Mybatis

基础知识：

对原生态jdbc程序（单独使用jdbc开发） 问题总结

Mybatis框架原理

Mybatis入门程序： 用户的增删改查

Mybatis 开发DAO的两种方法：

原始DAO开发方法（程序需要编写DAO接口和DAO实现类）（历史遗留）

Mapper接口（相当于DAO接口）代理开发方法（现阶段）

Mybatis配置文件SqlMapConfig.xml

Mybatis 核心：

输入映射，

输出映射

Mybatis的动态SQL

高级知识：

数据模型分析

高级结果集映射（一对以，一对多，多对多）

Mybatis延迟加载

Mybatis查询缓存（一级缓存，二级缓存）

Mybatis和spring进行整合

Mybatis逆向工程

1. 对原生态JDBC程序中问题总结
   1. 环境

Java环境：jdk1.7.0\_72

Eclipse: indigo

Mysql: 5.1

* 1. 创建mysql数据库

导入以下脚本：

Sql\_data.sql：记录测试数据，在实际企业开发中，最后提供一个初始化数据脚本

Sql\_table.sql: 记录表结构

4张表： items,

orderDetail,

order,

users

* 1. jdbc程序

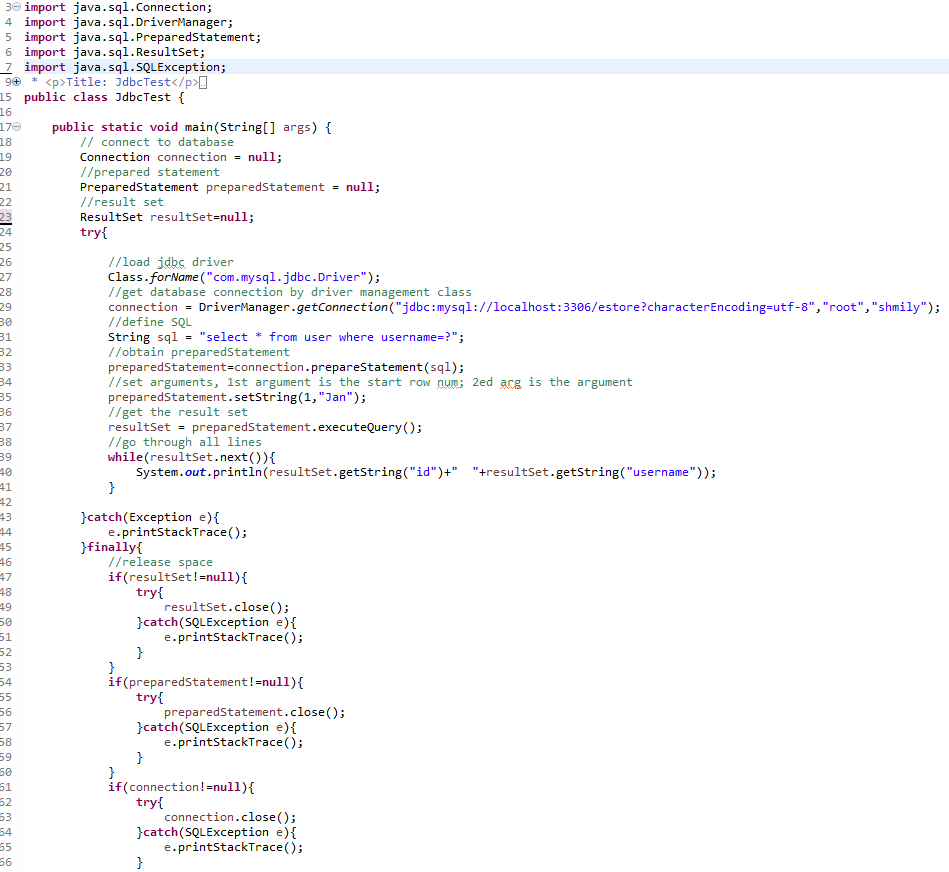
使用JDBC查询mysql数据库中用户表的记录

创建java工程，加入jar包：

Mysql-connector-java-5.1.5-bin.jar

预编译的statement(prepared): 通过这个statement向数据库发送SQL语句，

数据库进行编译后会储存在缓存中，下次再发如果SQL相同就不会再编译，可以提高数据库性能



* 1. 问题总结

1. 数据库连接，使用时就创建，不使用就立即释放，对数据库进行频繁开启和关闭，造成数据库资源的浪费，影响数据库性能。

解决方案：使用数据库的连接池管理数据库连接

1. 将SQL语句硬编码到java代码中，如果SQL语句要修改，需要重新编译java代码，不利于系统维护

设想：将SQL语句配置在XML配置文件中，即使sql变化，不需要对java代码进行重新编译

1. 向preparedStatement 中设置参数时，对占位符位置和设置的参数值硬编码在java代码中，不利于系统维护

设想：将SQL语句及占位符和设置参数值都配置在XML配置文件中，即使sql变化，不需要对java代码进行重新编译

1. 从resultSet中遍历结果集数据时，存在硬编码，将获取表的字段进行硬编码，不利于系统维护

设想：将查询结果集自动映射成java对象

1. mybatis框架
   1. mybatis是什么

是一个持久层框架，是apache下的顶级项目，后来托管到googlecode,现在在github

让程序员主要精力放在SQL上，通过mybatis提供的映射方式，自由灵活（半自动化，大部分需要程序员编写SQL）满足需要SQL语句

可以将向preparedStatement中输入的参数自动进行输入映射，将查询结果集灵活映射成java对象（输出映射）

* 1. mybatis框架图

SqlMapConfig.xml(mybatis的全局配置文件，名称是不固定的)，配置了数据源，事务等mybatis运行环境

配置映射文件（配置SQL语句）mapper.xml （映射文件）mapper.xml、。。。

输入参数类型：

Java简单类；

Hashmap

Pojo自定义

SqlSessionFactory（会话工厂）：根据配置文件创建工厂创建SqlSession

输出结果类型：

Java简单类；

Hashmap

Pojo自定义

SqlSession（会话）：是一个接口，面向用户（程序员）的接口，操作数据库（发出SQL增删查改）

Executor（执行器）：是一个接口（基本执行器，缓存执行器）sqlSession内部通过执行器操作数据库

Mapped statement（底层封装对象）：对操作数据库储存封装，包括sql语句，输入参数，输出结果类型

mysql

1. 入门程序
   1. 需求

根据用户id(主键) 查询用户信息

根据用户名称模糊查询用户信息

添加用户

更新用户

删除用户

* 1. 环境

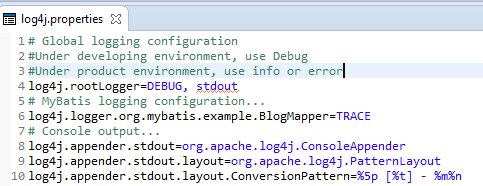
Mybatis运行环境（jar包）：从github下

lib下依赖包

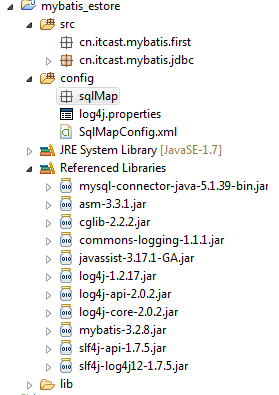
mybatis-3.2.7.jar核心包

加入mysql的驱动包

* 1. Log4j.properties



* 1. 工程结构



* 1. SqlMapConfig.xm

配置mybatis运行环境：数据源，事务管理



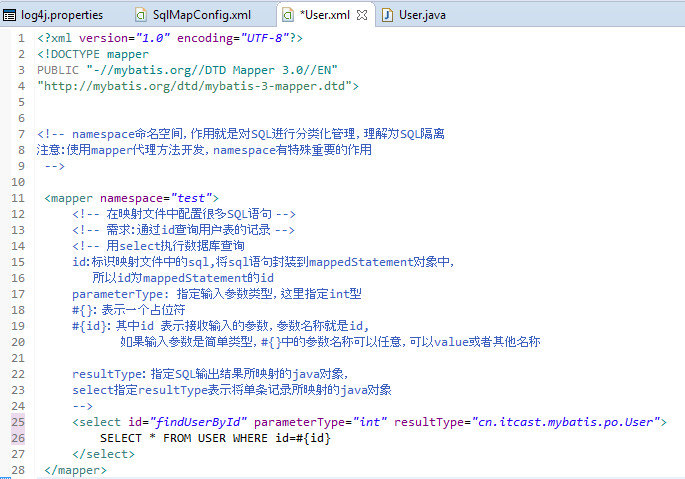
* 1. 功能1：根据用户ID（主键）查询用户信息
     1. 创建一个po类
     2. 映射文件

映射文件命名：

User.xml（原始ibatis命名）,

mapper代理开发映射文件名称叫XXXMapper.xml，如：UserMapper.xml、ItemMapper.xml等

在映射文件中配置SQL语句



* + 1. 在SqlMapConfig.xml中加载映射文件

在sqlMapConfig.xml中加载User.xml

<!-- 加载映射文件 -->

<mappers>

<mapper resource=*"sqlMap/User.xml"*/>

</mappers>

* + 1. 程序编写

**package** cn.itcast.mybatis.first;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** org.apache.ibatis.io.Resources;

**import** org.apache.ibatis.session.SqlSession;

**import** org.apache.ibatis.session.SqlSessionFactory;

**import** org.apache.ibatis.session.SqlSessionFactoryBuilder;

**import** org.junit.Test;

**import** cn.itcast.mybatis.po.User;

/\*\*

\* <p>Title: MybatisFirstr</p>

\* <p>Description: 入门程序 </p>

\* **@author** ChenAmber

\* **@date** 2016-10-4

\* **@version** 1.0

\*/

**public** **class** MybatisFirst {

//根据id查询用户信息，得到一条记录结果

@Test

**public** **void** findUserByIdTest() **throws** IOException{

//mybatis配置文件

String resource = "SqlMapConfig.xml";

//得到配置文件流

InputStream inputStream=Resources.*getResourceAsStream*(resource);

//创建会话工厂,传入mybatis的配置文件信息

SqlSessionFactory sqlSessionFactory= **new** SqlSessionFactoryBuilder().build(inputStream);

//通过工厂得到sqlSession

SqlSession sqlSession = sqlSessionFactory.openSession();

//通过SqlSession操作数据库

//第一个参数：映射文件中statement的id,等于=namespace+statement的id

//第二个参数：指定和映射文件中所匹配的parameterType类型的参数

//sqlSession.selectOne结果是与映射文件中所匹配的resultType类型的对象

User user = sqlSession.selectOne("test.findUserById", 1);

System.***out***.println(user);

//释放资源

sqlSession.close();

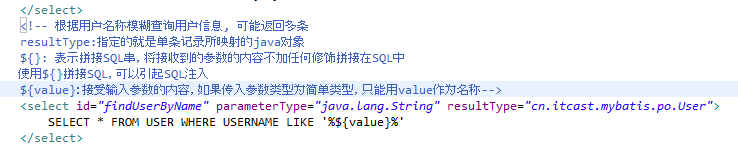
}

}

* 1. 功能2：根据用户名称模糊查询用户信息

3.7.1映射文件

使用user.xml，添加根据用户名称模糊查询用户信息的SQL语句



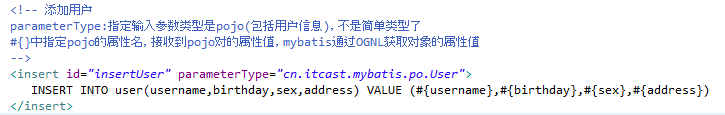
3.7.2 程序代码



* 1. 功能3：添加用户

3.8.1映射文件

在User.xml中配置添加用户的statement



3.8.2程序代码



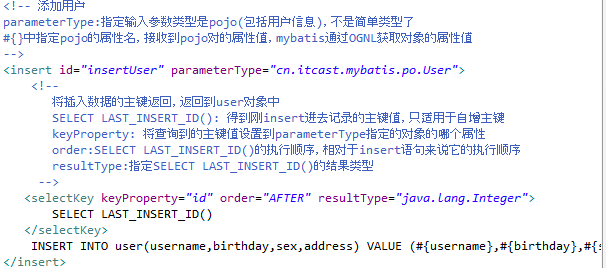
3.8.3自增主键返回

Mysql自增主键，执行insert提交之前自动生成一个自增主键。

通过mysql函数可以获取到刚刚插入记录的自增主键：LAST\_INSERT\_ID()

是在insert之后调用此函数。

需要修改insert SQL：

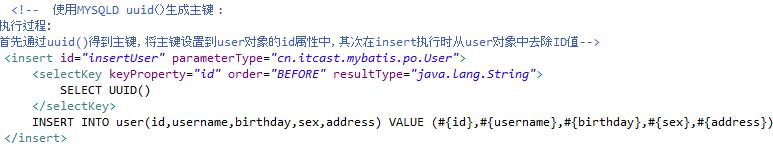


3.8.4非自增主键返回（使用uuid()）

使用mysql的uuid()函数生成主键，需要修改表中id字段类型为string,长度设置为35位;

执行的思路：先通过uuid()查询到主键，再将主键输入到SQL语句中，

所以执行UUID()语句顺序相对于insert语句是在之前

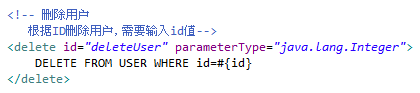


3.8.5通过ORACLE的序列生成主键

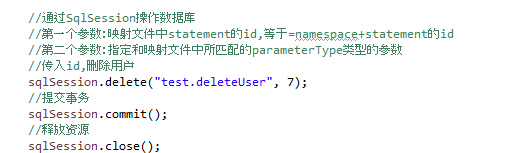
SELECT 序列名.nextval()

* 1. 功能4：删除用户

3.9.1映射文件

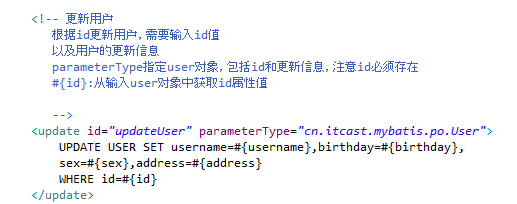


3.9.2程序代码



* 1. 功能5：更新用户

3.10.1映射文件



3.10.2程序代码



* 1. 小结
     1. parameterType

在映射文件中通过parameterType指定输入参数的类型

* + 1. resultType

在映射文件中通过resultType 来指定输出结果的类型

* + 1. #{} 和${}

#{}表示一个占位符，#{}接收输入参数，类型可以是简单类型，也可以是pojo,hashmap

如果接收简单类型，#{}可以写成value或其他名称

如果接收pojo对象值，通过OGNL读取对象中的属性值，通过属性.属性.属性…的方式获取对象属性值

${}表示一个拼接符号，会引起SQL注入，可能引起安全隐患，不建议使用

${}接收输入参数，类型可以是简单类型，也可以是pojo,hashmap

如果接收简单类型，${}只能写成value

如果接收pojo对象值，通过OGNL读取对象中的属性值，通过属性.属性.属性…的方式获取对象属性值

* + 1. SelectOne 和selectList

selectOne表示查询出一条记录来进行映射。如果使用selectOne可以实现使用selectList也可以实现（list中只有一条记录）

selectList表示查询出一个列表（多条记录）进行映射。如果使用selectList查询多条记录，不能使用selectOne, 会报错（toomanyresultsexception）

* 1. Mybatis和hibernate本质区别和应用场景

Hibernate: 是一个标准的ORM框架（对象关系映射）。入门门槛高，不需要程序员写SQL，SQL语句自动生成了。对SQL语句进行优化，修改比较困难。

应用场景：适用于需求变化不多的，较为固定的，中小型项目。比如：后台管理系统：erp、orm、oa等

Mybatis: 专注SQL本身，需要程序员自己编写SQL语句，SQL修改，优化比较方便。Mybatis是一个不完全的ORM框架，虽然程序员自己写SQL，mybatis也可以实现映射（输入映射，输出映射）。

应用场景：适用于需求变化较多的项目。比如：互联网项目。敏捷开发

要以低成本、高回报作为技术选型的原则。

1. Mybatis开发DAO的方法
   1. SqlSession使用范围
      1. SqlSessionFactoryBuilder

通过SqlSessionFactoryBuilder来创建会化工厂SqlSessionFactory

将SqlSessionFactoryBuilder当成一个工具类使用就可以，不需要使用单例模式来管理SqlSessionFactoryBuilder。在需要创建SqlSessionFactory时，只需要new一次SqlSessionFactoryBuilder即可。

* + 1. SqlSessionFactory

通过此来创建SqlSession, 使用单例模式管理SqlSessionFactory(工厂一旦创建，就一直使用一个实例)。将来和spring整合之后，使用单例模式管理SqlSessionFatcory。

* + 1. SqlSession

SqlSession是一个面向用户（程序员）的接口。

SqlSession中提供了很多操作数据库的方法：如：selectOne（返回单个对象）、selectList（返回单个或多个对象）。

SqlSession是线程不安全的，在SqlSession实现类中除了有接口中的方法（操作数据库的方法）还有数据域属性。

**SqlSession最佳应用场合是在方法体内，定义成局部变量。**多个线程访问一个对象，即使是用单例，每个线程对于不同方法都有不同的内存区域，所以不会有冲突。

4.2原始DAO开发方法（需要写DAO接口和dao实现类）

4.2.1 思路

需要向dao实现类中注入SqlSessionFactory，在方法体内通过SqlSessionFactory创建SqlSession

4.2.2 dao接口



4.2.3 dao实现类



4.2.4 dao实现类测试



4.2.5 原始DAO的开发问题

（1）DAO接口实现类方法中存在大量的模板方法，设想能否将这些代码提取出来，大大减轻程序员的工作量

（2）调用sqlSession方法时将statement的id硬编码了

（3）调用sqlSession方法时传入的变量，由于sqlSession方法使用泛型，即使变量类型传入错误，在编译阶段也不报错，不利于程序员发现问题。

* 1. mapper代理方法（程序员只需要写mapper接口，相当于dao接口）

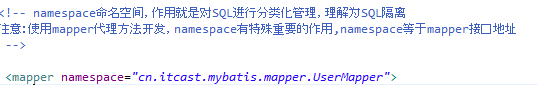
4.3.1思路（mapper代理的开发规范）

程序员需要编写mapper.xml 映射文件

程序员只需要写mapper接口（相当于dao接口），需要遵循一些开发规Mybatis 就可以自动的生成mapper 接口实现类代理对象

开发规范：

1. 在mapper.xml中的namespace 要等于mapper接口的地址



1. Mapper.java接口中的方法名和mapper.xml中statement的id要一致
2. Mapper.java接口中方法的输入参数类型和mapper.xml中statement的parameterType指定的类型一致
3. Mapper.java接口中方法的返回值类型和mapper.xml中statement的resultType指定的类型一致





总结：以上开发规范对下班的代码进行统一生产：

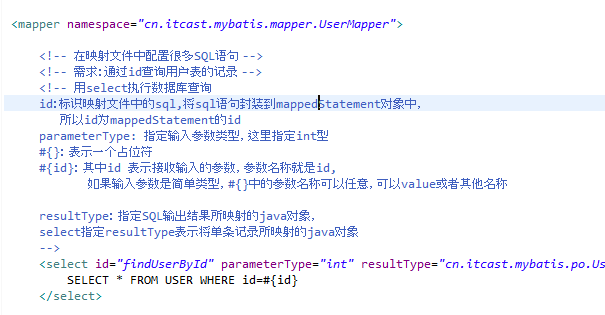
User user = sqlSession.selectOne(“test.findUserById”,id);

sqlSession.insert(“test.insertUser”,user);等

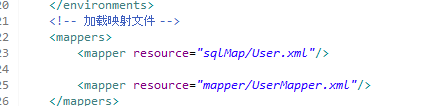
4.3.2 mapper.java(接口)



4.3.3 mapper.xml（映射文件）



4.3.4 要在SqlMapConfig.xml中加载mapper.xml



4.3.5测试



4.3.6 问题总结

4.3.6.1代理对象内部调用selectOne 或selectList

如果mapper方法返回单个pojo对象（非集合对象），代理对象内部就通过selectOne查询数据库；

如果mapper方法返回多个个pojo对象（集合对象），代理对象内部就通过selectList查询数据库；

要程序员自己判断决定返回结果个数

4.3.6.2mapper接口方法参数只能有一个是否影响系统开发？

是否不利于扩展维护？可以用包装类型

系统框架层，dao层的代码是被业务层公用的。

即使**mapper接口只有一个参数，可以使用包装类型的pojo满足不同的业务方法**的需求。

注意：持久层中方法的参数可以用pojo、map等，service方法中建议不要使用包装类型（不利于业务层的可拓展性）

1. SqlMapConfig.xml

Mybatis的全局配置文件SqlMapConfig.xml，配置内容如下：

Properties

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)

Enviroments（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）

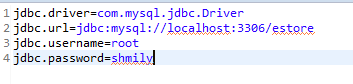
Mapper（映射器）

**5.1 Properties**

需求：将数据库连接参数单独配置在db.properties中，只需要在SqlMapConfig.xml

中加载db.properties的属性值，在SqlMapConfig.xml中就不需要对数据库的连接参数硬编码了

1. 将数据库的连接参数只配置在db.properties中，原因：方便对参数进行统一管理，其他xml文件也可以引用该db.properties



1. 在SqlMapConfig.xml中加载属性文件



1. Properties特性

注意： mybatis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中传递的属性，它会覆盖已读取的同名属性

建议：

不要在properties元素体内添加任何属性值，只将属性值定义在properties文件中

在properties文件定义属性名要有一定的特殊性，如：xxxx.xxxx.xxxx

**5.2 settings 全局参数配置**

Mybatis框架在运行时可以调整一些运行参数，会影响mybatis的运行行为

比如：开启延迟加载，开启二级缓存等等。详见mybatis/setting.xlsx

* 1. **typeAliases(别名)重点**

**5.3.1需求**

在mapper.xml中，定义很多的statement，statement需要parameter来指定输入参数的类型，需要resultType来指定输出结果的映射类型。

如果在指定类型是输入类型全路径，不便进行开发，可以针对parameterType或者resultType指定的类型定义一些别名，在mapper.xml中通过别名来进行定义，以方便开发。

**5.3.2**有一些默认支持的别名：

别名 映射的类型

\_byte byte

\_long long

\_short short

\_int int

\_integer integer

\_double double

\_float float

\_boolean boolean

string String

byte Btye

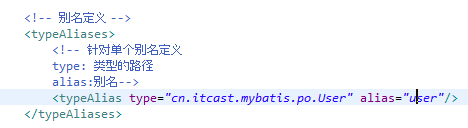
int Integer

date Date

…..

**5.3.3针对POJO自定义别名：**

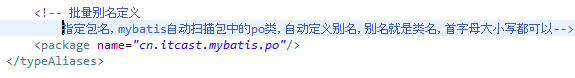
**5.3.3.1 单个别名定义**



**引用别名：**



**5.3.3.2批量别名定义**



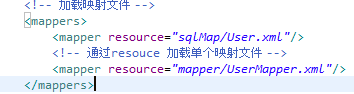
* 1. **typeHandlers 类型处理器**

mybatis中通过typeHandlers完成jdbc类型和java类型的转换

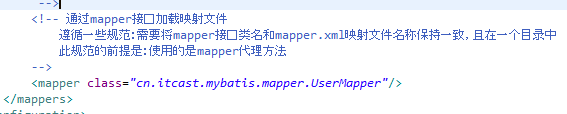
通常情况下，mybatis提供的类型处理器满足日常需要，不需要自定义

默认支持的类型处理器有：

* 1. **mappers(mapper配置，映射配置)**
     1. **通过resource或者url加载单个mapper文件**



* + 1. **通过mapper接口加载**



按照上面的规范：将mapper.java和mapper.xml放在一个目录下



* + 1. **通过package批量加载（推荐使用）**



1. 输入映射

通过parameterType来指定输入参数类型，类型可以是简单类型、hashmap、pojo的包装类型。

6.1 传递pojo的包装对象

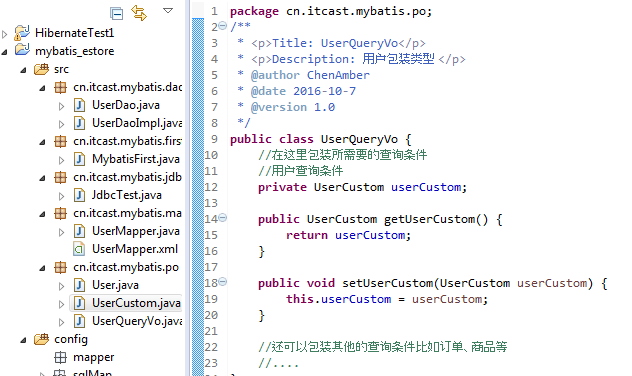
6.1.1 需求

完成给用户类型的综合查询，需要传入查询条件很复杂（可能包括用户信息、其他信息，比如商品、订单的）。

6.1.2定义包装类型pojo

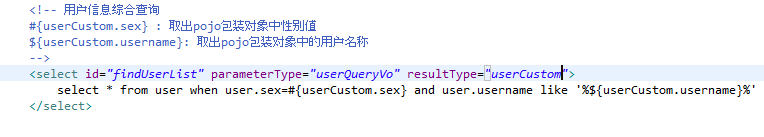
针对上面需求，建议使用自定义的包装类型的pojo.

在包装类型的pojo中，将复杂的查询条件包装进去

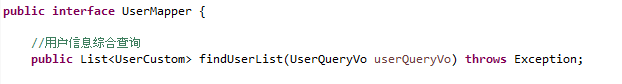


6.1.3 mapper.xml

在UserMapper.xml中定义用户信息综合查询（查询条件复杂，通过高级查询进行复杂关联查询）



6.1.4 mapper.java



6.1.5测试代码



6.2hashmap

1. 输出映射
2. 动态SQL