Mybatis 第一天课堂笔记

Mybatis课程安排：

第一天：

Jdbc编程中的问题？

Mybatis的框架原理（重点）

Mybatis的第一个程序：

实现用户信息的增、删、改、查。

Mybatis如何解决jdbc编程的问题。

与hibernate不同。

mybatis开发dao两种方法：（重点）

原始的dao开发方法（接口和接口实现）

Mapper动态代码方法

Mybatis的配置文件：

SqlMapConfig.xml（全局配置文件）

XXXXXMapper.xml（映射文件，重点）

Mybatis动态Sql（重点）

resultType（重点掌握）和resultMap区别。

第二天：

Mybatis关联查询：

一对一查询

一对多

多对多

Mybatis的缓存

一级缓存

二级缓存

延迟加载

Mybatis和springmvc整合（重点）

原始的dao开发方法（接口和接口实现）

Mapper动态代码方法

Mybatis逆向工程（由数据库表生成类文件和映射文件）

# Jdbc编程中的问题

## 写一个程序

目标：

1. 通过jdbc程序找出当中的问题
2. 重视jdbc编程，企业在技术选型时对于小型的项目或性能要求较高的项目，优先选用jdbc开发。

准备开发环境：

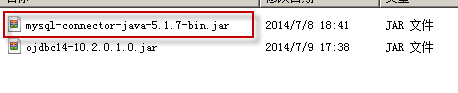
Jdk1.7.72

Eclipse indigo（3.7）

Mysql5.1

运行环境：

Mysql5.1数据库驱动



创建mysql数据库：



代码如下：

**public** **class** JdbcTest {

//定义sql语句，根据名称查询用户表的记录

**private** **static** String *sql* = "SELECT \* FROM USER WHERE username = ?";

**public** **static** **void** main(String[] args) **throws** SQLException {

//数据库连接

Connection connection = **null**;

//预编译statement

//好处：防止 sql注入，提高数据的性能

PreparedStatement preparedStatement = **null**;

//结果集

ResultSet resultSet = **null**;

**try** {

//加载数据库驱动

Class.*forName*("com.mysql.jdbc.Driver");

//连接数据库

connection = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/mybatis", "root", "mysql");

//构造preparedStatement

preparedStatement = connection.prepareStatement(*sql*);

//向preparedStatement中占位符的位置设置参数

preparedStatement.setString(1, "张三");

//发起数据库查询请求，得到结果集

resultSet = preparedStatement.executeQuery();

//遍历查询结果

**while**(resultSet.next()){

**int** id = resultSet.getInt("id");

String username = resultSet.getString("username");

Date birthday = resultSet.getDate("birthday");

System.*out*.println(id+" "+username+" "+birthday);

}

} **catch** (Exception e) {

e.printStackTrace();

}**finally**{

//释放资源

**if**(resultSet!=**null**){

resultSet.close();

}

**if**(preparedStatement!=**null**){

preparedStatement.close();

}

**if**(connection!=**null**){

connection.close();

}

}

}

}

## 总结问题

1. 将sql语句硬编码到java代码中，如果修改sql语句，需要修改java代码，重新编译。系统可维护性不高。

设想如何解决？

能否将sql单独 配置在配置文件中。

1. 数据库连接频繁开启和释放，对数据库的资源是一种浪费。

设想如何解决？

使用数据库连接池管理数据库连接。

1. 向preparedStatement中占位符的位置设置参数时，存在硬编码（占位符的位置，设置的变量值）

设想如何解决？

能否也通过配置的方式，配置设置的参数，自动进行设置参数。

1. 解析结果集时存在硬编码（表的字段名、字段的类型）

设想如何解决？

能否将查询结果集映射成java对象。

# Mybatis框架

## Mybatis是什么？

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。

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statement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框架图

Mybatis的配置

SqlMapConfig.xml（全局配置文件，数据源，事务，mapper.xml）

Mapper.xml mapper.xml mapper.xml ….

SqlSessionFactory会话工厂

作用：生成 SqlSession

SqlSession(面向用户的接口，操作数据库（增、删、改、查）)

Executor（数据库操作执行器是框架底层封装对象，两个实现：基本执行器、缓存执行器）

SqlSession内容使用Executor操作数据库

Mapped Statement

框架底层封装对象（sql语句、输入参数、输出结果类型）

Sql的输入映射参数

Java简单类型

Hashmap

自定义pojo

Sql输出映射

Java简单类型

Hashmap

自定义pojo

Database数据库

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。

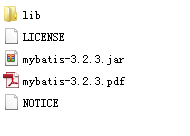
1. 通过mybatis环境等配置信息构造SqlSessionFactory即会话工厂
2. 由会话工厂创建sqlSession即会话，操作数据库需要通过sqlSession进行。
3. mybatis底层自定义了Executor执行器接口操作数据库，Executor接口有两个实现，一个是基本执行器、一个是缓存执行器。
4. Mapped Statement也是mybatis一个底层封装对象，它包装了mybatis配置信息及sql映射信息等。mapper.xml文件中一个sql对应一个Mapped Statement对象，sql的id即是Mapped statement的id。
5. Mapped 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。

# mybatis第一程序

## 准备环境：

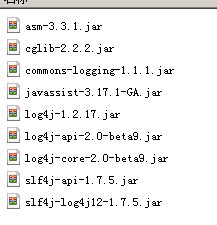
* Mybatis运行环境：

去mybatis官方网站下载Mybatis3.2.3。



mybatis-3.2.3.jar：mybatis核心包

mybaits的依赖包：



* 准备开发环境：

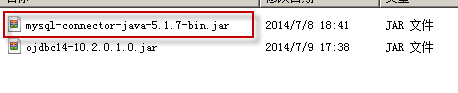
Jdk1.7.72

Eclipse indigo（3.7）

Mysql5.1

* 数据库运行环境：

Mysql5.1数据库驱动



* 创建mysql数据库：



## 用户查询开发

需求：

根据用户id查询一条记录（返回单条记录）

查询用户列表（返回list集合）

### 创建java工程



### 加入mybatis的jar包

加入 ：mybatis核心包、mybatis依赖包、数据驱动包



### 配置log4j.properties

# Global logging configuration

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

**注意：开发阶段使用DEBUG**

### 配置mybatis全局配置文件

全局配置文件：SqlMapConfig.xml（名称不是固定的）



配置的内容包括：

数据库运行环境（和spring整合废除了）

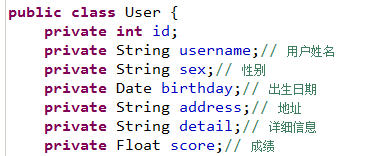
Mapper映射文件



### 创建po类



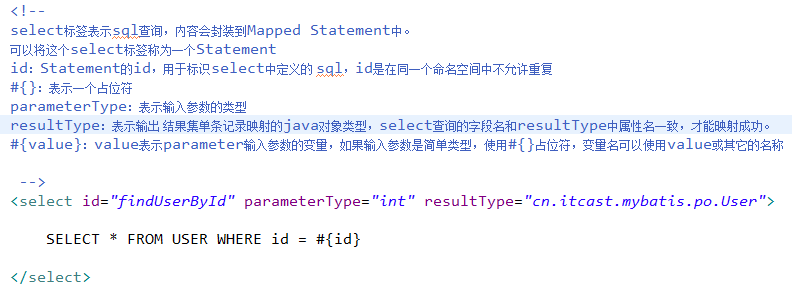
建议po类名和表名一致，属性名和表字段名一致。

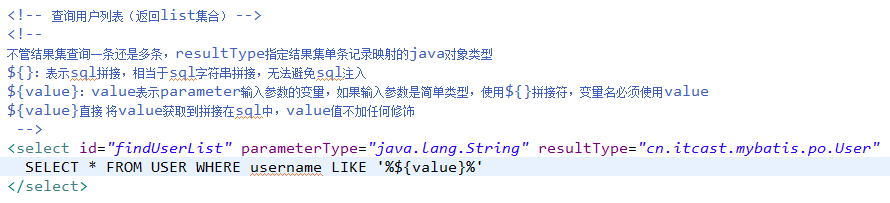


### 配置mybatis的mapper映射文件



User.xml（mybatis推荐的命名方式是XXXXMapper.xml）

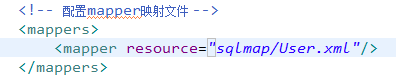




#{}：用于占位符

${}：用于sql拼接

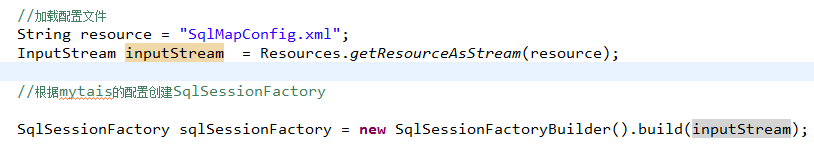
在全局配置文件SqlMapConfig.xml中配置User.xml。



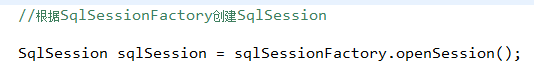
### 编写用户查询程序



第一步：根据mytais的配置创建SqlSessionFactory

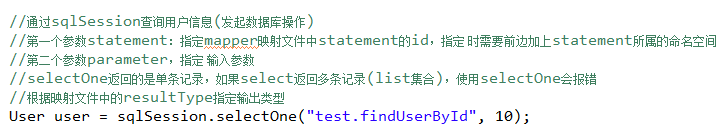


第二步：根据SqlSessionFactory创建SqlSession

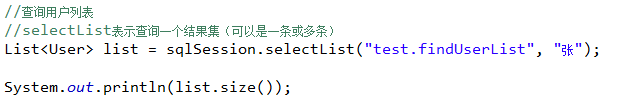


第三步：通过SqlSession操作数据库（根据用户id查询一条记录）

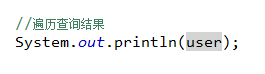
1. selectOne，查询单条记录



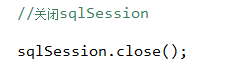
1. selectList，查询集合



第四步：遍历查询结果



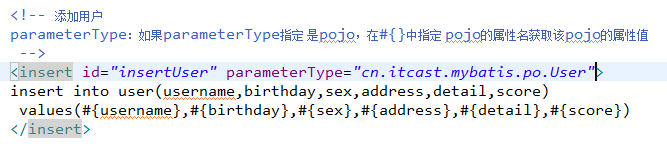
第五步：SqlSession关闭



## 用户添加

### 编写mapper.xml映射文件

在User.xml中定义添加用户的statement。



### 编写用户添加程序



### 主键返回问题

需求 ：对于新增的记录，需要将主键返回到pojo中，就可以从pojo中获取新添加的记录id。

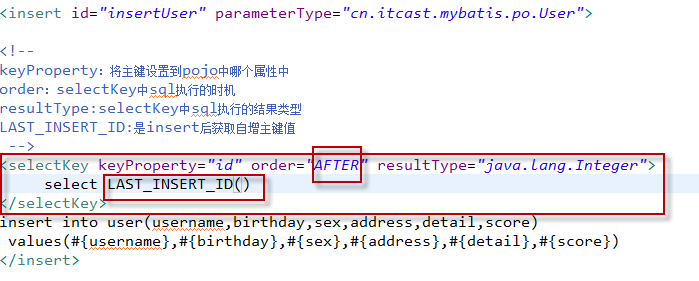
自增主键生成 Uuid主键生成时机区别：

自增主键在insert语句执行后生成 的。

Uuid主键在insert语句执行前生成 的。

#### Mysql自增主键获取：

在insertUser的statement中添加selectKey



#### Uuid主键获取

Uuid主键在insert语句执行前生成 的

如果使用uuid获取主键，定义selectkey

<insert id=*"insertUser"* parameterType=*"cn.itcast.mybatis.po.User"*>

<!--

keyProperty：将主键设置到pojo中哪个属性中

order：selectKey中sql执行的时机

resultType:selectKey中sql执行的结果类型

LAST\_INSERT\_ID:是insert后获取自增主键值

-->

<selectKey keyProperty=*"id"* order="*BEFORE*" resultType=*"java.lang.String"*>

select uuid()

</selectKey>

insert into user(id,username,birthday,sex,address,detail,score)

values(#{id},#{username},#{birthday},#{sex},#{address},#{detail},#{score})

</insert>

如果不用selectKey，也可以在调用SqlSession.insert()前，在输入参数设置id值 (生成uuid，设置到user的id属性中。)。

#### Oracle主键返回

Oracle没有自增主键，使用oracle的序列(可以生成流水号，类似 自增主键)生成主键。

通过序列获取流水号方法：

Select 序列名.next.val from dual

<insert id=*"insertUser"* parameterType=*"cn.itcast.mybatis.po.User"*>

<!--

keyProperty：将主键设置到pojo中哪个属性中

order：selectKey中sql执行的时机

resultType:selectKey中sql执行的结果类型

LAST\_INSERT\_ID:是insert后获取自增主键值

-->

<selectKey keyProperty=*"id"* order="*BEFORE*" resultType=*"java.lang.String"*>

Select 序列名.next.val from dual

</selectKey>

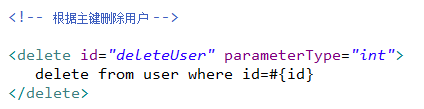
insert into user(id,username,birthday,sex,address,detail,score)

values(#{id},#{username},#{birthday},#{sex},#{address},#{detail},#{score})

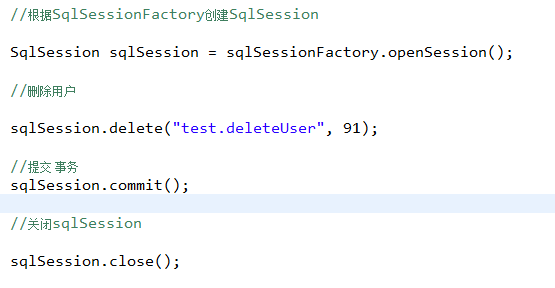
</insert>

## 用户删除

### Mapper.xml映射文件

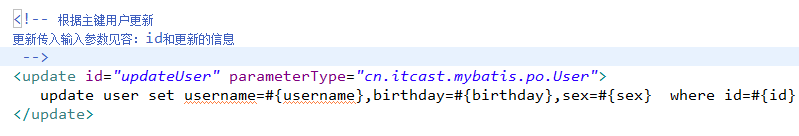


### 编写删除程序

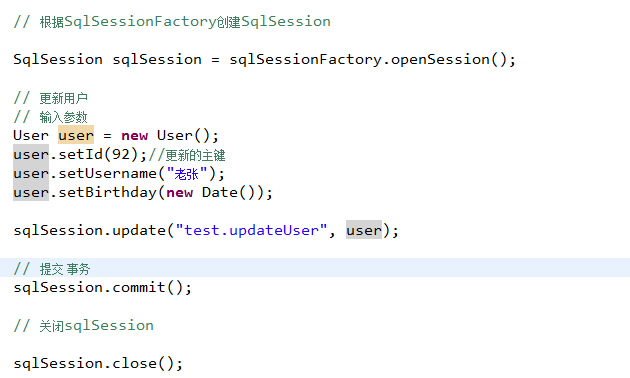


## 用户更新

### Mapper.xml映射文件



### 编写更新程序



## 小结

Mybatis开发：

第一步：编写SqlMapConfig.xml全局配置文件（重点配置mapper）

第二步：**编写mapper.xml映射文件(重点)**

定义statement（包括sql语句，输入和输出 参数类型）

在SqlMapConfig.xml全局配置文件配置mapper.xml(此配置可以通过规则省略)

第三步：**编写程序(重点)**

获取SqlSessionFactory

通过SqlSessionFactory获取SqlSession

通过SqlSession操作数据库（调用上边定义的statement）

得到statement的输出映射java对象

# 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。

# 与hibernate不同

Orm：mybatis不是一个完全的orm框架，hibernate是一个完全的orm框架。Mybatis需要程序员自己写sql，但是也存在映射（输入参数映射，输出结果映射）

学习门槛：mybatis比hibernate低。

应用的领域：

对于中小型项目，且需求固定，关系模型相对固定建议使用hibernate。适用传统项目，比如ERP。

对于需求不固定的项目，比如：互联网项目，建议使用mybatis，可以灵活去编写sql语句。

# SqlSession使用方法

SqlSessionFactoryBuilder：用于创建SqlSessionFactory，将SqlSessionFactoryBuilder当成工具类使用。

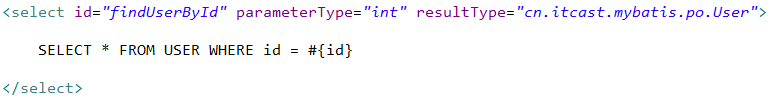
SqlSessionFactory：会话工厂，用于创建SqlSession，SqlSessionFactory一旦创建成功，不用每次创建工厂，建议单例模式使用工厂。如果和spring整合后，由spring来管理SqlSessionFactory，**在spring容器中SqlSessionFactory是一个单例对象**。

SqlSession：(重点)是一个面向用户的接口，通过SqlSessionFactory获取SqlSession，每次数据操作都需要创建新的SqlSession，SqlSession 不是线程安全，最佳应用场合是方法体内，在方法中定义一个SqlSession局部变量。

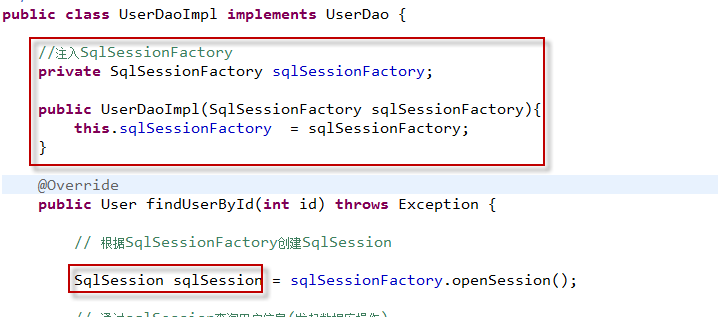
# 使用mybatis开发原始dao

程序员需要开发dao接口和dao的实现类。

编写User.xml映射文件：



Dao接口和实现类：



总结问题：

在每一个dao方法中在调用 sqlSession的接口方法时需要对statement的id进行硬编码。

在调用 sqlSession的接口方法时，向statement传入参数，由于sqlSession接口方法参数是object，如果类型设置错误，在编译阶段没有报错。

# Mapper动态代理方法

Mapper动态代理方法：程序员只需要写dao接口(Mapper)，而不需要写dao实现类，由mybatis根据dao接口和映射文件中statement的定义生成接口实现代理对象。可以调用代理对象方法。

Mybatis官方建议：将dao接口叫做mapper。

目标：通过一些规则让mybatis根据dao接口和映射文件中statement的定义生成接口实现代理对象

mybatis将以下代码自动在代理对象实现：

User user = sqlSession.selectOne("test.findUserById", id);

如何让mapper接口找到mapper.xml中statement的id。

## 第一步：开发mapper.xml映射文件



mapper.xml映射文件命名规则：XXXXMapper.xml

<!-- 根据用户id查询一条记录（返回单条记录） -->

<select id=*"findUserById"* parameterType=*"int"* resultType=*"cn.itcast.mybatis.po.User"*>

SELECT \* FROM USER WHERE id = #{id}

</select>

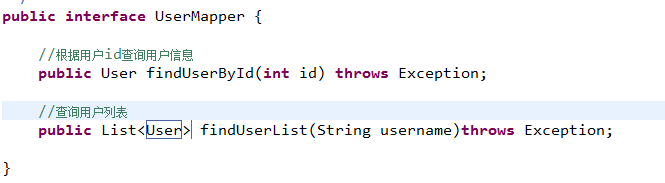
为了让mapper.xml和mapper.java对应起来，将mapper.xml中的namespace设置为mapper.java的全限定名。



## 第二步：开发mapper.java接口



mapper.java接口文件命名规则：XXXXMapper.java



Mybatis生成代理对象时，根据statement的标签决定调用 SqlSession的方法(select、insert、update..)

根据上边接口方法返回值 类型来决定 是调用 selectOne还是selectList，如果返回的是单个对象，动态代理调用selectOne()，如果返回的是集合对象，动态代理调用selectList()。

## Mapper开发规则

1. 在mapper.xml中将namespace设置为mapper.java的全限定名
2. 将mapper.java接口的方法名和mapper.xml中statement的id保持一致。
3. 将mapper.java接口的方法输入参数类型和mapper.xml中statement的parameterType保持一致
4. 将mapper.java接口的方法输出 结果类型和mapper.xml中statement的resultType保持一致

## 小结

在企业中使用原始dao开发仍然的是很多的，主要是由于前期使用ibatis。

建议使用mybatis动态代理方法，好处不用写实现类，开发快捷。使用动态代码方法需要遵循上边四点规范。

# SqlMapConfig.xml内容

SqlMapConfig.xml作为mybatis的全局配置文件，配置内容包括：数据库环境、mapper定义、全局参数设置。。。。

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）

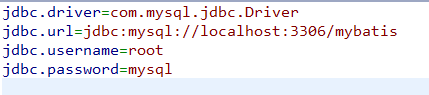
dataSource（数据源）

mappers（映射器）

## properties

将数据库连接参数单独在一个properties文件中配置，好处是：方便系统升级维护。

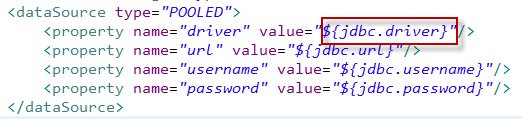
db.properties



此文件中的内容会通过SqlMapConfig.xml去加载 。



使用db.properties



**注意：由于mapper.xml中也可以引用属性文件中的内容，所以建议属性文件中key定义要有一定的规则，建议使用 前缀.XXX=XXXXX**

## settings（全局配置参数）

mybaits框架运行设置一些全局配置参数，比如：开启二级缓存 ，开启延迟载。。。

详细参考：

在ibatis中有一些设置性能参数（最大线程数、最大请求数。。），在mybatis中没有这些性能参数。

注意：设置全局参数会影响mybatis框架运行，谨慎设置。

## typeAliases（类型别名）

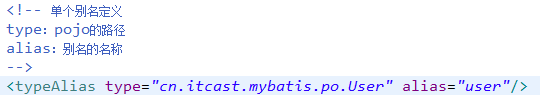
在parameterType和resultType设置时，为了方便编码，定义别名代替pojo的全路径。

* 框架默认支持别名

|  |  |
| --- | --- |
| 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 |

* 定义别名：

主要定义pojo的别名。





* 使用别名：



## typeHandlers（类型处理器）

类型处理器用于java类型和jdbc类型映射：

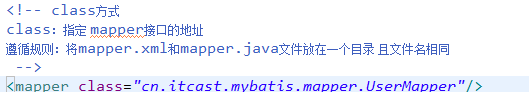
Mybatis提供 的类型处理器满足日常需要。

## mappers（映射器）重点掌握

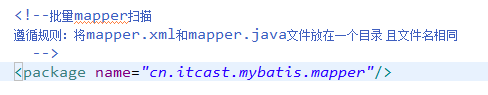
<mapper resource=" " />

使用相对于类路径的资源

如：<mapper resource="sqlmap/user.xml" />







**常用批量扫描方式。**

# mapper- parameterType(输入类型)

parameterType：用于设置输入参数的类型。

## #{}与${}

#{}：表示占位符，如果获取简单类型，#{}中可以使用value或其它名称 。有效防止sql注入。使用#{}设置参数无需考虑参数的类型。

如果使用#{}比较日期字段，select \* from tablename where birthday >=#{birthday}

${}：表示sql拼接，如果获取简单类型，#{}中只能使用value 。无法防止sql注入。使用${}设置参数必须考虑参数的类型，比如：使用oracle查询条件是日期类型，如果使用${}，必须人为将${}两边加单引号通过to\_date转日期。

Select \* from table where birthday >=to\_date(‘${birthday}’,’yyyy-MM-dd’)

在没有特殊要求的情况下，**建议使用#{}占位符**

有些情况必须使用${}，

比如：需要动态拼接表名,Select \* from ${tablename}

动态拼接排序字段：select \* from tablename order by ${username} desc

## 传递pojo对象

parameterType指定 输入参数为pojo自定义对象时，在sql中使用${}和#{}获取pojo的属性。

### 包装对象使用

开发中使用pojo传递查询条件 ，查询条件是综合的查询条件，不仅包括用户查询条件还包括其它的查询条件（是另一个pojo），使用包装对象传递输入参数。

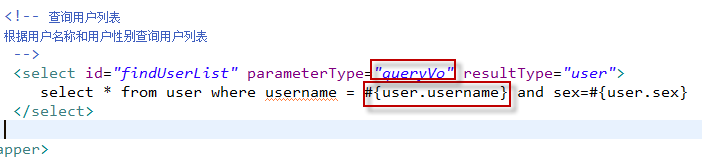
定义包装对象将查询条件(pojo)以类组合的方式包装起来。

parameterType使用包装对象：

包装对象：



mapper.xml



## 传递hashmap

parameterType指定 hashmap传递输入参数，#{}和${}中引用map的key。

Sql映射文件定义如下：

<!-- 传递hashmap综合查询用户信息 -->

<select id=*"selectUserByHashmap"* parameterType=*"hashmap"* resultType=*"user"*>

select \* from user where id=#{id} and username like '%${username}%'

</select>

## 小结：

如果使用动态代理方式，mapper接口的输入参数只有一个，是否有局限性？

Mapper就是dao，dao不是service，dao中没有业务逻辑，编写的dao具有公用的特性，被service公用 。

即使mapper接口方法输入参数只有一个，建议使用包装对象(注意不要使用map)，使用组合 方式将pojo包装进来。

对于parameterMap已经过期不建议使用。

# resultType

## 返回pojo

resultType：将sql查询结果集映射为java对象。要求sql查询的字段名和resultType指定pojo的属性名一致，才能映射成功。

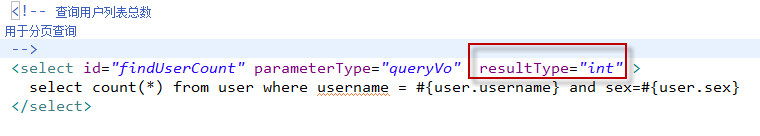
如果全部字段和pojo的属性名不一致，映射生成 的java对象为空，只要有一个字段和pojo属性名一致，映射生成 的java对象不为空。

结论：sql查询字段名和pojo的属性名一致才能映射成功。

不管select返回的是单个 对象还是集合对象，resultType要指定单条记录映射的java对象。

## 返回简单类型

如果 sql查询的结果集只有一行且一列，resultType可以返回简单类型。



## 返回hashmap

输出pojo对象可以改用hashmap输出类型，将输出的字段名称作为map的key，value为字段值。

* Mapper.xml

<select id=*"findUserListReturnMap"* parameterType=*"queryVo"* resultType=*"hashmap"*>

select id,username username\_ from user where username = #{user.username} and sex=#{user.sex}

</select>

* Mapper.java

*Public Map findUserListReturnMap(QueryVo queryVo);*

建议不使用map作为返回值 ，因为需要对key在代码中硬编码。

## 小结

1. 如果返回pojo，要求sql查询字段名和pojo的属性名一致。
2. 不管select返回的是单个 对象还是集合对象，resultType指定相同的映射类型（单条记录映射的java对象类型。）

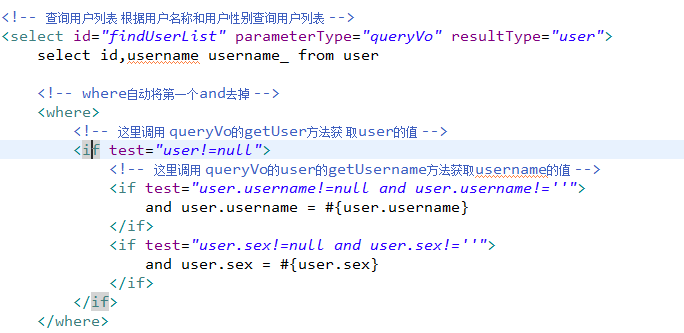
企业中常用resultType，针对sql查询结果定义pojo即可 。

# 动态 sql

Mybatis提供 了很多标签，用于拼接sql语句。

## if

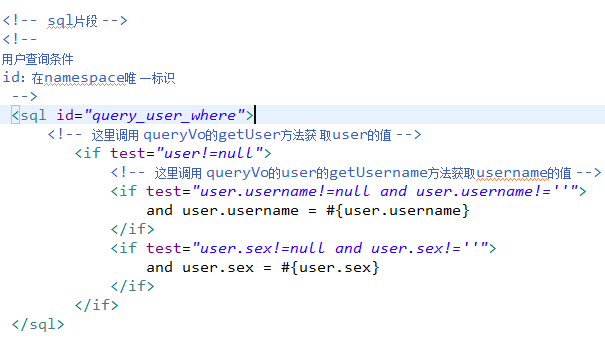
判断，如果条件满足，进行sql拼接。



## sql片段

sql片段：可以将一些sql定义成一个片段，可以被其它statement引用。

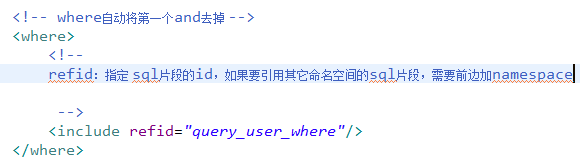
Sql片段定义：



开发经验 ：

建议以单表抽取sql，sql片段的可重用性才高。

Sql片段使用：



## foreach

需求 ：传入多个id查询用户

Sql语句：

两个方法：

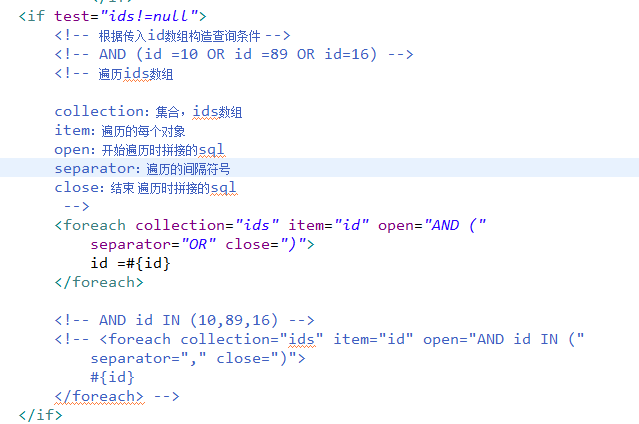
SELECT \* FROM USER WHERE username LIKE '%张%' AND (id =10 OR id =89 OR id=16)

SELECT \* FROM USER WHERE username LIKE '%张%' id IN (10,89,16)

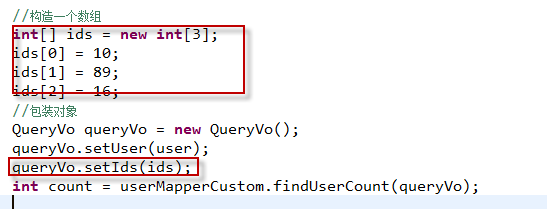
传入参数：使用整型数组存储多个id值，通过QueryVo传递整型数组 。



Mapper.xml：



测试 代码：



# resultMap

resultMap：如果sql查询字段名和pojo的属性名不一致，可以通过resultMap将字段名和属性名作一个对应关系 。

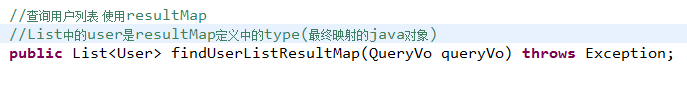
mapper.xml

定义resultMap:





Mapper.java



# resultType和resultMap区别

resultType：sql查询结果集使用resultType映射，要求sql查询字段名和pojo的属性名一致，才能映射成功。

resultMap： 如果sql查询结果集的字段名和pojo的属性名不一致，使用resultMap将sql查询字段名和pojo的属性作一个对应关系 ，首先需要定义resultMap，最终要使用pojo作为结果集映射对象。

建议：一般情况下建议使用resultType，因为简单方便。

针对一对多查询或要使用延迟加载 ，必须使用resultMap。

# 问题

org.apache.ibatis.exceptions.PersistenceException:

### Error querying database. Cause: org.apache.ibatis.reflection.ReflectionException: There is no getter for property named 'usernam' in 'class cn.itcast.mybatis.po.User'

### The error may exist in cn/itcast/mybatis/mapper/UserMapperCustom.xml

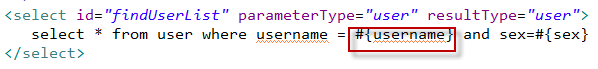
### The error may involve cn.itcast.mybatis.mapper.UserMapperCustom.findUserList-Inline

### The error occurred while setting parameters

### SQL: select \* from user where username = ? and sex=?

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

原因是：



#｛username｝，mybatis进行输入参数映射时，调用parameterType指定 的pojo的get方法。如果属性写错了，报上边的错误 。

org.apache.ibatis.exceptions.PersistenceException:

### Error building SqlSession.

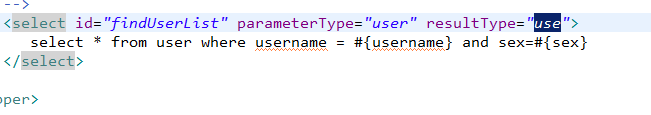
### The error may exist in cn/itcast/mybatis/mapper/UserMapperCustom.xml

### Cause: org.apache.ibatis.builder.BuilderException: Error parsing SQL Mapper Configuration. Cause: org.apache.ibatis.builder.BuilderException: Error parsing Mapper XML. Cause: org.apache.ibatis.builder.BuilderException: Error resolving class. Cause: org.apache.ibatis.type.TypeException: Could not resolve type alias 'use'. Cause: java.lang.ClassNotFoundException: Cannot find class: use

... 25 more

Caused by: org.apache.ibatis.builder.BuilderException: Error parsing Mapper XML. Cause: org.apache.ibatis.builder.BuilderException: Error resolving class. Cause: org.apache.ibatis.type.TypeException: Could not resolve type alias 'use'. Cause: java.lang.ClassNotFoundException: Cannot find class: use

原因是：



无法解析上边的”use”别名。