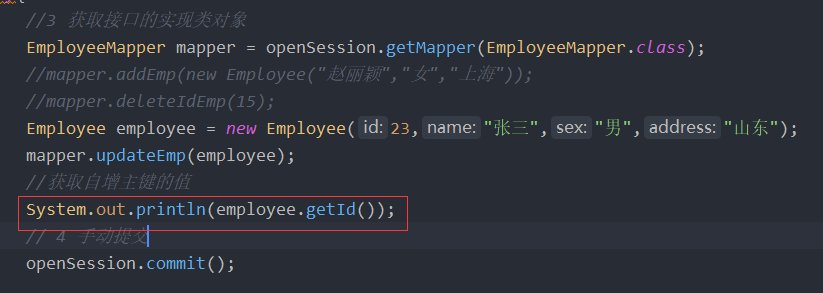
useGeneratedKeys="true" 使用自增键获取主键策略。

keyProperty：指定对应主键属性，也就是mybtish获取到主键值后，将这个值封装给javaBean的那个属性。（前提：数据库 支持自动生成主键的字段（比如 MySQL和 SQL Server））

通过useGeneratedKeys和keyProperty 设置后的insert标签如下



在测试类中，可以直接输出javaBean中封装的id值。



### 3.参数传值

#### 3.1 单个参数

可以接受基本类型，对象类型，集合类型的值。这种情况MyBatis可直接使用这个参数，不需要经过任何处理。

#### 3.2 多个参数

在查询数据时多参数传递。

多个参数：mybatis 会做特殊处理

任意多个参数都会被mybatis封装成一个Map传入。

key：param1....paramN，或者参数的索引

value:传入的参数值

#{}就是从map中获取指定的key的值

正确sql: select \* from test where id=#{param1} and name=#{param2}

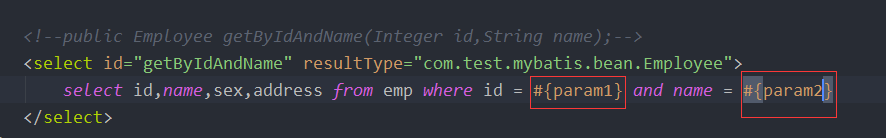
错误sql: select \* from test where id=#{name} and name=#{name}

定义的接口



取值 id🡺#{param1} name🡺#{param2}

映射文件



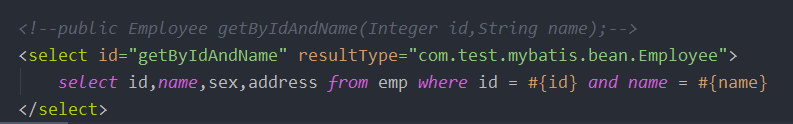
#### 3.3 命名参数

为参数使用@Param起一个名字，MyBatis就会将这些参数封装进map中，key就是我们自己指定的名字。

接口



映射文件



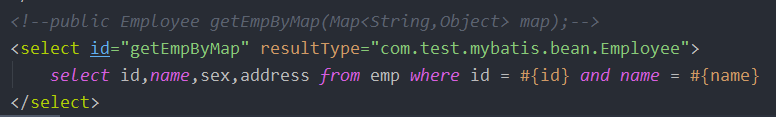
#### 3.4 Map集合

我们还可以直接封装多个参数为map，直接传递

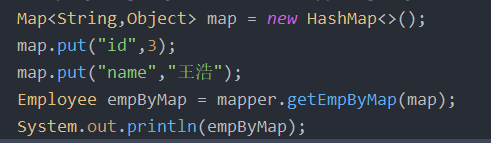
接口



映射文件



测试类



总结：参数多时会封装map，为了不混乱，我们可以使用@Param来指定封装时使用的key；

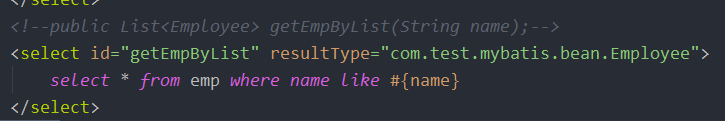
#{key}就可以取出map中的值；

### 4. 查询返回值是一个集合

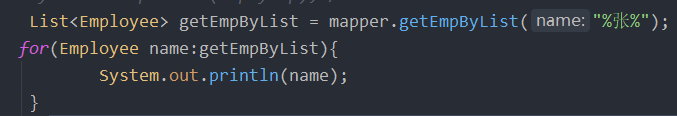
接口



映射文件



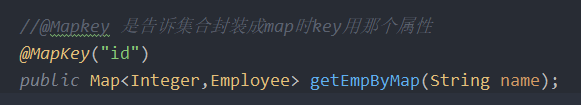
测试方法



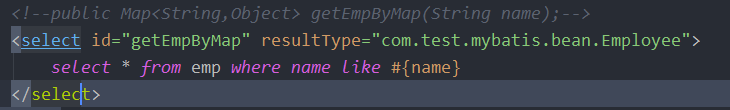
1. 查询结果是一个Map集合

接口

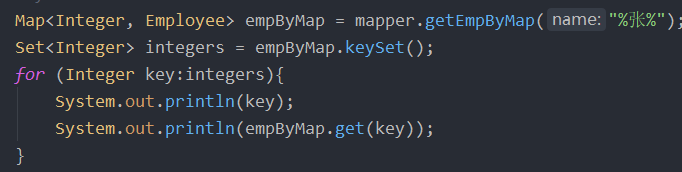
Map中的key是一个数据库中的id属性，value是一个Employee类型

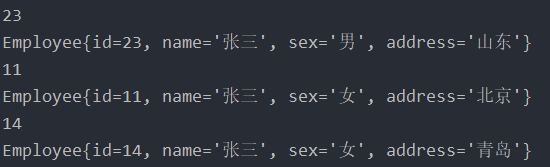


映射文件



测试类



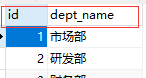


### 5. resultMap

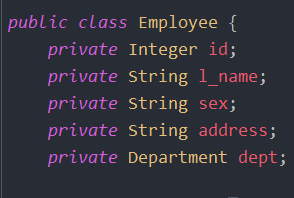
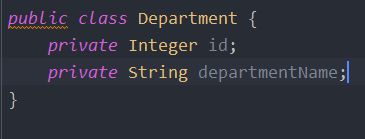
外部resultMap的命名规则，和resultType属性不能同时使用。

resultMap作用：自定义某个javaBean的封装规则

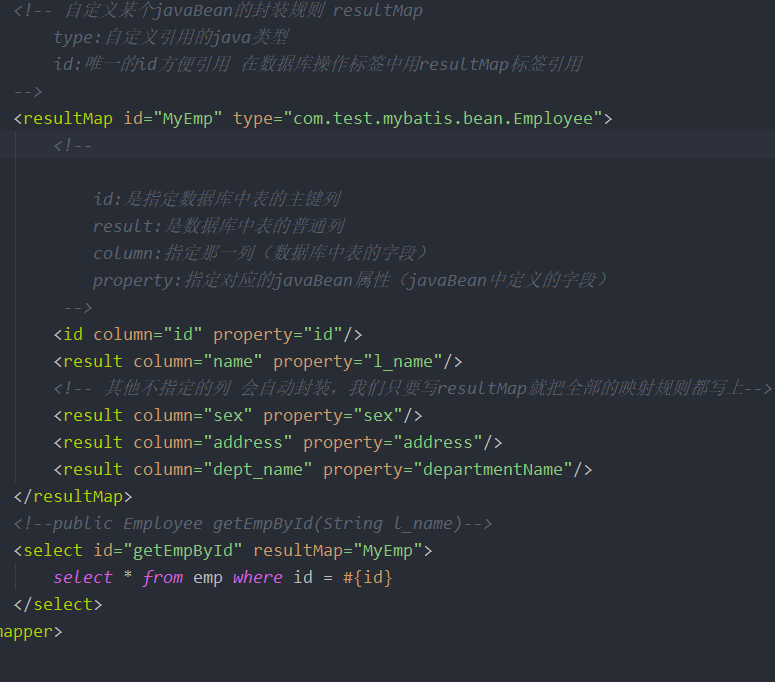
数据库结构（emp表和dept表）



封装类

映射文件



测类结果集



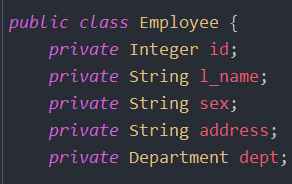
### 多表联合查询

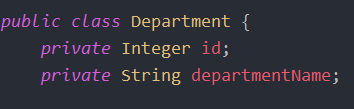
#### 场景一

查询Employee的同时查出员工对应的department  
Employee（员工）==Department（部门）（emp表中的d\_id对应dept表中的id）

方法一：级联属性封装结果集

封装类





映射文件



例如 select e.id e\_id....其中e\_id 是从数据库中查询的结果另起一个新的名字要和column对应。否则会封装失败。

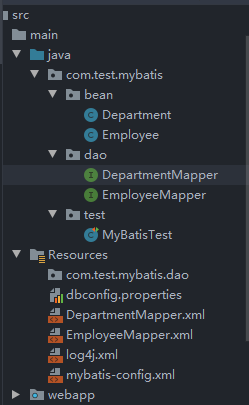
方法二：使用association定义关联的单个对象的封装规则（嵌套结果集）



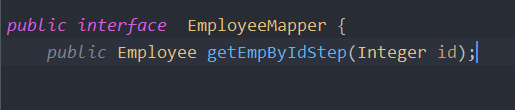
方法三：使用association进行分步查询

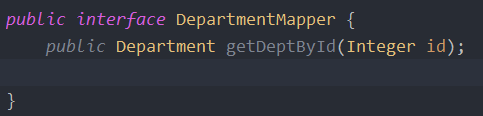
1.按照员工id查询员工信息在emp表中  
2.根据查询的员工信息中的d\_id值去部门表查询部门信息在dept表中  
3.部门设置到员工中

结构

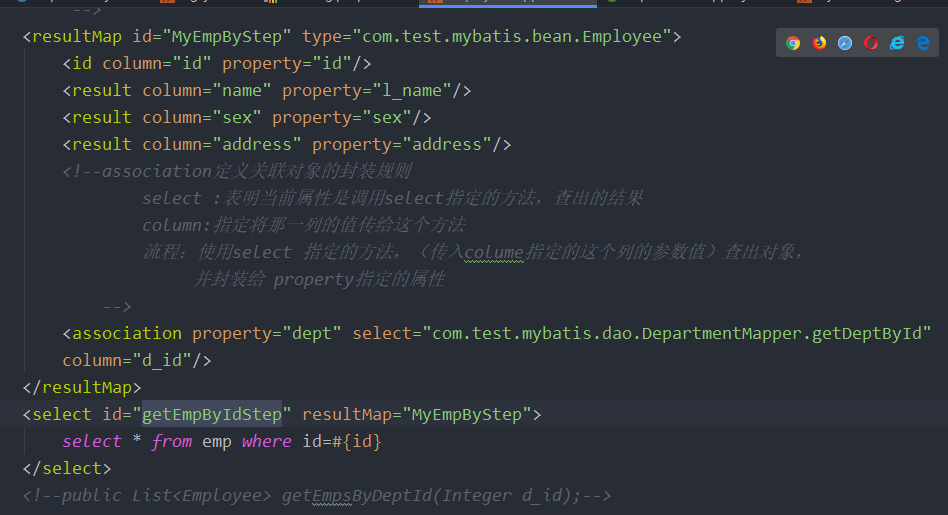


接口



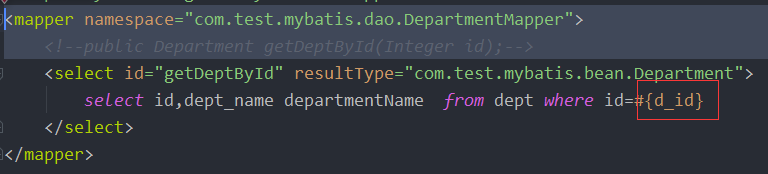


EmployeeMapper接口的映射文件EmployeeMapper.xml

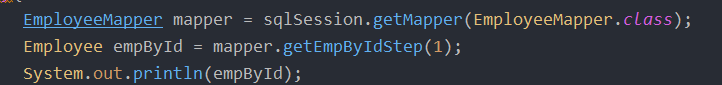


DepartmentMapper接口对应的映射文件DepartmentMapper.xml

EmployeeMapper.xml映射文件中查询的d\_id,通过association调用DepartmentMapper.xml中的方法。查询dept表



测试类



结果集



#### 分段查询&延时加载

分布查询可以使用延时加载

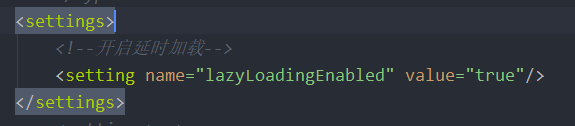
Employee🡺Dapt

我们每次查询Employye对象的时候，都将一起查询出来

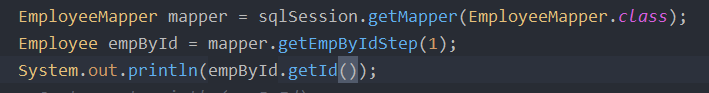
部门信息在我们使用的时候再去查查询；

在分段查询的基础之上 加上两个配置：

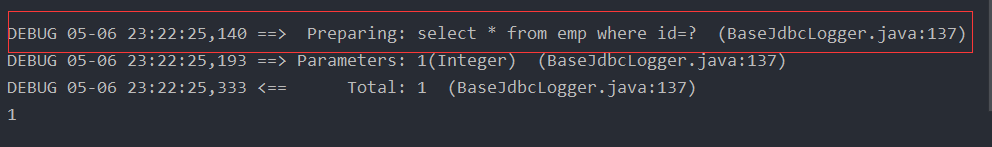
全局配置文件



测试类

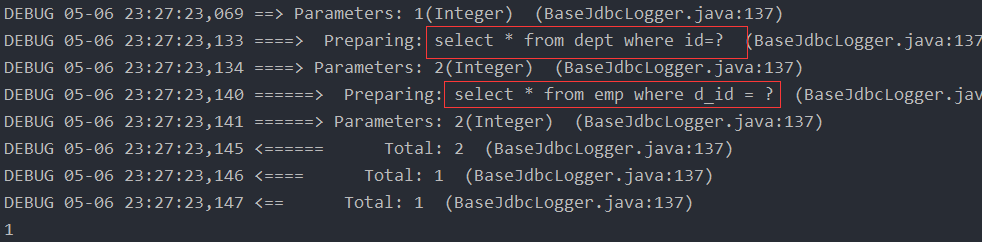


开启延时加载后的结果



开启延时加载后，当我们只输出员工表中的id是，系统只执行一条sql（因为该查询只用到了一条sql语句）。

没开启延时加载的结果



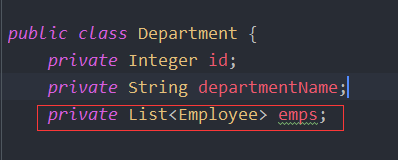
总结：延时加载加载的好处是，当方法中调用相应的数据库，就会去执行该sql语句，否则不执行。

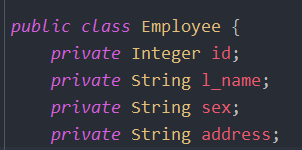
#### 场景二

查询部门的时候将部门对应的所有员工信息也查询出来

方法一 定义关联的单个对象的封装规则（嵌套查询）

封装类



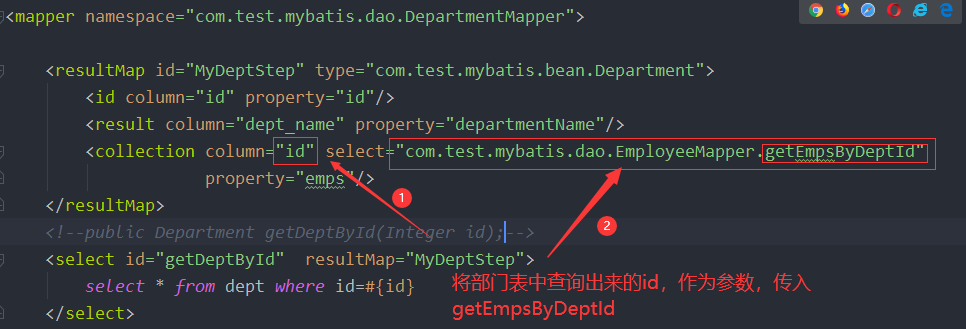


DepartmentMapper.xml映射文件

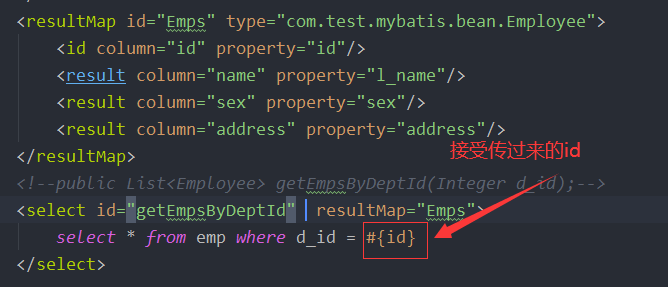


方法二 (分布查询)

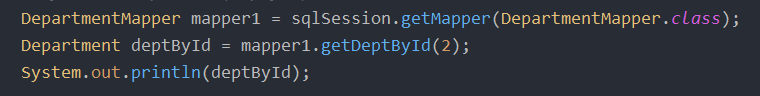
DepartmentMapper.xml映射文件



EmployeeMapper.xml映射文件



测试类



结果





### Mybatis-动态sql

•动态 SQL是MyBatis强大特性之一。极大的简化我们拼装

SQL的操作。

• 动态 SQL 元素和使用 JSTL 或其他类似基于 XML 的文本处

理器相似。

• MyBatis 采用功能强大的基于 OGNL 的表达式来简化操作。

– if

– choose (when, otherwise)

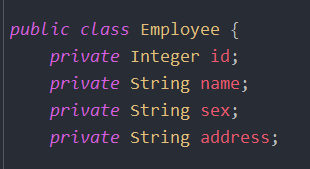
– trim (where, set)

– foreach

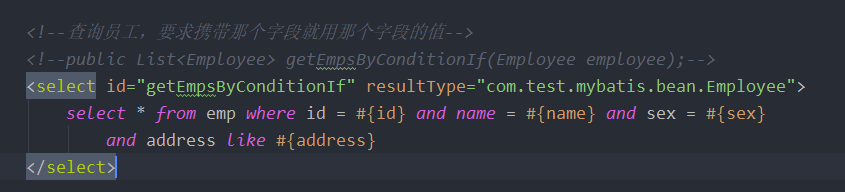
查询员工，要求携带那个字段就用那个字段的值

错误写法

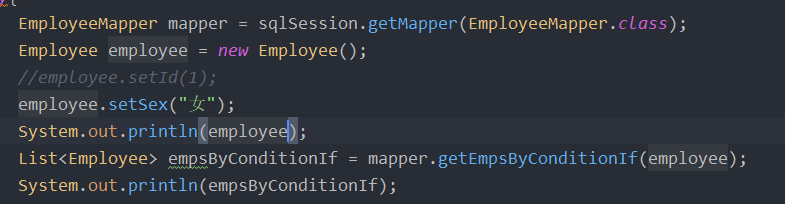
封装类



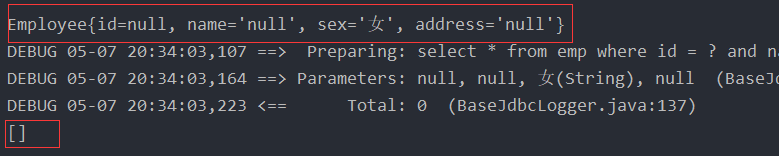
映射文件



测试类



结果



没查出原因是因为含有空值，无法对空值进行处理。

#### 7.1 if语句

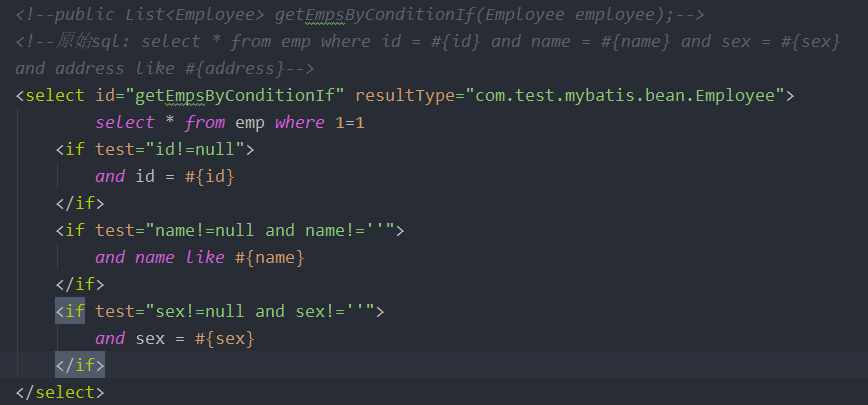
使用if标签是用的一种拼接方法，将sql条件进行拼接

查询的时候如果某些标签条件没有带可能sql拼接会出现问题

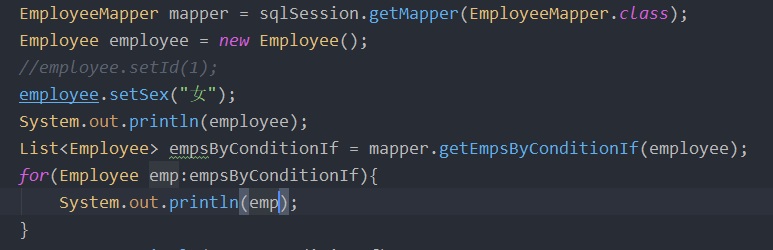
方法一（不使用where标签）

1.给where 后面加上 1=1，后面的条件都用and：例如<if>and....</if>

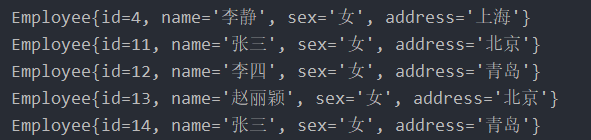
映射文件



测试类



结果集



方法二（使用where标签）

2. 使用where标签将所有的查询条件包括在内mybatis就会将whereb标签中拼装的sql，多出来的andh或者or去掉

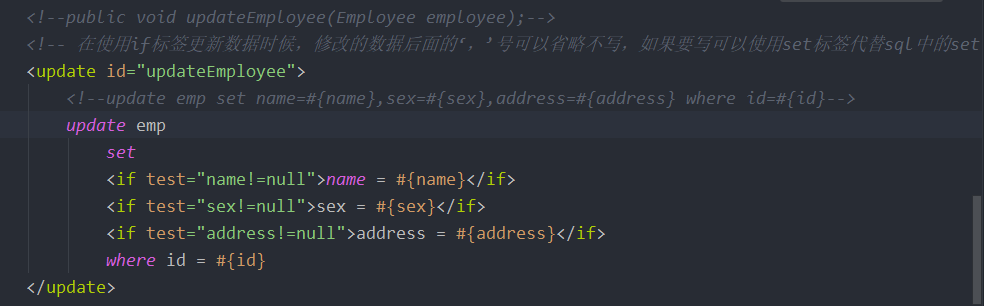
注意：1.使用 where标签 sql语句中的where就不用在写了

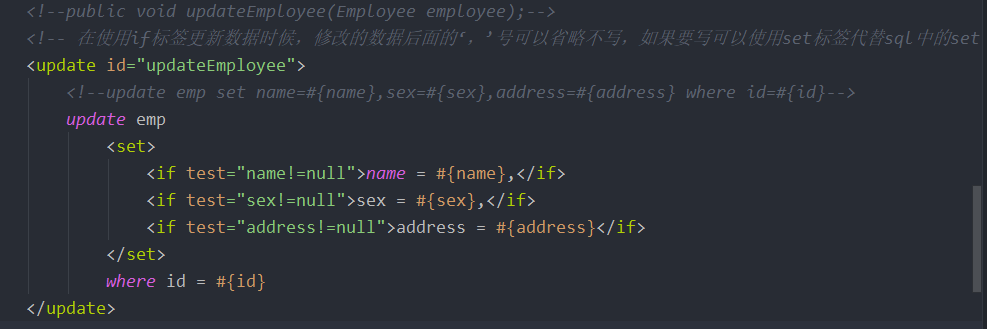
2.where标签只会去掉第一个and或者or（and或者or在条件前面）因为where标签默认拼接字符串带一个默认的and或者or前缀。

映射配置文件如下

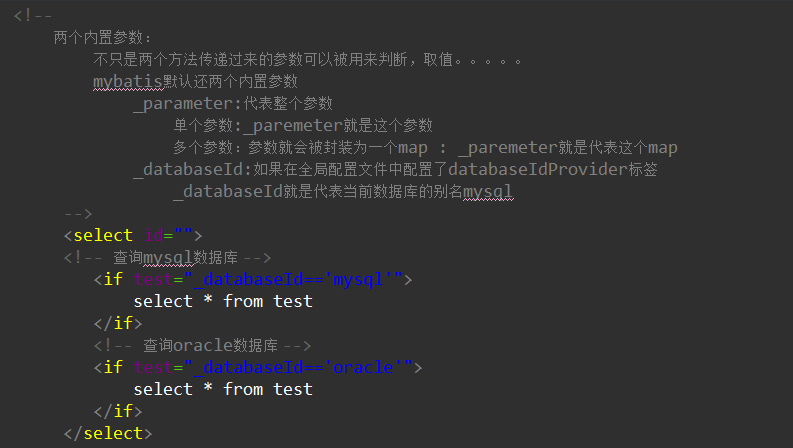


对于修改updae中的set





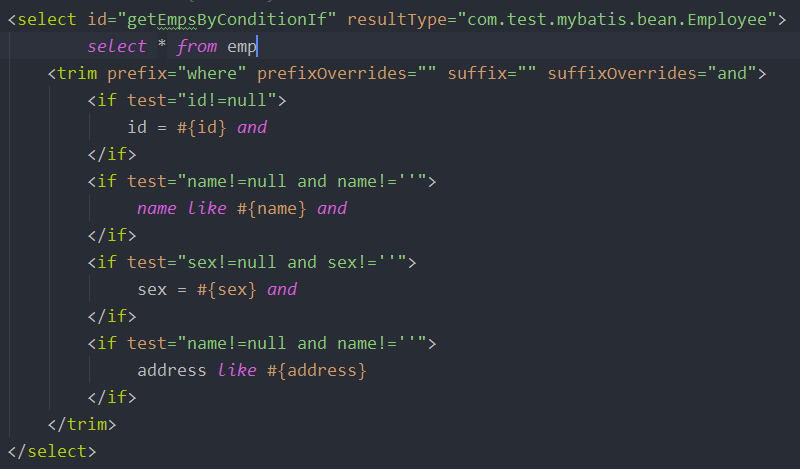
可以选择数据库厂家



#### 7.2 trim

where标签只能解决前面多出的and或者or 后面多除的不能解决这时我们可以使用trim标签  
 prefix=""：前缀：trim标签中是整个字符串拼串后的结果  
 prefix：给拼串后的整个字符串加一个前缀  
 prefixOverrides="":  
 前缀覆盖：去掉整个字符串前面多余的字符  
 suffix=""：后缀  
 suffix：给拼串后的整个字符串加一个后缀  
 suffixOverrides=""  
 后缀覆盖：去掉整个字符串前面多余的字符

映射文件



#### 7.3 choose

* choose：分支选择相当于java中的swtich(条件)-case(选择)(when,otherwise)
* 如果带了id就用id查询，如果带了name就用name查。



#### 7.4 foreach

动态 SQL 的另外一个常用的必要操作是需要对一个集合进行遍历，通常是在构建 IN 条件语句的时候。

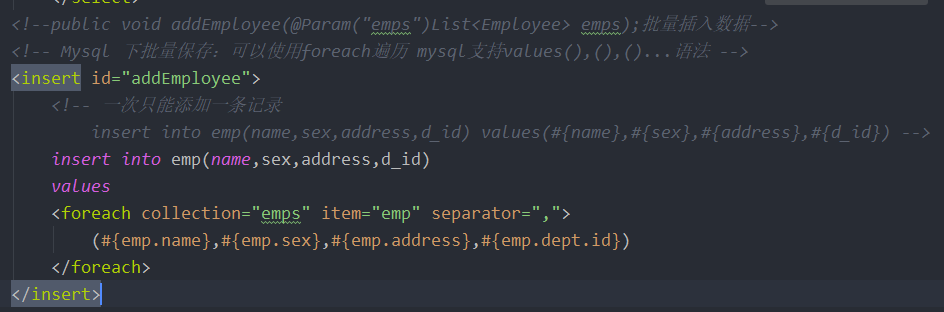
1.查询员工的id在给定的集合当中



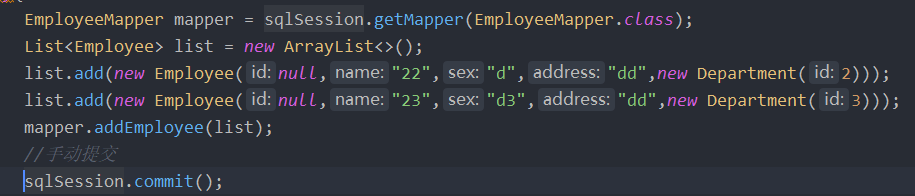
2 批量插入数据方

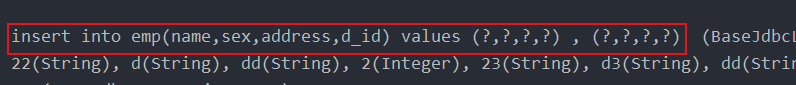
方法一

映射文件

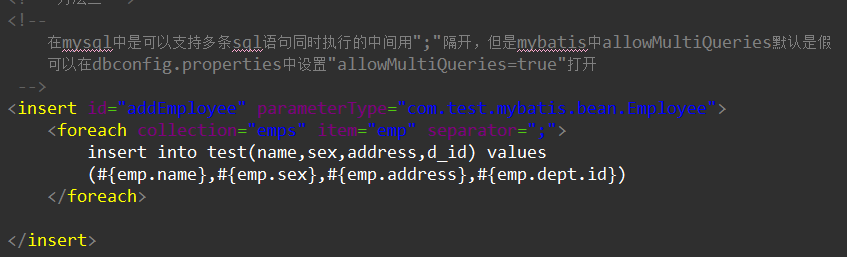


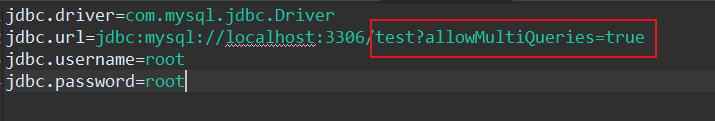
测试类





方法二





## 五 Mybatis中的缓存机制

mybatis默认有两级缓存

**一级缓存：**

（本地缓存）sqlSession级别的缓存。一级缓存是一直开启的，我们没有办法关闭。

什么时候出现一级缓存，例如：当第一次定义一个查询对象去查询后，向数据库发送一条sql。

如果再次查询同一数据，此时就算在定义一个新 对象去向数据库，发送sql查询时。系统直接从第一次查询的缓存中，去数据，而不是再次向数据库发送新的请求。

***作用：***与数据库同一次会话期间查询到的数据库会放在本地缓存中。

以后如果获取相同的数据，直接从缓存中拿，没必要再次去查询数据库；

一级缓存失效（没有使用当前一级缓存的情况，效果就是，还需要再向数据库发出请求）

1.sqlSession不同。

2.sqlSession相同，查询条件不同（当前一级缓存中还没有这条数据）。

3.sqlSession相同，在查询期间，执行了增、删、改操作（因为这次增删改操作可能对本次查询的数据有影响）。

4.sqlSession相同，手动清除了一级缓存（清空缓存后一级缓存中就有没有数据了，使用openSession.clearCachche()清空一级缓存）。

**二级缓存：**

（全局缓存）基于namespace级别的缓存：一个namespace对应一个 二级缓存（namespace是局部变量中mapper标签中的一个属性）

***工作机制：***

1.一个会话，查询一条数据，这个数据就会被放在当前会话的一级缓存中；

2.如果会话关闭；那么一级缓存中数据会被保存到二级缓存中；新的会话可以参照二级缓存中的内容

3.sqlSession==EmplloyeeMapper==>Emplloyee

DeparmentMapper==>Deparment

不同的namespace查出的数据会放在自己对应的缓存中（map）

效果：数据会从二级缓存中获取

查出的数据都会默认放在一级缓存中，只有会话提交或者关闭后，一级缓存才会转移到二级缓存中

***使用方法：***

1.开启全局二级缓存配置：在全局配置文件中开启<setting name="cacheEnabled" value="true"/>

2.去mapper.xml（sql映射文件）中配置使用二级缓存：

<cache></cache>，即在那个mapper中写<cache></cache>，那个mapper就会有二级缓存

3.我们的POJO需要实现序列化接口，即需要用到二级缓存的ben下的实现类继承Serializable

***和缓存有关的设置/属性***

1.cacheEnabled=true;默认false关闭缓存（关闭二级缓存，一级缓存不会关闭一直可用）

2.每个select标签都有useCache="true"；

false：（不使用缓存）一级缓存不会关闭，二级缓存关闭

3.每个增删改标签都有flushCache="true",(默认true)，

增删改执行完成后就会清除缓存

测试flushCache="true"一二级缓存都会被清空

3.查询标签中也有flushCache="false"（默认false）

如果flushCache="true"；每次查询前都会清空缓存，缓存也是没有被使用的

4.sqlSession.clearCache();只是清除当前session的一级缓存

## 六 Mybatis-逆向工程

### 1 配置文件

简介

是一个专门为MyBatis框架使用者定制的代码生成器，可以快速的根据表生成对应的映射文件，接口，以及bean类。支持基本的增删改查，以及QBC风格的条件查询。但是表连接、存储过程等这些复杂sql的定义需要我们手工编写。

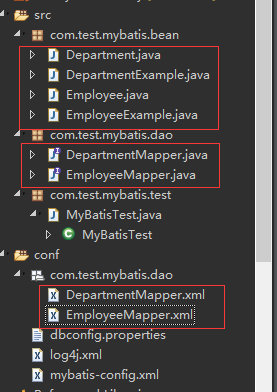
1导包

mybatis-generator-core-1.3.2.jar

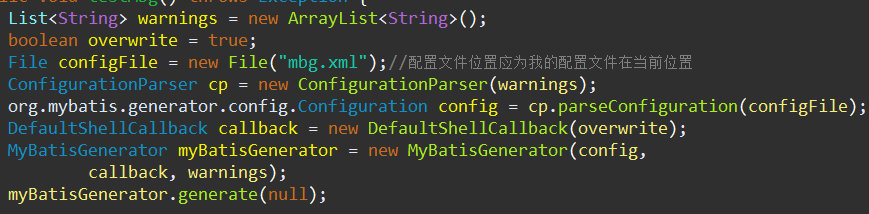
2. 配置文件内容



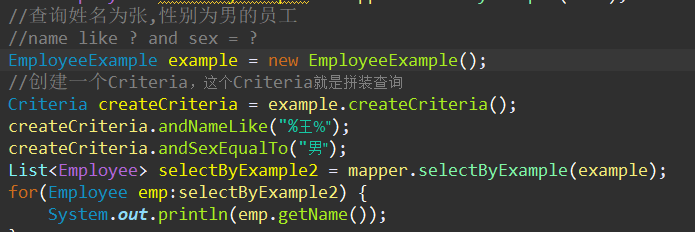
生成的文件包括



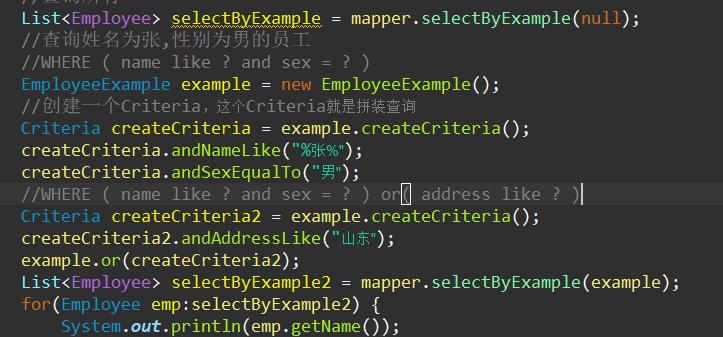
测试类生成器代码



查询姓名为张,性别为男的员工



查询姓名为张,性别为男的员工或住址在山东



## 七 Mybatis 工作原理

