mybatis基础知识

教学思路：

mybatis：基础知识

mybatis是一个java优秀的持久层框架，java中操作关系型 数据库用的是jdbc，mybatis是对jdbc的一个封装。

ORM： Object Relational Mapping (对象，javaBean， Pojo。

RDB， 关系型数据库 。 Mapping ）hibernate 是 一个 完全的 ORM 框架 。Mybatis 是一个 不完全的ORM 框架 。

学习内容

mybatis

1. 熟悉一下 我们以前学习的 JDBC， 主要的目的是挑刺 。
2. mybatis 的架构 （重点）
3. mybatis 的hello world 。（重点）
4. 基于 Dao 持久 层 。（增 CRUD） （重点）（ibatis）
5. 复习一下 junit
6. dao编写， 完成一个 CRUD 的 例子
7. dao的方式进行一个总结。
8. mapper代理方式
9. SqlMapConfig.sql 进行一个完整解读。
10. 输入和输出的映射 。

复习：

1. spring jdbc
2. Jdbc template ( 就是 一个 类似于 dbutils）
3. 数据库连接池的 配置。C3p0.
4. 如何配置 c3p0, spring 如何配置 ？
5. 将 数据库的信息 放到 db.properties 文件。
6. 在 java 程序 。可以 注入吗 ？
7. 使用 jdbc template 开发一个 dao 层代码 ？

AccountDao， jdbcDaoSupport, 如何取得数据连接池 。

Template. Update(). query() queryForObject()

1. Spring 事务管理
2. 事务的 隔离级别： 4.
3. 事务的丢失更新： 如何避免丢失更新 ？
4. 事务的传播级别 。7， require， require\_new nested
5. Jar 包 。 tx， jdbc
6. 事务管理： 是编程式、是声明式（xml，注解）
7. 其他

复习：

1. 什么是mybatis， 什么是 Ibatis ？
2. ORM， mybatis 是 一个不完全的 ORM框架 。
3. Mybatis： sqlMapConfig.xml -》 sqlsessionfactory -》 sqlSession -> executor ->mappedStatement->db
4. Sql, 处理的是 sql语句的 输入 和 输出 。

Mybatis 可以接受的输入类型 有 什么 ？ 基本的java数据类型、javaBean（pojo） + List、Map

1. User.xml:

Namespace: test,

<select insert update delete

2 个缓存相关的

Sql 片段

reulstMap

#{ order.id} ${} 有 什么区别 ？ OGNL， el

1. Ibatis 方式的 dao的开发：

Insert 语句如何取得 自增的主键

1. Mybatis 和 hibernate的区别 ？
2. Mapper的开发方式 。Mapper开发方式 有 4 个要求规范： ？Namespace、mapper命名、interface的方法的输入参数 同 xml里面 ，返回值、方法名称 同 xml里面的id 一样 。
3. sqlSession 里面 有一个方法： getMapper（
4. sqlMapConfig.xml

别名配置。（使用包扫描）

Mapper的 配置 。（可以使用 包扫描）

1. 其他

什么是 MyBatis ？

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。

帮助改进文档...

该文档 xdoc 格式的源码文件可通过[项目的 Git 代码库](https://github.com/mybatis/mybatis-3/tree/master/src/site)来获取。Fork 该源码库，做出更新，然后提交一个 pull request 吧。

# 开发环境

用 maven来管理

mysql 5.1

JDK 1.7

eclipse Photon 4.8 (STS, 写 xml 很方便， spring boot 的支持 是 很好的 ）

<build>

<resources>

<resource>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.tld</include>

</includes>

<filtering>false</filtering>

</resource>

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.tld</include>

</includes>

<filtering>false</filtering>

</resource>

</resources>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-source-plugin</artifactId>

<version>2.2.1</version>

<executions>

<execution>

<goals>

<goal>jar-no-fork</goal>

</goals>

<phase>verify</phase>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.0</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

</build>

## 创建数据库

通常需要提供初始化数据的数据库脚本。

Mysql -u root –p

Create database mybatis01

Use mybatis

source d:/sql/sql\_table.sql

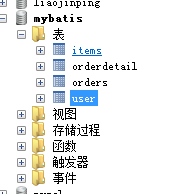
source d:/sql/sql\_data.sql

database : mybatis

CONSTRAINT `FK\_orderdetail\_1` FOREIGN KEY (`orders\_id`) REFERENCES `orders` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `FK\_orderdetail\_2` FOREIGN KEY (`items\_id`) REFERENCES `items` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

如果 不定义， 不影响 我们的关联查询 。





# jdbc编程中问题

企业开发中，根据项目大小、特点进行技术选型 ，jdbc操作数据库时效率是很高的，jdbc也是技术选型的参考。

## jdbc程序

有自己的适应场景，主要是考虑 效率的情况 。使用 JDBC 来操作数据库 。

## jdbc问题总结

问题：

1. 代码长 ，
2. 连接的时候，多次 输入 用户名和密码（连接池）
3. 经常需要关闭资源（连接池）
4. 结果集 需要进行代码处理 才能取得结果
5. 容易被注入 。
6. 不同数据库厂商实现不同的接口 。

问题：

经常需要 关闭资源。 很麻烦 ！

连接池的问题 ， 连接资源经常需要 打开 和 释放，效率很低 。

sql 语句硬编码， 已经进入 java 源程序。

输入的条件也是 硬编码

输出结果也是 硬编码 。

这些都是 jdbc 程序存在的问题 ？

# mybatis架构(重点)

## mybatis介绍 （SSI)= ssm

mybatis， 持久层的架构 。 dao 层 。 ibatis， apache.org 下的一个项目。 mybatis。github.com

MyBatis 本是[apache](https://baike.baidu.com/item/apache" \t "_blank)的一个开源项目[iBatis](https://baike.baidu.com/item/iBatis" \t "_blank), 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。2013年11月迁移到Github。

iBATIS一词来源于“internet”和“abatis”的组合，是一个基于Java的[持久层](https://baike.baidu.com/item/%E6%8C%81%E4%B9%85%E5%B1%82" \t "_blank)框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects（sDAO）

连接数据库、操作数据库。 数据库的SQL 语句、数据库的输出结果 到 DAO 层。 是一个 优秀的持久层框架 。 ORM 。hibernate 是 一个 完全 ORM 实现 。 mybatis 是一个 不完全的 ORM。

ORM: (Object Relational Mapping) (对象关系映射） 面向对象（对象） javaBean

Mysql： rdb， relational database, 关系型数据。

javaBean 对应一条 表记录

List<Bean> 对应一个表， 多表查询， 一对一， 一对多 。

## mybatis架构

SqlMapConfig.xml 全局配置文件（文件名不定）。

mapper.xml 定义 sql 语句 。

SqlSessionFactory . (会话工厂)

SqlSession（ 我们操作数据库的主要接口和实现方法

Executor (执行器) 两种、 普通的执行器、 缓存执行器

mapperStatement（具体操作数据库的对象）（通过 JDBC）

Mysql （database） oracle

普通java类型、pojo类、hashmap

普通java类型、pojo类、hashmap

MapperStatement

SqlSession

SqlSessionFactoryBuilder 类

SqlMapConfig.xml（mapper.xml)

sqlSessionFactory

Executor 执行器

Pojo

基本的数据类型

hashMap

Pojo

集合

基本的数据类型

hashMap

DB

# mybatis入门程序（HelloWorld）

## 需求

实现 用户的查询。

输入用户的id， 查出 用户的信息 。

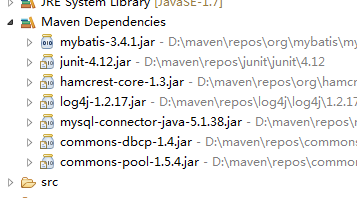
## 导入jar包

从mybatis管网下载（地址：<https://github.com/mybatis/mybatis-3/releases>）

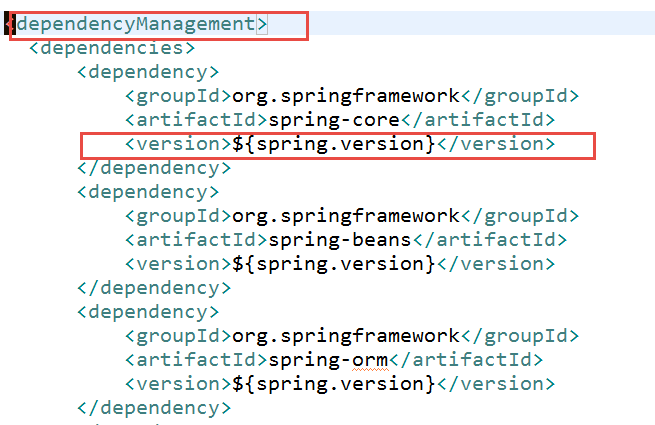
mvnrepository.com

是配置 pom.xml

mybatis， log4j ， mysql-connector

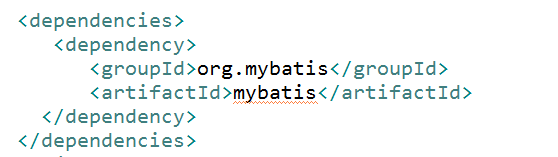


继承 管理版本



继承 管理依赖





<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.hc</groupId>

<artifactId>mybatisday01</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>mybatisday01</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<argLine>-Dfile.encoding=UTF-8</argLine>

<junit.version>4.12</junit.version>

<spring.version>4.3.12.RELEASE</spring.version>

<log4j.version>1.2.17</log4j.version>

<mysql.version>5.1.38</mysql.version>

<mybatis.version>3.4.1</mybatis.version>

<mybatis-spring.version>1.3.1</mybatis-spring.version>

<dbcp.version>1.4</dbcp.version>

<jstl.version>1.2</jstl.version>

<servlet-api.version>2.5</servlet-api.version>

<commons-logging.version>1.1.1</commons-logging.version>

</properties>

<!-- 管理了 jar 包的版本 -->

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>${mybatis-spring.version}</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${log4j.version}</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>${commons-logging.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>${mysql.version}</version>

</dependency>

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>${dbcp.version}</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatis.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>${servlet-api.version}</version>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

</dependency>

</dependencies>

<build>

<resources>

<resource>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.tld</include>

</includes>

<filtering>false</filtering>

</resource>

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.tld</include>

</includes>

<filtering>false</filtering>

</resource>

</resources>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

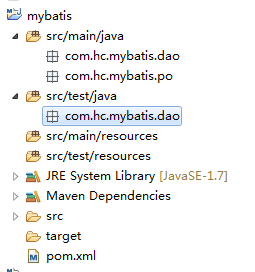
</plugin>

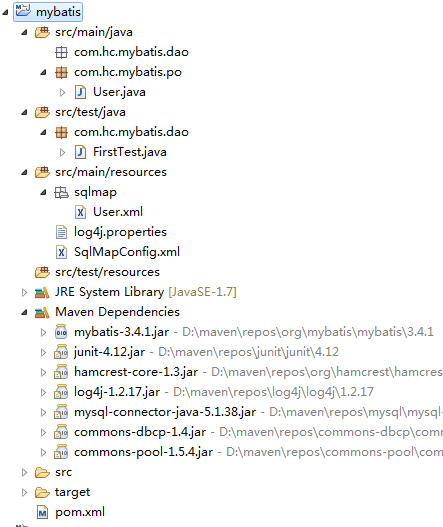
</plugins>

</build>

</project>

## 工程结构





## log4j.properties（公用文件）

# Global logging configuration，建议开发环境中要用debug

log4j.rootLogger=DEBUG, stdout

# Console output...

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n

## SqlMapConfig.xml(公用文件)

通过SqlMapConfig.xml加载mybatis运行环境。

<!DOCTYPE configuration

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

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

<configuration>

<!-- 和spring整合后 environments配置将废除-->

<environments default="development">

<environment id="development">

<!-- 使用jdbc事务管理-->

<transactionManager type="JDBC" />

<!-- 数据库连接池-->

<dataSource type="POOLED">

<property name="driver" value="com.mysql.jdbc.Driver"/>

<property name="url" value="jdbc:mysql://localhost:3306/mybatis?useSSL=false"/>

<property name="username" value="peter"/>

<property name="password" value="peter"/>

</dataSource>

</environment>

</environments>

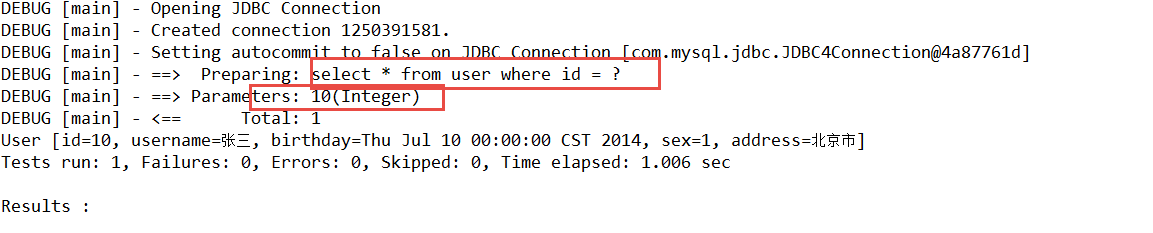
<mappers>

<mapper resource="sqlmap/Student.xml" />

</mappers>

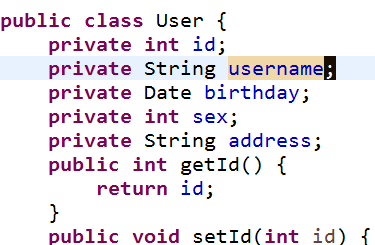
</configuration>





## 根据id查询用户

### pojo（User.java）



### User.xml（重点）

建议命名规则：

表名 + .xml (Ibatis xml 的命名规则)

#{}， 这个 是占位符， 相当于 PreparedStatement 的 ？

源文件：

<?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">

<!-- namespace命名空间，为了对sql语句进行隔离，方便管理 ，mapper开发dao方式，使用namespace有特殊作用

mapper代理开发时将namespace指定为mapper接口的全限定名

-->

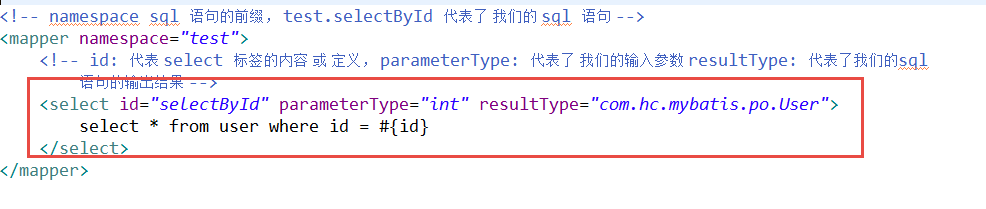
<mapper namespace="test">

<!-- 在mapper.xml文件中配置很多的sql语句，执行每个sql语句时，封装为MappedStatement对象

mapper.xml以statement为单位管理sql语句

-->

</mapper>



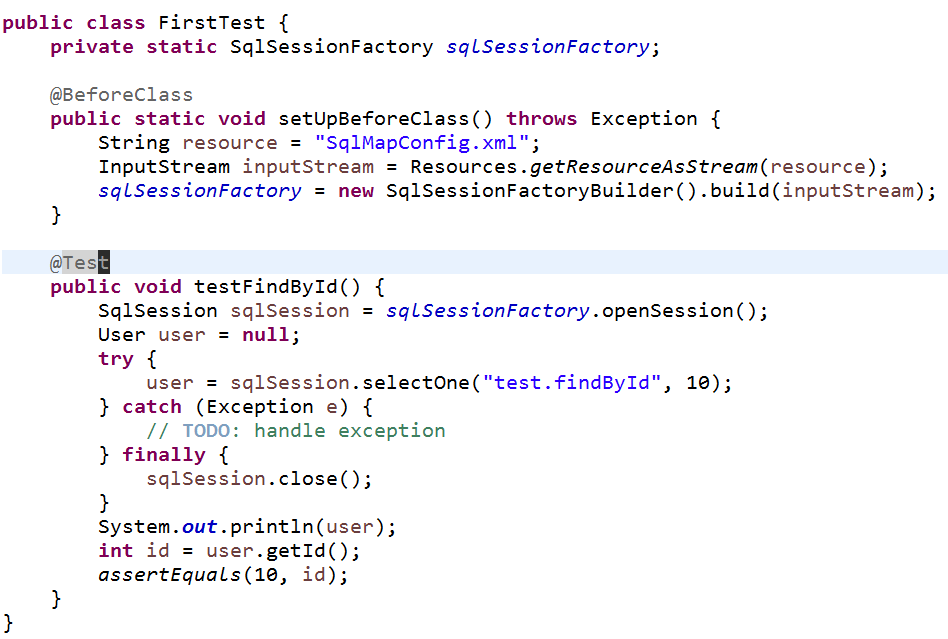
#{id} #{}, ? 相等于 占位符， preparedStatement ?

### 编码

HelloUser.java



创建SqlSessionFactory:



注意 sqlSession 用完后要求 关闭 。

## mybatis开发过程小结

1、编写SqlMapConfig.xml

2、编写User.xml

定义了statement (namespace.statement\_id) selectById sql 语句

3、编程通过配置文件创建SqlSessionFactory

4、通过SqlSessionFactory获取SqlSession

5、通过SqlSession操作数据库

selectOne()

selectList()

如果执行添加、更新、删除需要调用SqlSession.commit()

sqlSession.commit();

6、SqlSesion使用完成要关闭

# 原始Dao的开发方式(Ibatis 开发方式)

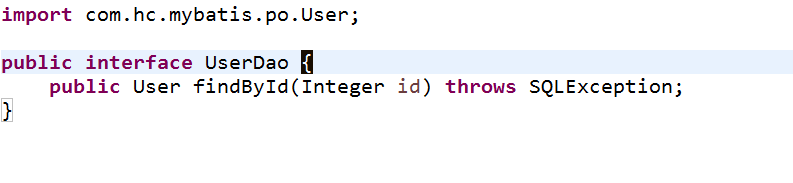
## 需求

查询（ 普通的查询、模糊查询）

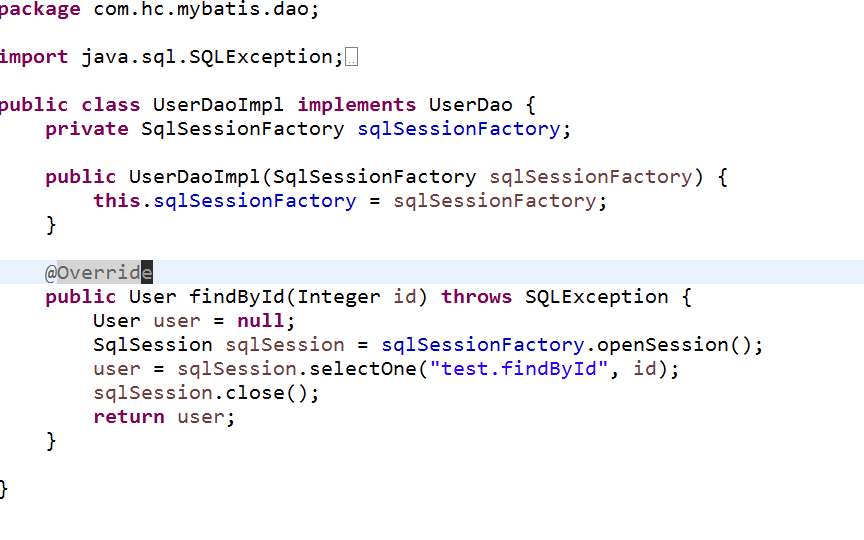
增加、删除、修改

## 复制 selectById 进入我们的 dao 模式

### Dao接口



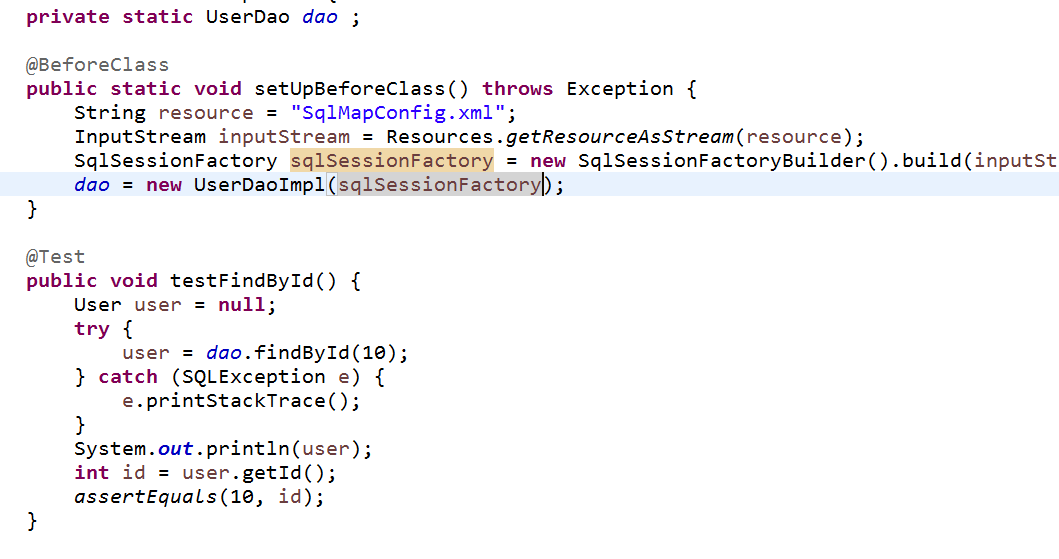
### Dao的接口实现



### 测试代码

初始化代码

测试 代码

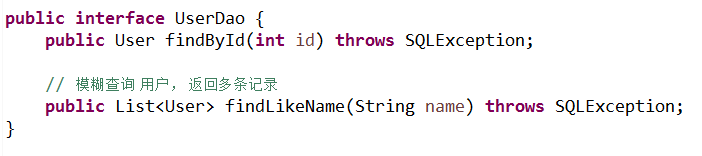


## 根据用户名称模糊查询用户信息

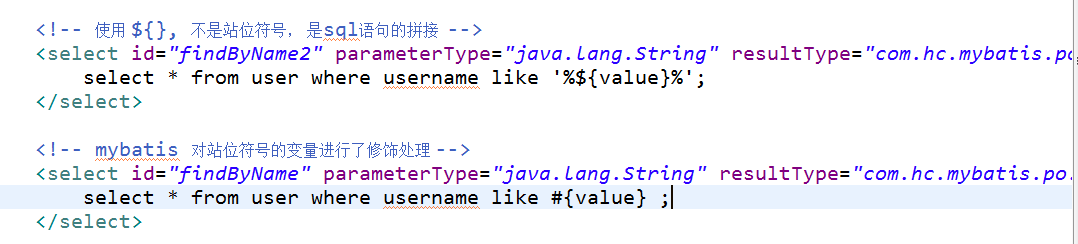
根据用户名称模糊查询用户信息可能返回多条记录。

要查找 名字里面包含 小明的人 。

### 写接口 UserDao.java



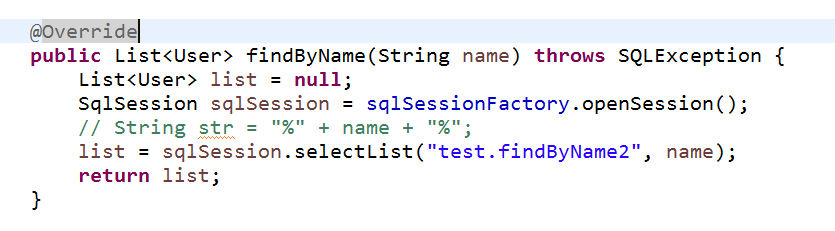
### User.xml



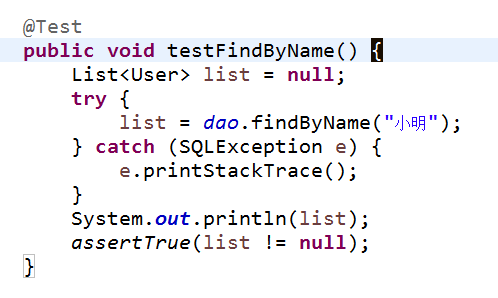
### 编码

selectList() 返回多条记录

selectOne() 返回单条记录



### 测试



### 使用${}接收参数 #{}

不是 占位符， 是 sql 语句的 拼接 ?

${} 符号 不进行 如何串（类型）的处理工作， 不会对串加引号 。

#{username} ？ 站位符号。 ${value} 就是原始值 。

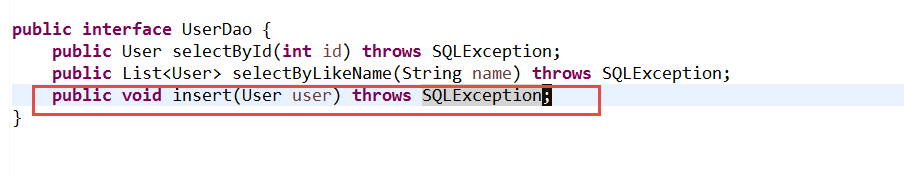
### 编码

### 测试

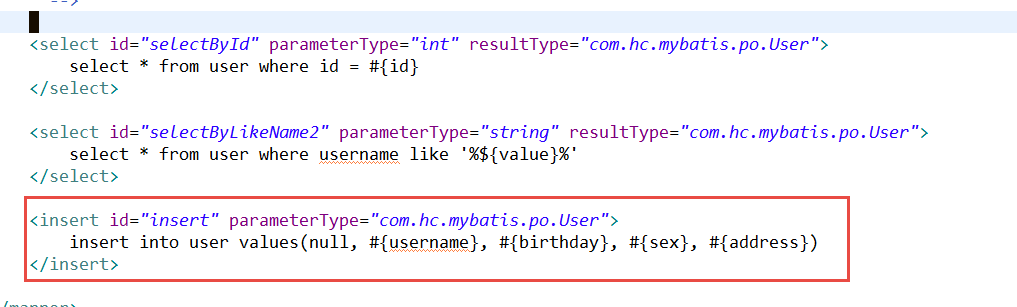
测试代码 同上。

## 用户添加基础

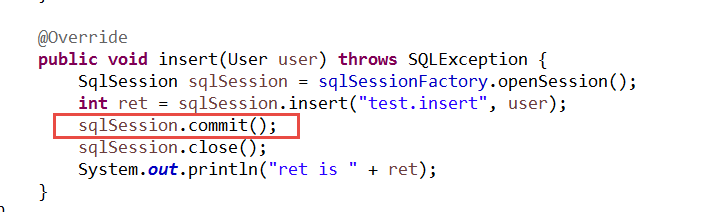
### 修改 UserDao.java



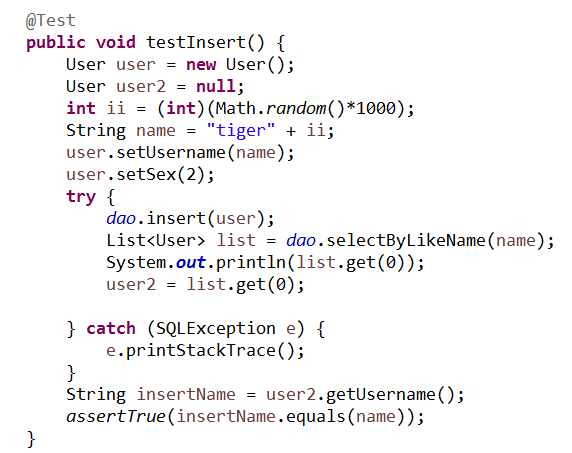
### 修改 User.xml



### 实现 UserDaoImpl.java



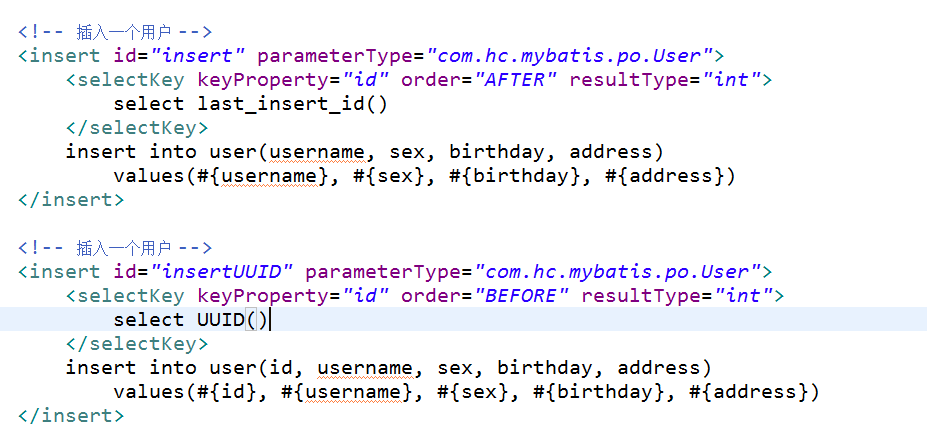
### 写单元测试代码 UserDaoImplTest.java



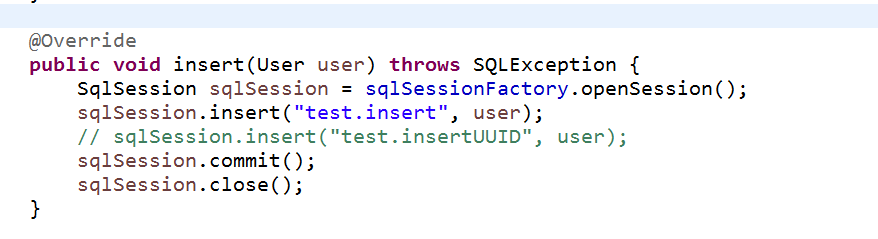
## 用户添加

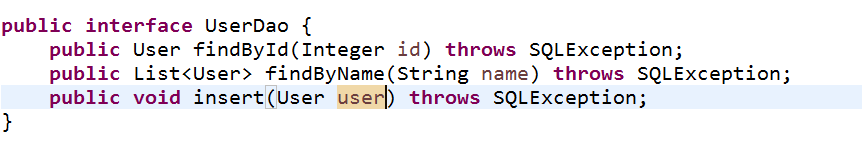
向用户表插入一条记录。

### User.xml

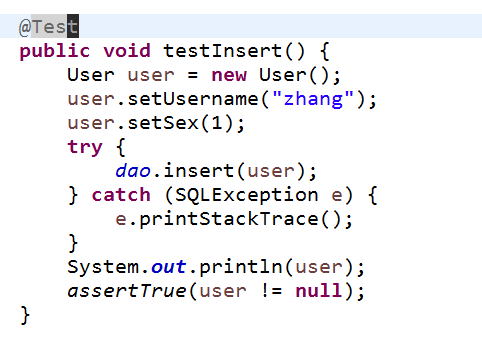


### 编码





### 测试



### 主键返回

需求：user对象插入到数据库后，新记录的主键要通过user对象返回，通过user获取主键值。

解决思路：

通过LAST\_INSERT\_ID()获取刚插入记录的自增主键值，在insert语句执行后，执行select LAST\_INSERT\_ID()就可以获取自增主键。

User.xml修改：

Order 后面的 AFTER 必须大写 。

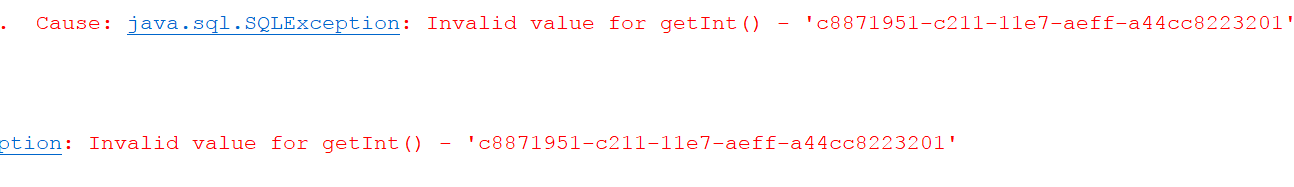
使用mysql的uuid机制生成主键：

使用uuid生成主键的好处是不考虑数据库移植后主键冲突问题。

实现思路：

先查询uuid得到主键，将主键设置到user对象中，将user对象插入数据库。

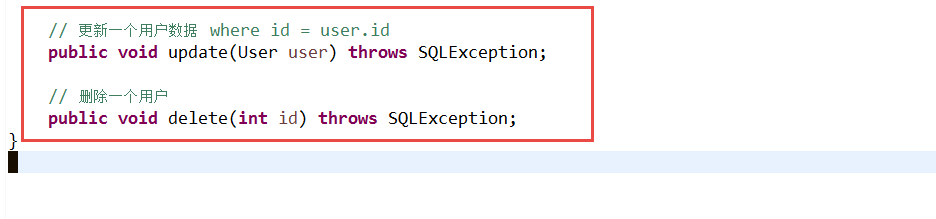
出现错误， 正常 ， 为什么 ？



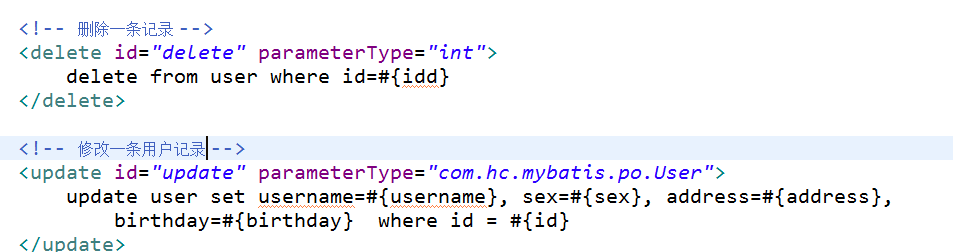
## 用户删除和更新

用户 删除 和 更新的 User.xml 的 编码：

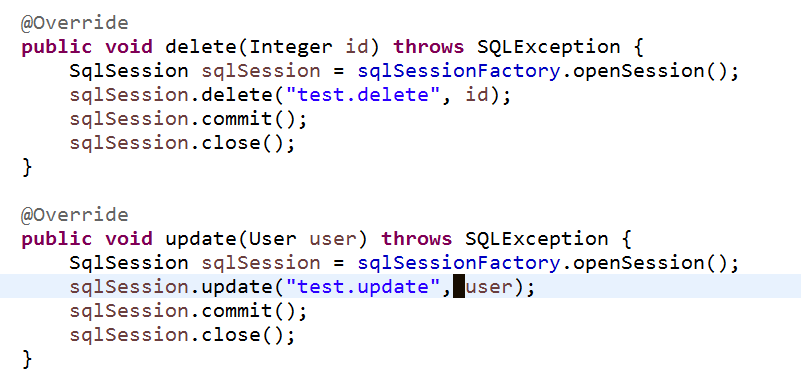
### 编写 interface， UserDao.java



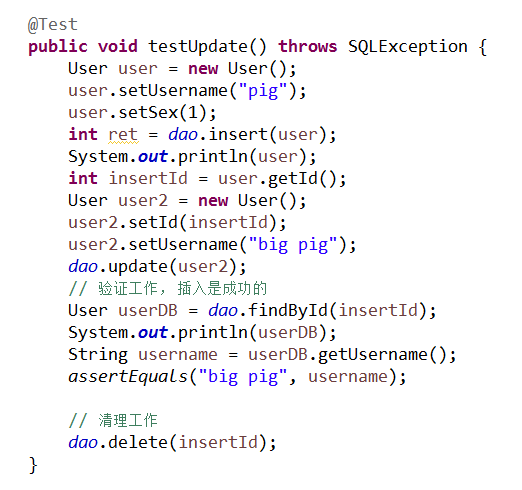
### User.xml



### 编码



### 测试





## Mybatis(Ibatis)解决jdbc编程的问题

1. 数据库链接创建、释放频繁造成系统资源浪费从而影响系统性能，如果使用数据库链接池可解决此问题。

解决：在SqlMapConfig.xml中配置数据链接池，使用连接池管理数据库链接。

1. Sql语句写在代码中造成代码不易维护，实际应用sql变化的可能较大，sql变动需要改变java代码。

解决：将Sql语句配置在XXXXmapper.xml（XXX.xml）文件中与java代码分离。

1. 向sql语句传参数麻烦，因为sql语句的where条件不一定，可能多也可能少，占位符需要和参数一一对应。

解决：Mybatis自动将java对象映射至sql语句，通过statement中的parameterType定义输入参数的类型。(ORM)

1. 对结果集解析麻烦，sql变化导致解析代码变化，且解析前需要遍历，如果能将数据库记录封装成pojo对象解析比较方便。

解决：Mybatis自动将sql执行结果映射至java对象，通过statement中的resultType定义输出结果的类型。

## mybatis与hibernate重要区别

企业开发进行技术选型 ，考虑mybatis与hibernate适用场景。

mybatis：入门简单，程序容易上手开发，节省开发成本 。mybatis需要程序员自己编写sql语句，是一个不完全 的ORM框架，对sql修改和优化非常容易实现 。

mybatis适合开发需求变更频繁的系统，比如：互联网项目。

hibernate：入门门槛高，如果用hibernate写出高性能的程序不容易实现。hibernate不用写sql语句，是一个完全的 ORM框架。

hibernate适合需求固定，对象数据模型稳定，比如：企业OA系统， CRM， HR，ERP

CRM: customer Relationship Managerment . 客户关系管理系统

OA： 办公自动化系统

HR： 人力资源管理系统

ERP: 企业资源整合平台（）

总之，企业在技术选型时根据项目实际情况，以降低成本和提高系统 可维护性为出发点进行技术选型。

## 总结

### SqlMapConfig.xml

是mybatis全局配置文件，只有一个，名称不固定的，主要mapper.xml，mapper.xml中配置 sql语句

### mapper.xml （User.xml)

mapper.xml是以statement（语句）为单位进行配置。（把一个sql称为一个statement），satatement中配置 sql语句、parameterType输入参数类型（完成输入映射）、resultType输出结果类型（完成输出映射）。

还提供了parameterMap配置输入参数类型（过期了，不推荐使用了）

还提供resultMap配置输出结果类型（完成输出映射），明天重点讲通过resultMap完成复杂数据类型的映射（一对多，）

### #{}

表示一个占位符，向占位符输入参数，mybatis自动进行java类型和jdbc类型的转换。

程序员不需要考虑参数的类型，比如：传入字符串，mybatis最终拼接好的sql就是参数两边加单引号。

#{}接收pojo数据，可以使用OGNL解析出pojo的属性值，

sql 语句里面， 变成了 一个 ？

### ${}

表示sql的拼接，通过${}接收参数，将参数的内容不加任何修饰拼接在sql中。

${}也可以接收pojo数据，可以使用OGNL解析出pojo的属性值，里面的内容有要求。

缺点：不能防止sql注入。

### selectOne

用于查询单条记录，不能用于查询多条记录，否则异常：

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

## selectList

用于查询多条记录，可以用于查询单条记录的。

# mybatis开发dao的方法

## SqlSession作用范围

是使用局部变量、成员变量。。。。？？？

### SqlSessionFactoryBuilder

一次性使用， 用完就不用考虑 。 new

### sqlSessionFactory

用它产生 sqlSession， 产生一次， 以一个单例模式 。多次使用 。

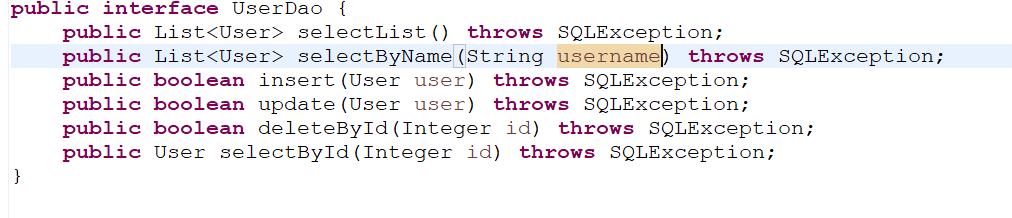
### SqlSession

每次的 数据库操作都要使用 。而且 sqlSession 不是线程安全 。使用范围就是 我们的方法内部。在里面 创建 和 关闭 。

## 原始dao开发方式(ibatis)

程序员需要写dao接口和dao 的实现 类

### dao接口



### dao接口实现

### 测试代码

## mapper代理的方式

程序员只需要写dao接口，dao接口实现对象由mybatis自动生成代理对象。

本身dao在三层架构中就是一个通用的接口。

### 上边原始dao开发方式的问题

1 dao的实现类中存在重复代码，整个mybatis操作的过程代码模板重复（先创建sqlsession、调用sqlsession的方法、关闭sqlsession）

2、dao的实现 类中存在硬编码，调用sqlsession方法时将statement的id硬编码。

### mapper开发规范

要想让mybatis自动创建dao接口实现类的代理对象，必须遵循一些规则：

1. mapper.xml中namespace指定为mapper接口的全路径名

此步骤目的：通过mapper.xml和mapper.java进行关联。

2、mapper.xml中statement的id就是mapper.java中方法名

3、mapper.xml中statement的parameterType和mapper.java中方法输入参数类型一致

4、mapper.xml中statement的resultType和mapper.java中方法返回值类型一致.

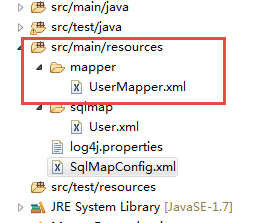
### mapper.xml（映射文件）

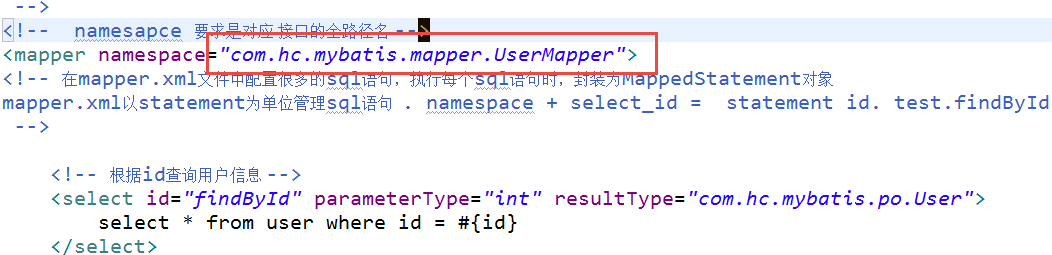
mapper映射文件的命名方式建议：表名Mapper.xml

ibatis: User.xml

mapper: UserMapper.xml

namespace指定为mapper接口的全限定名



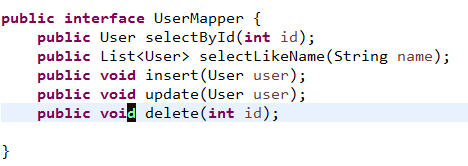


### mapper接口

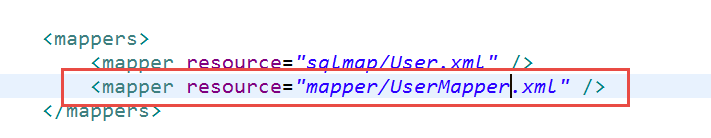
mybatis提出了mapper接口，相当 于dao 接口。

mapper接口的命名方式建议：表名Mapper

UserMapper.java

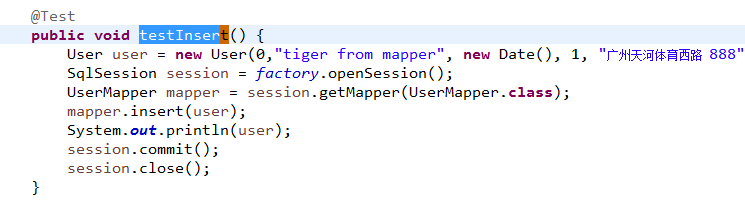


### 将mapper.xml在SqlMapConfig.xml中加载



### 测试

Update, insert, delete 语句 不要 忘记 commit。当然 sqlSession 也必须close。

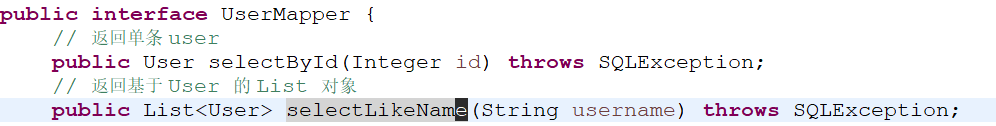


1. selectList（）
2. list.size() = 1, return User ,
3. size() > 1 return List<User>

### mapper接口返回单个对象和集合对象

不管查询记录是单条还是多条，在 statement中resultType定义一致，都是单条记录映射的pojo类型。

mapper接口方法返回值，如果是返回的单个对象，返回值类型是pojo类型，生成的代理对象内部通过selectOne获取记录，如果返回值类型是集合对象，生成的代理对象内部通过selectList获取记录。



### 问题

#### 返回值的问题

如果方法调用的statement，返回是多条记录，而mapper.java方法的返回值为pojo，此时代理对象通过selectOne调用，由于返回多条记录，所以报错：

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

#### 输入参数的问题（可以解决）

使用mapper代理的方式开发，mapper接口方法输入 参数只有一个，可扩展性是否很差？？

可扩展性没有问题，因为dao层就是通用的，可以通过扩展pojo（定义pojo包装类型）将不同的参数（可以是pojo也可以简单类型）传入进去。

# sqlMapConfig.xml

SqlMapConfig.xml中配置的内容和顺序如下：

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属性定义

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。

建议使用properties，不要在properties中定义属性，只引用定义的properties文件中属性，

并且properties文件中定义的key要有一些特殊的规则。



## settings全局参数配置

ibatis的全局配置参数中包括很多的性能参数（最大线程数，最大待时间。。。），通过调整这些性能参数使ibatis达到高性能的运行，mybatis没有这些性能参数，由mybatis自动调节。

## typeAliases(常用)

类型是 别名。

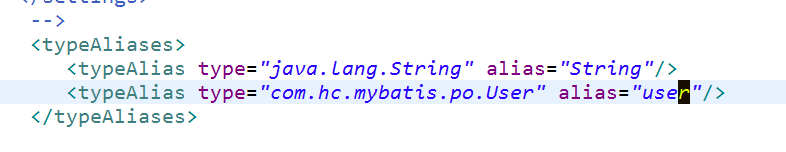
可以将parameterType、resultType中指定的类型 通过别名引用。

### mybaits提供了很多别名

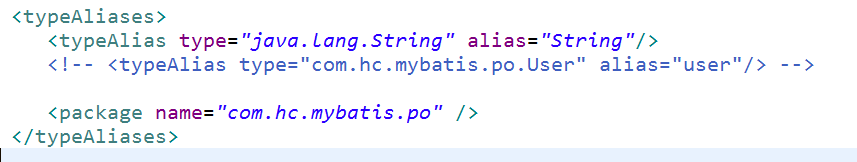
|  |  |
| --- | --- |
| 别名 | 映射的类型 |
| \_byte | byte |
| \_long | long |
| \_short | short |
| \_int | int |
| \_integer | int |
| \_double | double |
| \_float | float |
| \_boolean | boolean |
| string | String |
| byte | Byte |
| long | Long |
| short | Short |
| int | Integer |
| integer | Integer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

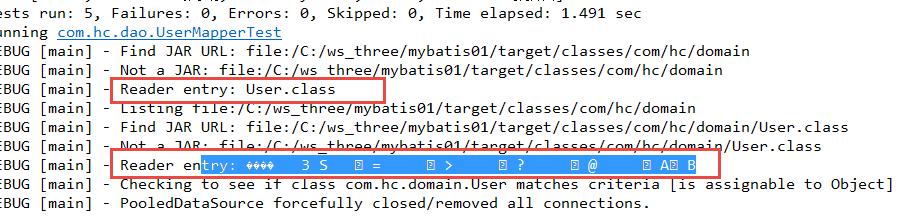
### 自定义别名

自定义别名：



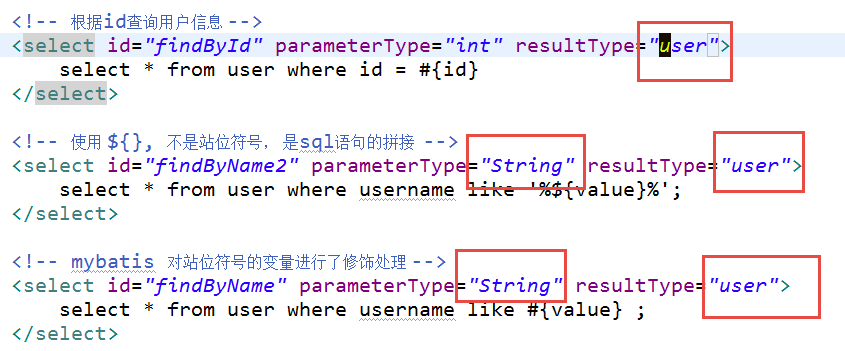
自定义别名包扫描：





### 使用别名

在parameterType、resultType中使用别名：



## typeHandlers

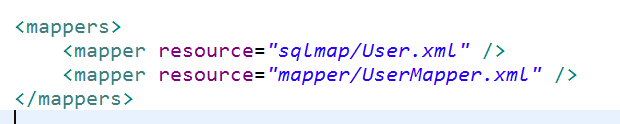
类型处理器将java类型和jdbc类型进行映射。

mybatis默认提供很多类型处理器，一般情况下够用了。



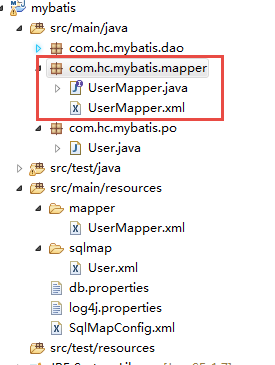
。。。

## mappers



#### mapper的 class 配置

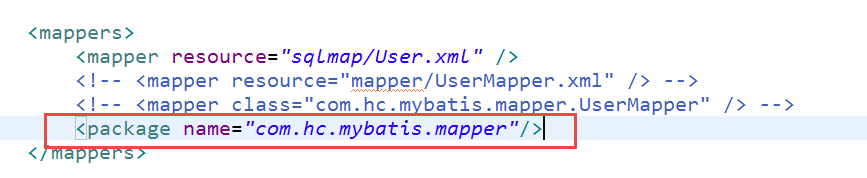
XXXMapper.java 和 XXXMapper.xml 前面同名并且 都在一个目录 下。 一般在 java 的 mapper包的目录下



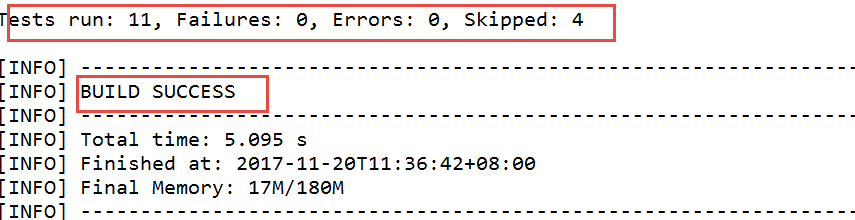


#### mappers 的 package 配置

学习 一下 mappers 里面的 package 的用法， 如果 XXXMapper.java 和 XXXMapper.xml 前面同名并且 都在一个目录 下。 一般在 java 的 mapper包的目录下 。就可以 使用 package来扫描 。这适合 mapper 就不需要 配置了 。



测试： mvn clean compile

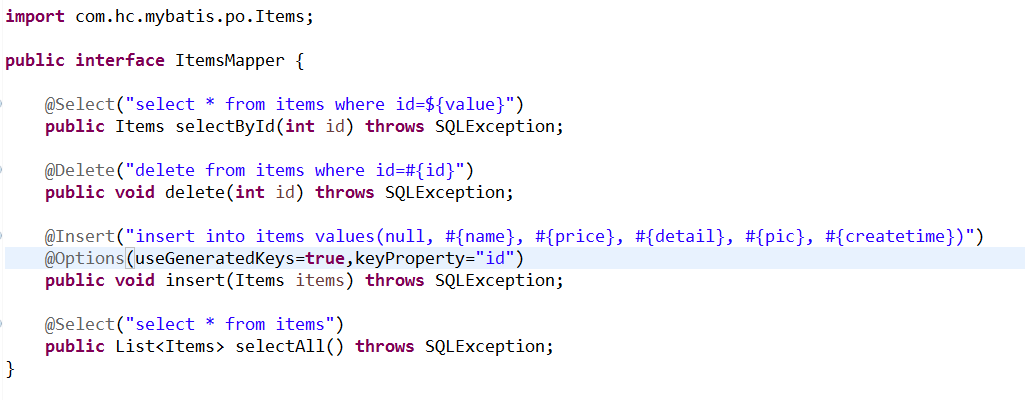


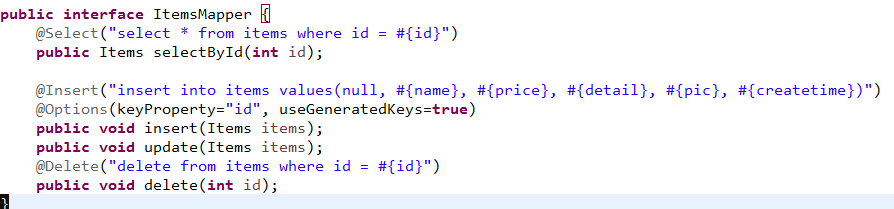
# Mybatis 之使用 注解开发（了解，掌握一下）

Mybatis 的 特点就是 sql 语句的灵活性，注解开发不推荐， 大家可以 了解一下。

## 编码

ItemsMapper.java



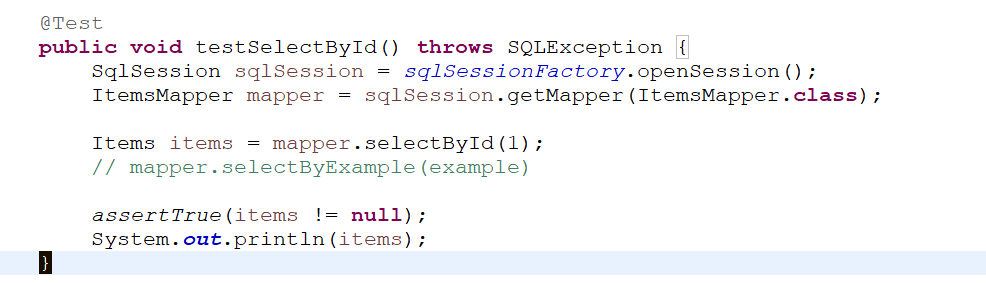


复杂的例子：

## 测试

ItemsMapperTest.java

同普通 mapper 的测试代码 差别不大





# 输入和输出映射（重点）

通过parameterType完成输入映射，通过resultType和resultMap完成输出映射。

## parameterType 对象的扩充

### 需求

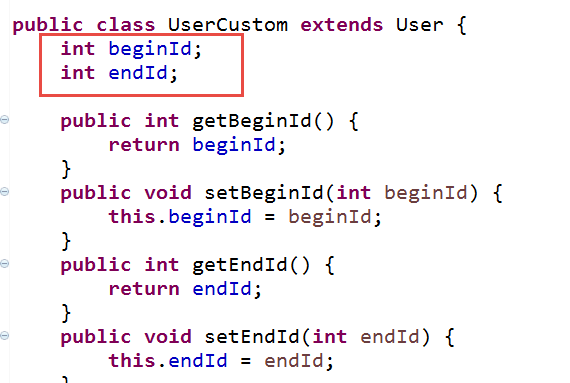
如果我们计划查询 id 在 20 到 40 之间的 用户，sql：

Select \* from user where id > 20 and id < 40

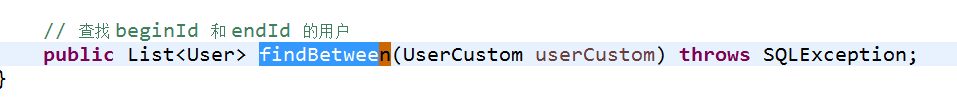
这里面 需要 输入 2 个参数 beginId and endId .

### 扩展 pojo User.java 的类型

创建一个 UserCustom.java pojo

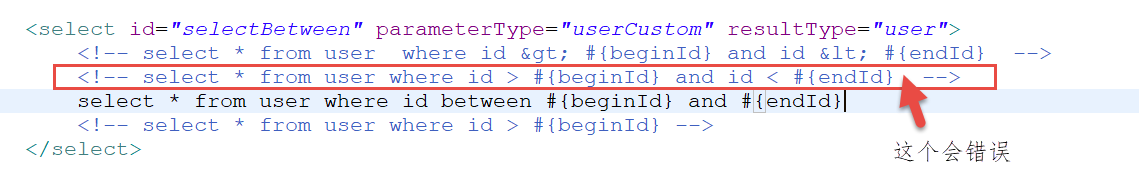


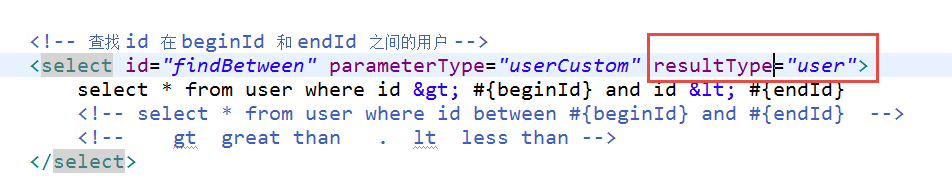
### 修改 UserMapper.java



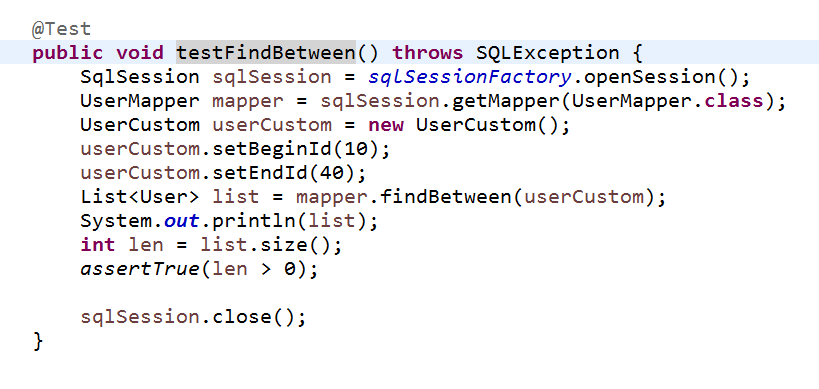
### 修改 UserMapper.xml

其他的 都 是 正确的。 为什么 会 这样 ？





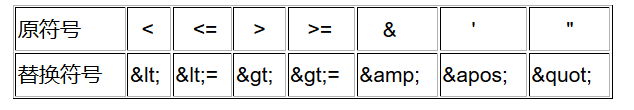
### 测试



### 问题（xml内部格式） xml 如何

在 mybatis 的 xml 里面 ，请谨慎 使用 大于， 小于 符号。特别是同时出现2个

字母 没有问题，但是 下面的 > < >= <= 的符号有问题



上面的例子 可以使用 between and 关键字， 但是 符号 有 问题

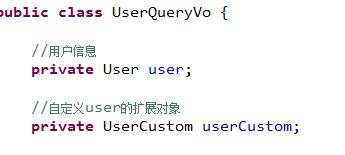
## parameterType传递pojo包装对象

可以定义pojo包装类型扩展mapper接口输入参数的内容。

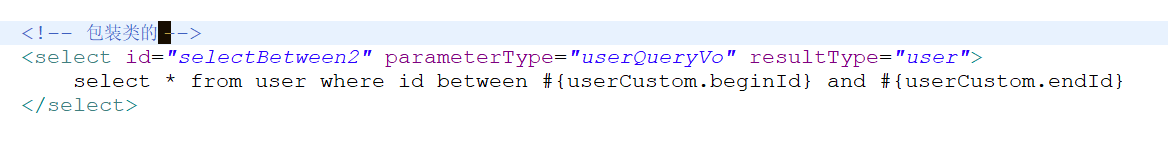
需求：

自定义查询条件查询用户信息，需要向statement输入查询条件，查询条件可以有user信息、商品信息。。。。

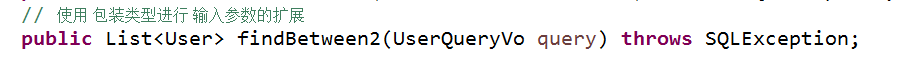
### 包装类型



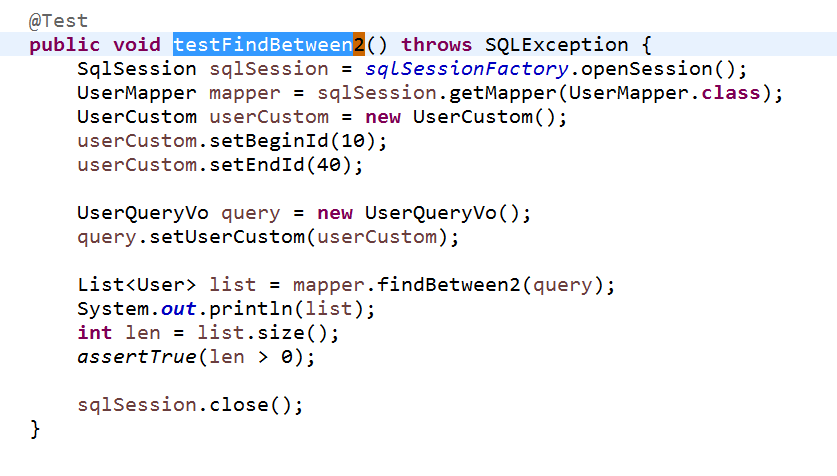
### mapper.xml



### mapper.java



### 测试



### 异常

如果parameterType中指定属性错误，异常，找不到getter方法：

org.apache.ibatis.exceptions.PersistenceException:

### Error querying database. Cause: org.apache.ibatis.reflection.ReflectionException: There is no getter for property named 'endid' in 'class com.hc.mybatis.po.UserCustom'

### Cause: org.apache.ibatis.reflection.ReflectionException: There is no getter for property named 'endid' in 'class com.hc.mybatis.po.UserCustom'

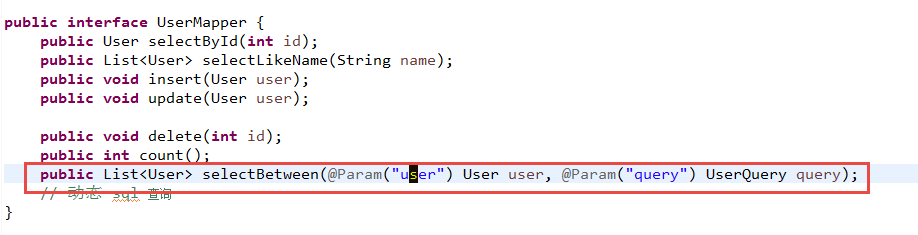
at org.apache.ibatis.exceptions.ExceptionFactory.wrapException(ExceptionFactory.java:30)

注意：如果将来和spring整合后，不是通过调用getter方法来获取属性值，通过反射强读取pojo的属性值。

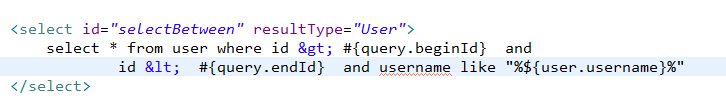
## 使用 @param 参数引入多个输入参数

使用：

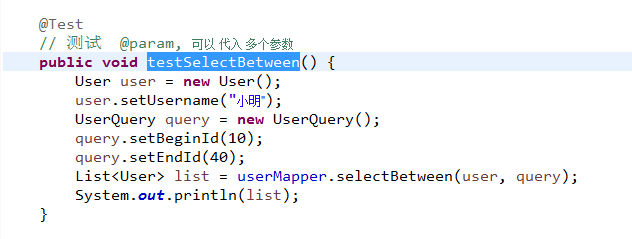
Mapper.java 的定义



Mapper.xml 的定义



测试：

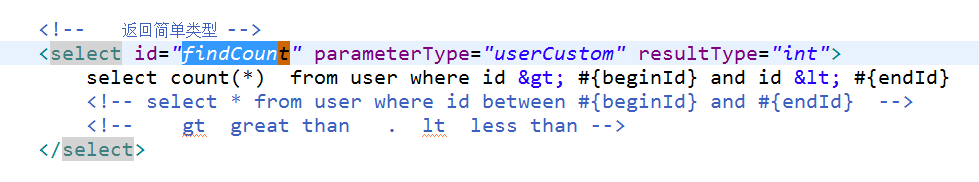


## resultType

指定输出结果的类型（pojo、简单类型、hashmap..），将sql查询结果映射为java对象 。

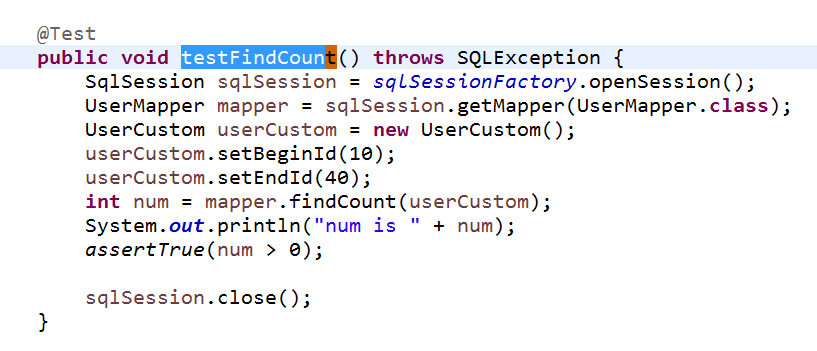
### 返回简单类型

mapper.xml



mapper.java





注意：

如果查询记录结果集为一条记录且一列再使用返回简单类型。

## resultMap(入门)（重点）

resultType ：指定输出结果的类型（pojo、简单类型、hashmap..），将sql查询结果映射为java对象 。

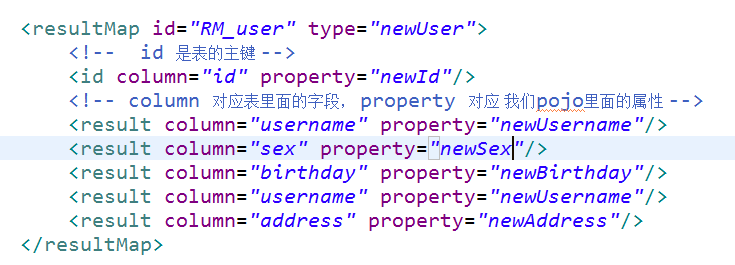
使用resultType注意：sql查询的列名要和resultType指定pojo的属性名相同，指定相同 属性方可映射成功，如果sql查询的列名要和resultType指定pojo的属性名全部不相同，list中无法创建pojo对象的。

resultMap：将sql查询结果映射为java对象。

如果sql查询列名和最终要映射的pojo的属性名不一致，使用resultMap将列名和pojo的属性名做一个对应关系 （列名和属性名映射配置）

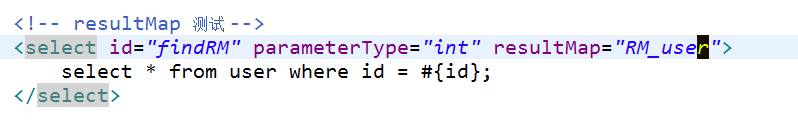
### resultMap配置

userRM 是 resultMap 的 id， 就是 它的标识 。



### 使用resultMap

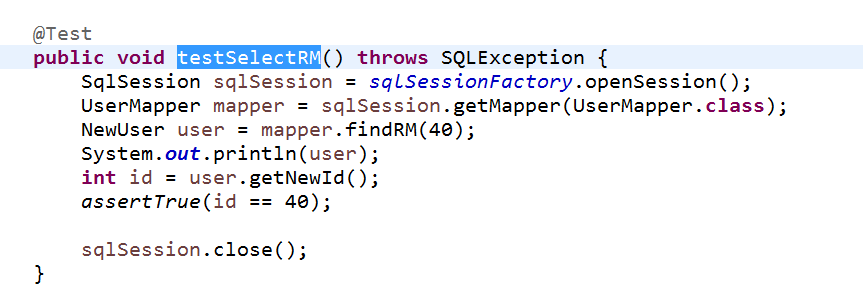
如果 resultMap 和 使用的地方是 同一个 xml， 就直接使用 resultMap 的 id，否则就必须是namesapce + id， 像 com.hc.mybatis.mapper.userRM



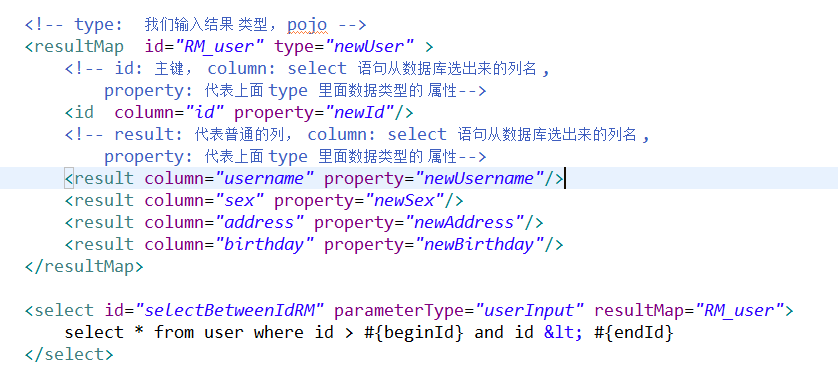
### mapper.java



测试代码：



### 新例子



# 动态sql(重点)

mybatis重点是对sql的灵活解析和处理。

## 常用标签

# if，

# choose (when, otherwise)，

# trim (where, set)，

# Foreach

## Choose

<select id="queryBy"

resultType="Blog">

SELECT \* FROM BLOG WHERE 1=1

<choose>

<when test="title != null">

AND title like #{title}

</when>

<otherwise>

AND id= 1

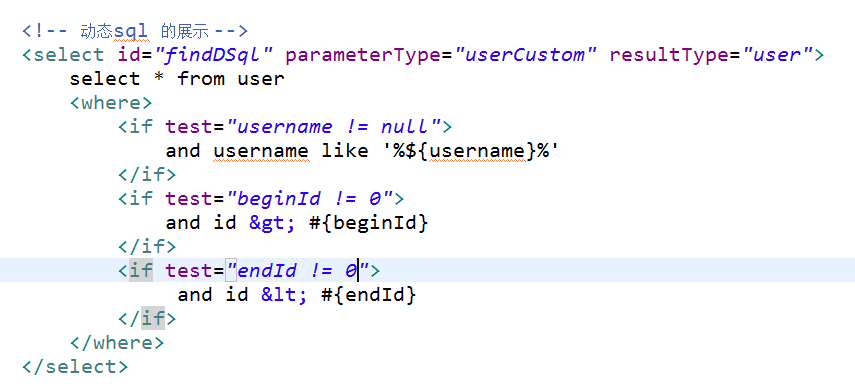
</otherwise>

</choose></select>

## 需求

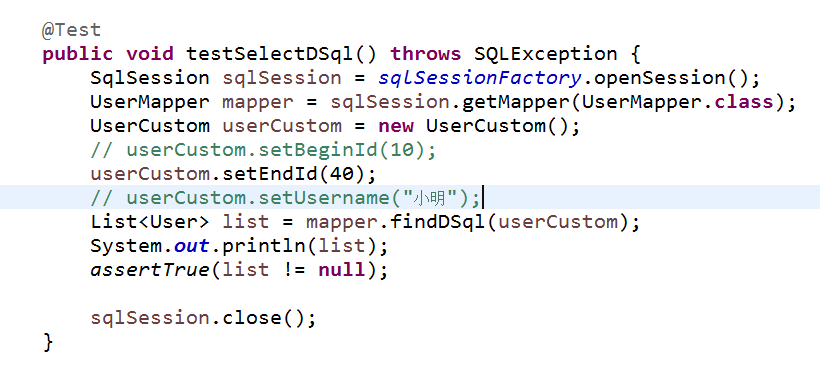
将自定义查询条件查询用户列表和查询用户列表总记录数改为动态sql。例如 查询 id 在 10 到 40之间的 小明 用户 。

## if和where





## 测试代码



## 错误

### SQL: select \* from user where and id > ? and id < ?

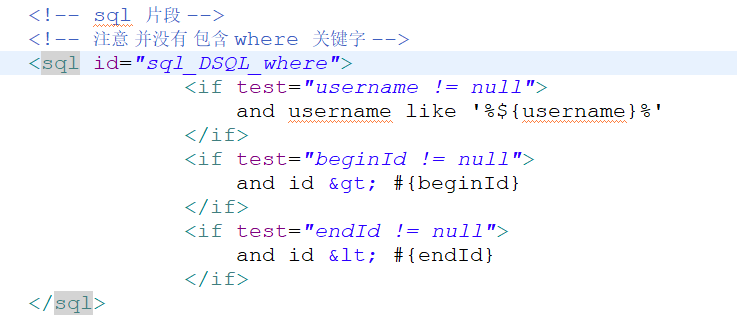
## sql片段

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。

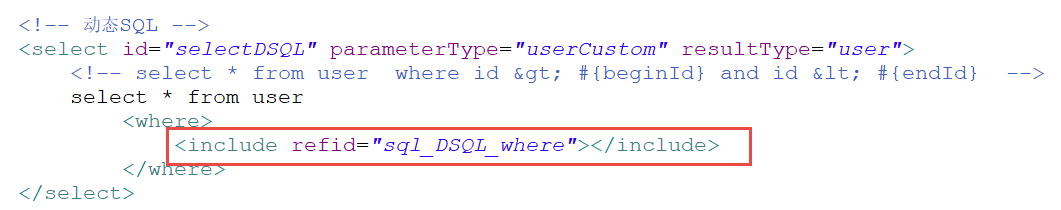
通用的sql语句，常用：where条件、查询列

### sql片段的定义

关键词： sql



### 引用sql片段



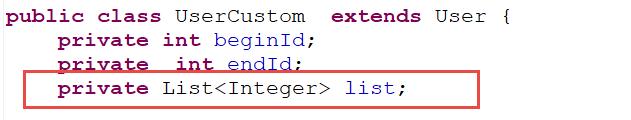
## foreach

在statement通过foreach遍历parameterType中的集合类型。

需求：

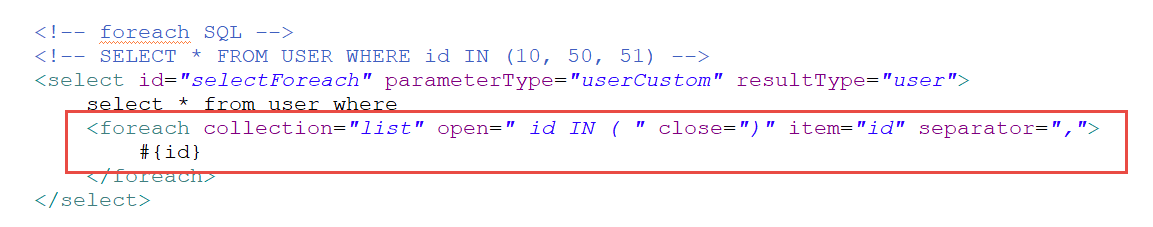
根据多个用户id查询用户信息。

### 在userCustom中定义list<Integer> ids

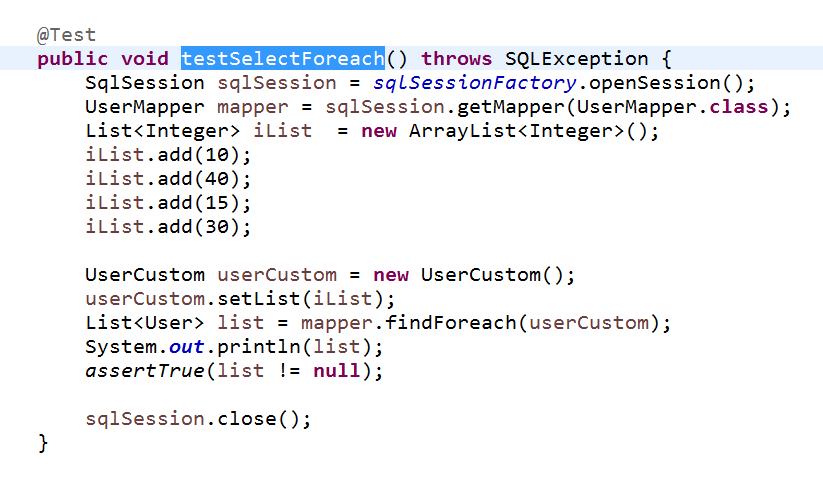


### 修改where语句

使用foreach遍历list：



### 测试代码



# 第一部分总结

mybatis 是 什么 ？ mybatis 是一个 优秀的持久层 架构 。 它是一个 不完全的 ORM。

。

特点： 在于 sql 语句的灵活使用。 入门比较容易 。对项目需求变化的适应性强

mybatis 。

SqlMapConfig.xml 最主要的一个 配置 。

sqlSesionFactory

sqlSession

利用 sqlSesson 可以操作数据库

excutor ，mappedStatement 。

mybatis 实现方式：

原始的 DAO， 写 interface, user.xml DaoImpl.java

新的 DAO， mapper。 写 interface 和 usermapper.xml . 就完成了 dao 层的编码

sqlMapConfig.xml 文件 配置 。

poperties

typealias 。 package

mapper

package 扫描 com.hc.mybatis.mapper . 自动引入 userMapper.xml

mybatis 注解开发 。（了解）

输入和输出映射 。

parameterType .

返回简单类型

使用resultMap 。

动态sql 。

if where

sql

foreach