mybatis基础知识

mybatis第一天：基础知识

mybatis是一个java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。

1、从一个jdbc程序开始，通过jdbc 程序找到使用原生态的jdbc开发程序，存在哪些问题？？通过学习mybatis，mybatis是如何解决这些问题。

2、mybatis的架构（重点）

3、mybatis的入门程序（重点）

实现 用户的查询、添加、修改、删除。

4、mybatis开发dao的两种方法。（重点）

原始的dao开发方式（dao接口和dao实现都需要编写）

mapper代理方式（只需要写dao接口）

5、输入映射类型和输出映射类型。

6、动态sql

第二天：高级知识

高级映射查询（一对一、一对多、多对多）（重点）

查询缓存

延迟加载

mybatis和spring整合（重点）

mybatis逆向工程 。

# 开发环境

jdk1.7.0\_72

eclipse：eclipse-3.7-indigo

mysql：mysql5.1

## 创建数据库



先导入sql\_table.sql，再导入sql\_data.sql(记录系统的初始化数据)

通常需要提供初始化数据的数据库脚本。



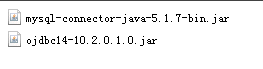
# jdbc编程中问题

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

参考教案

需要数据库的驱动包：



上边是mysql的驱动，下边是oracle的驱动。

## jdbc问题总结

1、数据库连接频繁的创建和关闭，缺点浪费数据库的资源，影响操作效率

设想：使用数据库连接池

2、sql语句是硬编码，如果需求变更需要修改sql，就需要修改java代码，需要重新编译，系统不易维护。

设想：将sql语句 统一配置在文件中，修改sql不需要修改java代码。

3、通过preparedStatement向占位符设置参数，存在硬编码（ 参数位置，参数）问题。系统不易维护。

设想：将sql中的占位符及对应的参数类型配置在配置文件中，能够自动输入 映射。

4、遍历查询结果集存在硬编码（列名）。

设想：自动进行sql查询结果向java对象的映射（输出映射）。

# mybatis架构(重点)

## mybatis介绍

MyBatis 本是[apache](http://baike.baidu.com/view/28283.htm)的一个开源项目[iBatis](http://baike.baidu.com/view/628102.htm), 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis，实质上Mybatis对ibatis进行一些改进。 目前mybatis在github上托管。git（分布式版本控制，当前比较流程）

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。

## mybatis架构

SqlMapConfig.xml（mybatis全局配置文件，名称不固定）

运行环境（数据源、事务）

mapper.xml(配置sql语句), mapper.xml.....

SqlSessionFactory（会话工厂）

作用：创建SqlSession

SqlSession（向面用户的接口，数据库操作方法）

作用：操作数据库

Executor（数据库操作的执行器）

Executor是一个接口，有两个实现 （默认执行器和缓存执行器）

mysql

MappedStatement（mybatis的封装对象）

封装：sql语句(占位符)

输入 映射

java简单类型

pojo类型

hashmap

输出 映射

java简单类型

pojo类型

hashmap

# mybatis入门程序

## 需求

实现用户查询：

根据用户id（主键）查询用户信息（单条记录）

根据用户名称模糊查询用户信息（多条记录）

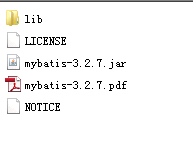
用户添加

用户删除

用户修改

## 导入jar包

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



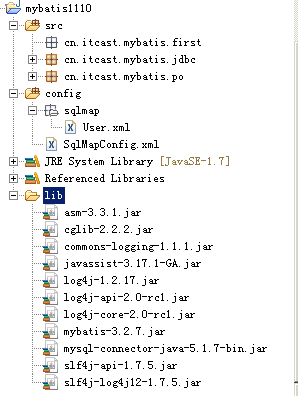
mybatis-3.2.7.pdf---操作手册

mybatis-3.2.7.jar--核心 jar包

依赖的jar包



## 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

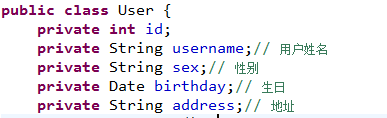
## SqlMapConfig.xml(公用文件)

通过SqlMapConfig.xml加载mybatis运行环境。



## 根据id查询用户

### pojo（User.java）



### User.xml（重点）

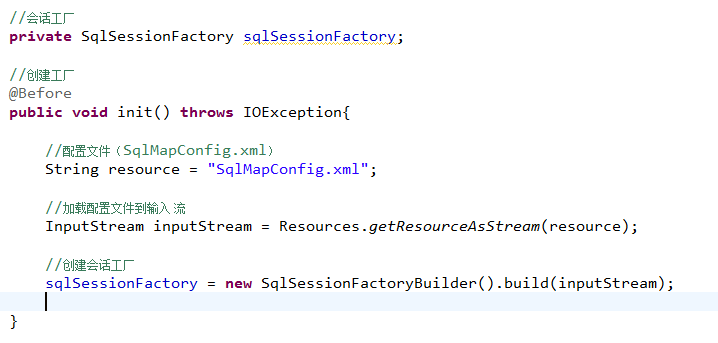
建议命名规则：表名+mapper.xml

早期ibatis命名规则：表名.xml



### 编码

创建SqlSessionFactory:

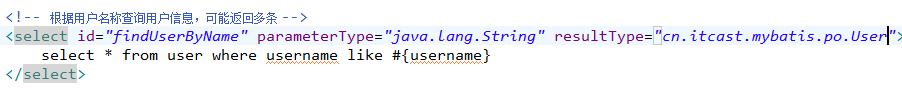




## 根据用户名称模糊查询用户信息

根据用户名称模糊查询用户信息可能返回多条记录。

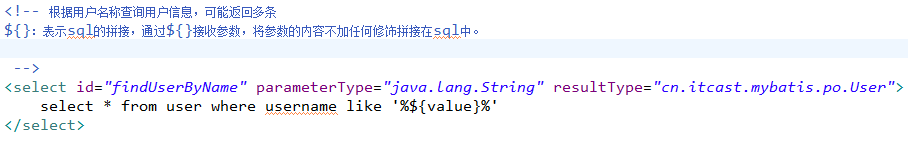
### User.xml



### 编码



### 使用${}接收参数



## mybatis开发过程小结

1、编写SqlMapConfig.xml

2、编写mapper.xml

定义了statement

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

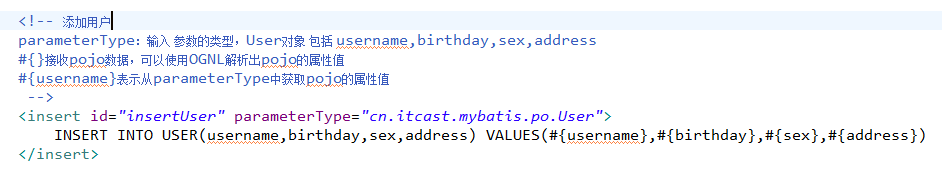
如果执行添加、更新、删除需要调用SqlSession.commit()

6、SqlSesion使用完成要关闭

## 用户添加

向用户表插入一条记录。

### User.xml



### 编码



### 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：

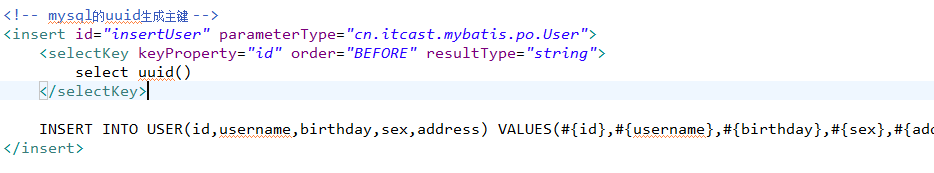


使用mysql的uuid机制生成主键：

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。

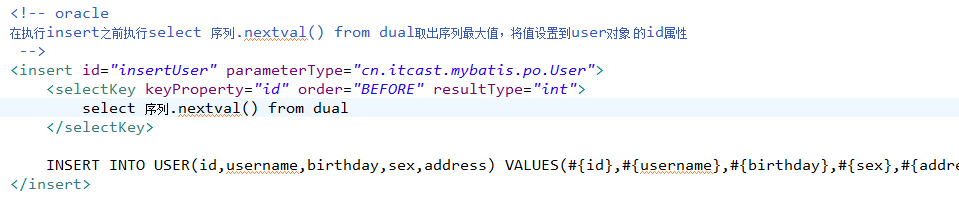


实现 oracle数据库主键返回，如何做？？

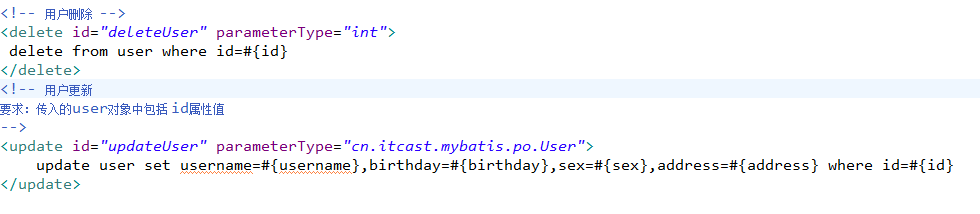
oracle没有自增主键机制，使用序列完成主键生成。

实现思路：

先查询序列得到主键，将主键设置到user对象中，将user对象插入数据库。



## 用户删除和更新



### 编码

// 测试根据id删除用户(得到单条记录)

@Test

**public** **void** testDeleteUser() {

// 通过sqlSessionFactory创建sqlSession

SqlSession sqlSession = sqlSessionFactory.openSession();

// 通过sqlSession操作数据库

**try** {

sqlSession.delete("test.deleteUser", 35);

// 需要提交事务

sqlSession.commit();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

// 关闭sqlSession

sqlSession.close();

}

}

// 测试根据id更新用户(得到单条记录)

@Test

**public** **void** testUpdateUser() {

// 通过sqlSessionFactory创建sqlSession

SqlSession sqlSession = sqlSessionFactory.openSession();

// 通过sqlSession操作数据库

// 创建更新数据对象，要求必须包括 id

User user = **new** User();

user.setId(35);

user.setUsername("燕青");

user.setAddress("河南郑州");

// user.setBirthday(new Date());

user.setSex("1");

**try** {

sqlSession.update("test.updateUser", user);

// 需要提交事务

sqlSession.commit();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

// 关闭sqlSession

sqlSession.close();

}

System.*out*.println("用户的id=" + user.getId());

}

## Mybatis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。

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 ORM框架。

hibernate适合需求固定，对象数据模型稳定，中小型项目，比如：企业OA系统。

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

mapper.xml是以statement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，多对多映射）

### #{}

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

### ${}

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

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

SqlSessionFactoryBuilder是以工具类方式来使用，需要创建sqlSessionFactory就new一个SqlSessionFactoryBuilder。

### sqlSessionFactory

正常开发时，以单例方式管理sqlSessionFactory，整个系统运行过程中sqlSessionFactory只有一个实例，将来和spring整合后由spring以单例方式管理sqlSessionFactory。

### SqlSession

sqlSession是一个面向用户（程序员）的接口，程序员调用sqlSession的接口方法进行操作数据库。

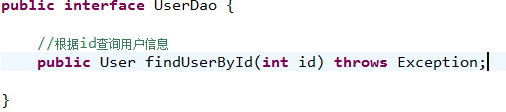
sqlSession能否以单例 方式使用？？

由于sqlSession是线程不安全，所以sqlSession最佳应用范围在方法体内，在方法体内定义局部变量使用sqlSession。

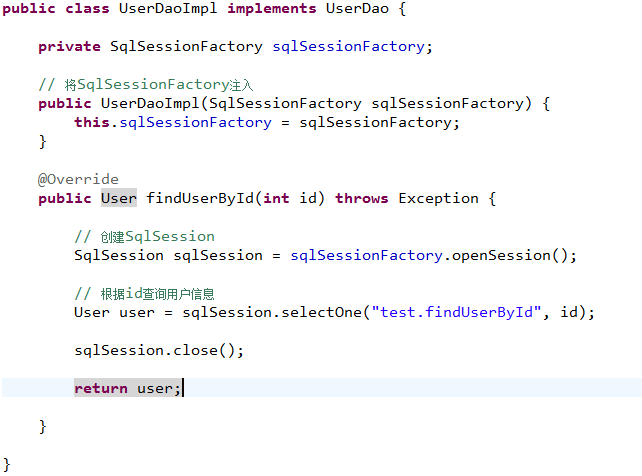
## 原始dao开发方式

程序员需要写dao接口和dao 的实现 类

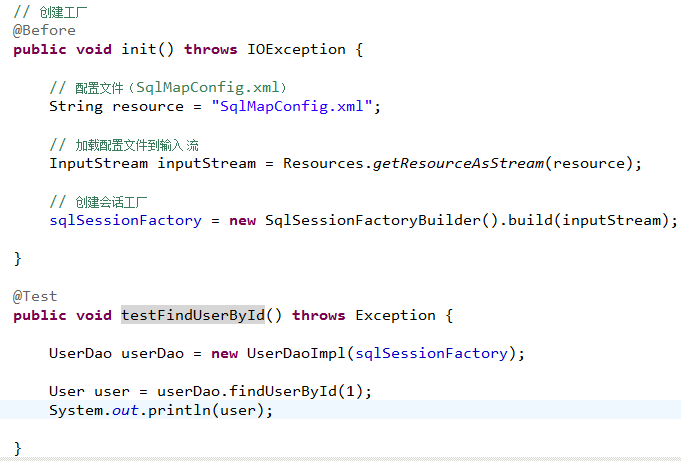
### dao接口



### 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

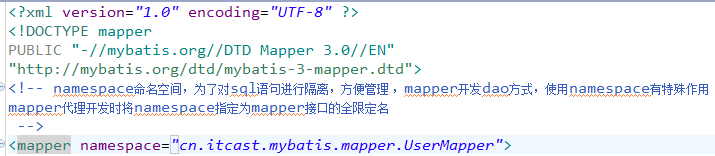
1 dao的实现类中存在重复代码，整个mybatis操作的过程代码模板重复（先创建sqlsession、调用sqlsession的方法、关闭sqlsession）

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接口的全限定名



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

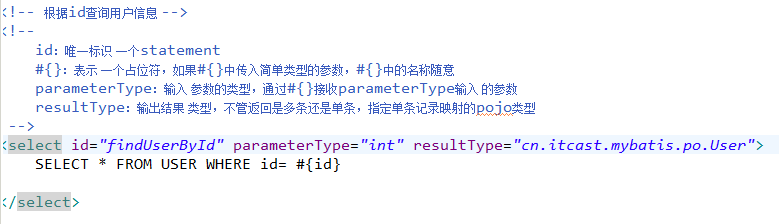
3、mapper.xml中statement的parameterType和mapper.java中方法输入参数类型一致

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

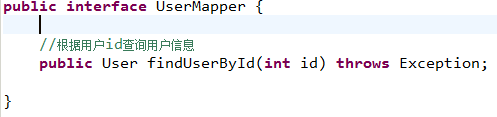
namespace指定为mapper接口的全限定名



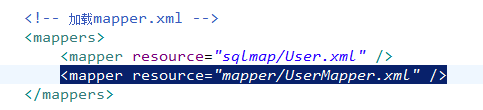
### mapper接口

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



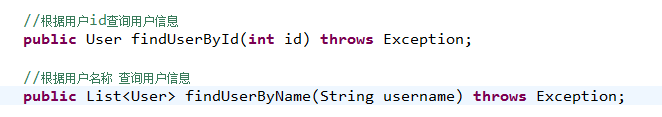
### 将mapper.xml在SqlMapConfig.xml中加载



### 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

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

可以把一些通用的属性值配置在属性文件中，加载到mybatis运行环境内。

比如：创建db.properties配置数据库连接参数。



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

mybatis运行时可以调整一些全局参数（相当于软件的运行参数），参考：mybatis-settings.xlsx

根据使用需求进行参数配置。

注意：小心配置，配置参数会影响mybatis的执行。

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)

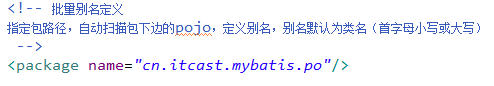
可以将parameterType、resultType中指定的类型 通过别名引用。

### 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 |

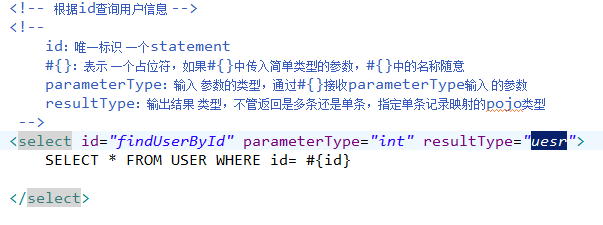
### 自定义别名





### 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



# 输入和输出映射

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

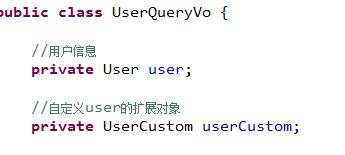
## parameterType传递pojo包装对象

可以定义pojo包装类型扩展mapper接口输入参数的内容。

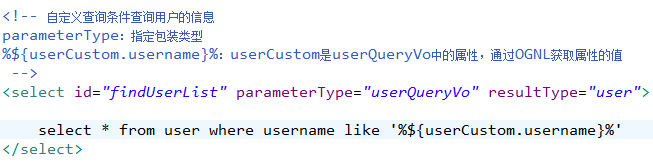
需求：

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



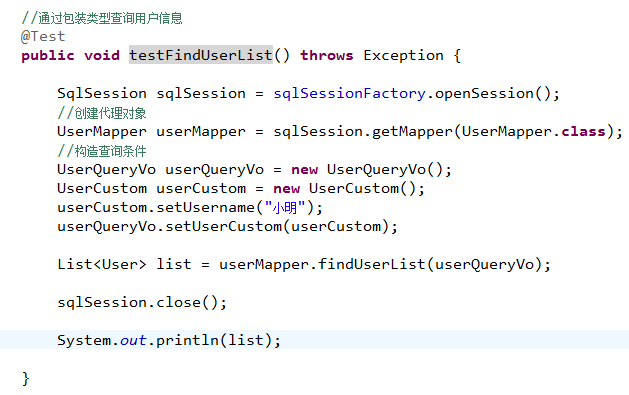
### mapper.xml



### 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 'userCusto' in 'class cn.itcast.mybatis.po.UserQueryVo'

### Cause: org.apache.ibatis.reflection.ReflectionException: There is no getter for property named 'userCusto' in 'class cn.itcast.mybatis.po.UserQueryVo'

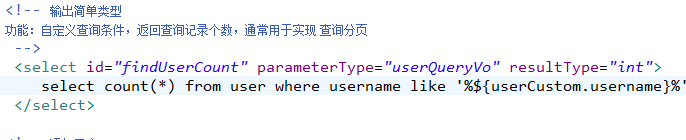
注意：如果将来和spring整合后，不是通过调用getter方法来获取属性值，通过反射强读取pojo的属性值。

## 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 。

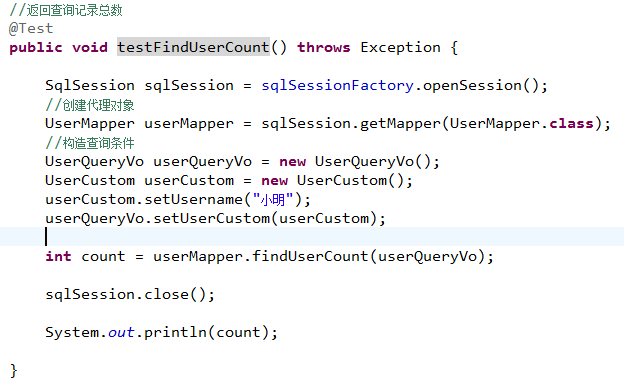
### 返回简单类型

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)

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。

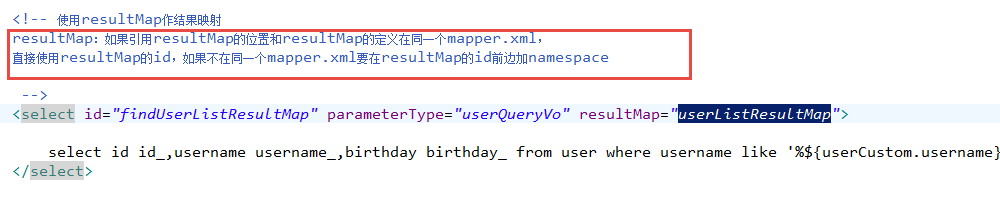
resultMap：将sql查询结果映射为java对象。

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



### 使用resultMap



### mapper.java



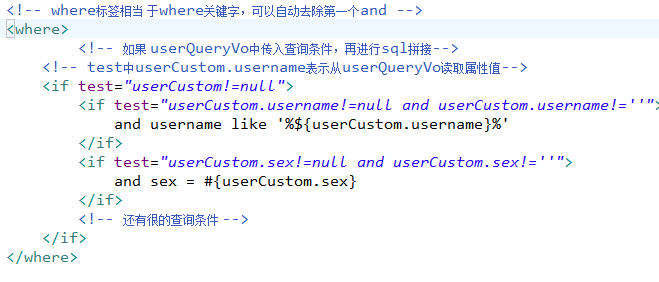
# 动态sql(重点)

mybatis重点是对sql的灵活解析和处理。

## 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

## if和where



## sql片段

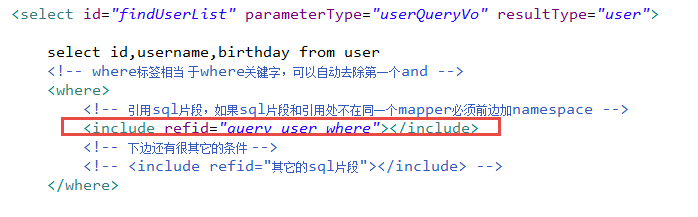
通过sql片段可以将通用的sql语句抽取出来，单独定义，在其它的statement中可以引用sql片段。

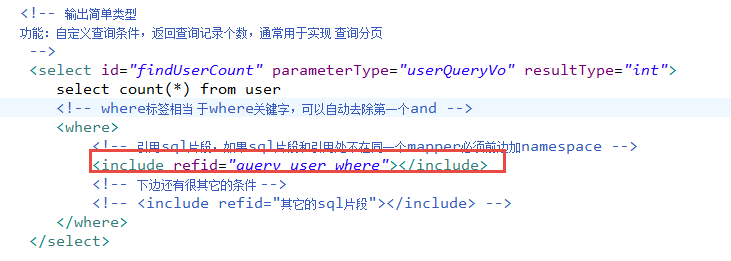
通用的sql语句，常用：where条件、查询列

### sql片段的定义



### 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。

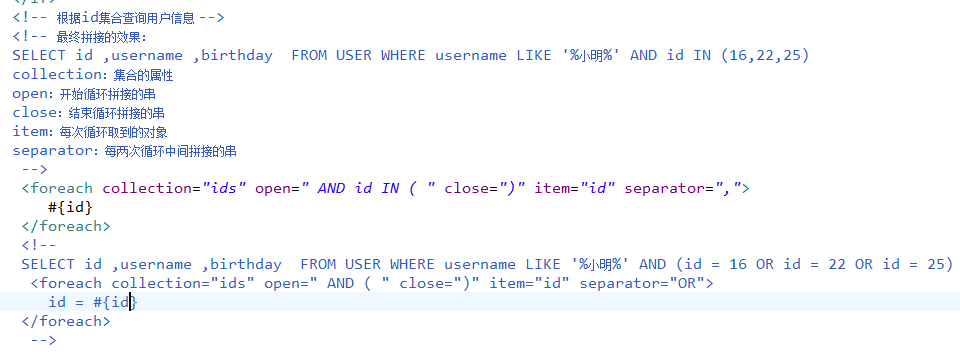
### 在userQueryVo中定义list<Integer> ids

在userQueryvo中定义list<Integer> ids存储多个id



### 修改where语句

使用foreach遍历list：



### 测试代码

