浙江工业大学

JavaEE技术实验报告

实验名称：MyBatis 基础应用

学 院： 计算机学院

班 级： 软件工程1801

姓 名： 陈南

学 号： 201806062101

组 号： 无

时 间： 2020年11月18日

**一、基础实验——MyBatis 框架搭建**

（一）实验目的

1、掌握 MyBatis 开发环境搭建的基本步骤，了解 MyBatis 的基本原理；

2、观察持久化类与数据库表的映射关系，观察相应的 Mybatis 映射文件配置，

并能够做简单应用；

3、观察 MyBatis 配置文件中的主要元素及属性配置，并能够做简单应用。

（二）基本知识与原理

1、MyBatis 是一个基于 Java 的持久层 ORM（Object-Relational Mapping）框架，

其提供的持久层框架包括 SQL Maps 和 Data Access Objets（DAO），几乎消除

了几乎所有的 JDBC 代码和参数的手工设置以及结果集的检索；

2、MyBatis 使用简单的 XML 或注解用于配置和原始映射，将接口和 Java 的 POJOs

映射成数据库中的记录；

（三）实验内容及步骤

1、登录 https://dev.mysql.com/downloads/connector/j/站点，下载并安装 MySQL

数据库；

2、在 MySQL 中创建一个名称为 mybatisdb 的数据库，并在该数据库中创建一个

名称为 user 的数据表，表结构如表 5-1 所示：

3、在表 customer 中添加若干条记录；

4、登录 http://downloads.mysql.com/archives/c-j/站点，下载 MySQL JDBC 驱动；5、在 Eclipse 中新建工程 mybatis-prj1，并添加 MySQL 驱动程序库文件到工程中；

6、登录 https://github.com/mybatis/mybatis-3/releases 站点，下载 MyBatis 发布

版，将核心 jar 包、依赖 jar 包添加到工程 mybatis-prj1 中，如图 5-1 所示；

图 5-1 MyBatis 相关 JAR 包

7、创建一个名为 com.mybatis.po 包，在该包中创建持久化类 MyUser，类中声明

的属性与数据表 user 的字段一致；

8、创建映射文件：创建一个名为 com.mybatis.mapper 的包，在该包中创建映射

文件 UserMapper.xml，并在其中配置操作数据库的 SQL 语句，具体代码如下：

<?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.mybatis.mapper.UserMapper">

<select id="selectUserById" parameterType="Integer"

resultType="com.mybatis.po.MyUser">

Select \* from user where uid = #{uid}

</select>

<select id="selectAllUser" resultType="com.mybatis.po.MyUser">

Select \* from user

</select>

<insert id="addUser" parameterType="com.mybatis.po.MyUser">

Insert into user (uname,usex) values(#{uname},#{usex})

</insert>

<update id="updateUser" parameterType="com.mybatis.po.MyUser">

update user set uname=#{uname}, user=#{usex} where uid=#{uid}

</update>

<delete id="deleteUser" parameterType="Integer">delete from user where uid = #{uid}

</delete>

</mapper>

9、创建配置文件：创建 MyBatis 的核心配置文件 mybatis-config.xml，在该文件

中配置了数据库环境和映射文件的位置，具体代码如下：

<?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="com.mysql.jdbc.Driver"/>

<property name="url"

value="jdbc:mysql://localhost:3306/mybatisdb"/>

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

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

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource="com/mybatis/mapper/UserMapper.xml"/>

</mappers>

</configuration>

10、创建测试类：创建一个名为 com.mybatis.test 的包，在该包中创建测试类

MyBatisTest，在其中使用输入流读取配置文件，然后根据配置信息构建

SqlSesstionFactory 对象，再通过 SqlSessionFactory 对象创建 SqlSession 对象，

并使用 SqlSession 对象的方法执行数据库操作，部分代码如下：

package com.mybatis.test;

import java.io.IOException;

import java.io.InputStream;

import java.util.List;

……

public class MyBatisTest {

Public static void main(String[] args){

try {

InputStream config=Resources.

getResourceAsStream("mybatis-config.xml");

SqlSessionFactory ssf=

new SqlSessionFactoryBuilder().build(config);SqlSession ss=ssf.openSession();

//查询一个用户

MyUser mu =

ss.selectOne("com.mybatis.mapper.UserMapper.selectUs

erById", 1)

System.out.println(mu);

//添加一个用户

MyUser addmu=new MyUser();

addmu.setUname("张三");

addmu.setUsex("男");

ss.insert("com.mybatis.mapper.UserMapper.addUser",addmu);

//修改一个用户

……

//删除一个用户

……

//查询所有用户

……

ss.commit();

ss.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

11、运行以上测试类，并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）简述 MyBatis 的工作原理；

（3）简述 MyBatis 和 Hibernate 的异同点和优缺点；

（4）碰到的问题及解决方案或思考；

（5）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**二、提高实验——映射器**

（一）实验目的1、进一步熟悉 MyBatis 应用的开发方法；

2、掌握 MyBatis 映射器的作用，熟悉映射文件中主要元素及其属性的含义和作

用，并能进行正确应用；

3、掌握 MyBatis 映射文件中的输入参数映射和输出结果映射。

（二）基本知识与原理

1、映射器是 MyBatis 最复杂且最重要的组件，由一个接口加上 XML 文件（SQL

映射文件）组成；

2、SQL 映射文件的常用配置元素如表 5-2 所示：

3、映射文件中<select>元素的常用属性如表 5-3 所示：

它和 Mapper 的命名空间组合起来使用，是唯一标识符，供 MyBatis 调用

parameterType 表

SQL 语句执行后返回的类型（全限定名或者别名）。如果是集合类型，

返回的是集合元素的类型。返回时可以使用 resultType 或 resultMap 之一

resultMap

它是映射集的引用，与< resultMap>元素一起使用。返回时可以使用

resultType 或 resultMap 之一

flushCache

它的作用是在调用 SQL 语句后，是否要求 MyBatis 清空之前查询本地缓

存和二级缓存。默认值为 false。如果设置为 true，则任何时候只要 SQL

语句被调用，都将清空本地缓存和二级缓存

useCache

启动二级缓存的开关。默认值为 true，表示将查询结果存入二级缓存中

timeout

用于设置超时参数，单位是秒。超时将抛出异常。

fetchSize

获取记录的总条数设定告诉 MyBatis 使用哪个 JDBC 的 Statement 工作，取值为 STATEMENT

statementType （ Statement ） 、 PREPARED （ PreparedStatement ） 、 CALLABLE

（CallableStatement）

这是针对 JDBC 的 ResultSet 接口而言，其值可设置为 FORWARD\_ONLY

resultSetType （只允许向前访问）、SCROLL\_SENSITIVE（双向滚动，但不及时更新）、

SCROLL\_INSENSITIVE（双向滚动，及时更新）

4、<resultMap>元素表示结果映射集，是 MyBatis 中最重要也是最强大的元素。

主要用来定义映射规则、级联的更新以及定义类型转化器等。<resultMap>元

素包含了一些子元素，结构如下：

<resultMap type="" id="">

<constructor><!-- 类在实例化时，用来注入结果到构造方法 -->

<idArg/><!-- ID 参数，结果为 ID -->

<arg/><!-- 注入到构造方法的一个普通结果 -->

</constructor>

<id/><!-- 用于表示哪个列是主键 -->

<result/><!-- 注入到字段或 JavaBean 属性的普通结果 -->

<association property=""/><!-- 用于一对一关联 -->

<collection property=""/><!-- 用于一对多、多对多关联 -->

<discriminator javaType=""><!-- 使用结果值来决定使用哪个结果映射 -->

<case value=""/> <!-- 基于某些值的结果映射 -->

</discriminator>

</resultMap>

其中，<resultMap>元素的 type 属性表示需要的 POJO，id 属性是 resultMap 的唯

一标识。子元素<constructor>用于配置构造方法（当 POJO 未定义无参数的构造

方法时使用）。子元素<id>用于表示哪个列是主键。子元素<result>用于表示 POJO

和 数 据 表 普 通 列 的 映 射 关 系 。 子 元 素 <association> 、 <collection> 和

<discriminator>是用在级联的情况下。

（三）实验内容及步骤

1、<select>元素的应用：在实际开发中，查询 SQL 语句经常需要多个参数，例如

多条件查询。当传入多个参数时，MyBatis 允许 Map 接口通过键值对传递多

个参数，对应 SQL 文件的代码片段如下：

<select id="selectAllUser" resultType="com.mybatis.po.MyUser"

**parameterType="map"**>

Select \* from user

where uname like concat(‘%’,#{u\_name},’%’)

and usex = #{u\_sex}</select>

2、修改测试代码，将查询所有用户信息的代码片段修改为“查询陈姓男性用户

信息”，并记录运行结果，代码片段如下所示：

public class MyBatisTest {

Public static void main(String[] args){

try {

……

Map<String, Object> map=new HashMap<>();

map.put("u\_name", "陈");

map.put("u\_sex", "男");

List<MyUser> list=**ss.selectList**

**("com.mybatismapper.UserMapper.selectAllUser",map)**;

For(MyUser myUser : list){

System.out.println(myUser);

}

……

} catch (IOException e) {

……

}

}

}

3、使用 Map 不能限定其传递的数据类型，若 SQL 语句较为复杂，参数很多，也

会造成 Map 使用很不方便，此时可以使用 JavaBean 传递多个参数。创建一

个名为 com.pojo 的包，在包中创建一个 POJO 类 SelectUserParam，代码片段

如下：

package com.pojo;

public class SelectUserParam {

private String u\_name;

private String u\_sex;

//省略 setters/getters 方法

}

4、修改映射文件 UserMapper.xml 中的“查询陈姓男性用户信息”的代码，代码

片段如下：

<select id="selectAllUser" resultType="com.mybatis.po.MyUser"

**parameterType="com.pojo.selectUserParam"**>

Select \* from user

where uname like concat(‘%’,#{u\_name},’%’)

and usex = #{u\_sex}

</select>

5、修改测试代码中“查询陈姓男性用户信息”的部分代码，并记录运行结果，

代码片段如下所示：SelectUserParam su = new SelectUserParam();

su.setU\_name("陈");

Su.setU\_sex("男");

List<MyUser> list=**ss.selectList**

**("com.mybatismapper.UserMapper.selectAllUser",su)**;

For(MyUser myUser : list){

System.out.println(myUser);

}

6、同样，在任何 select 语句中都可以使用 Map 存储结果，对应 SQL 文件的代码

片段如下：

<select id="selectAllUser" **resultType="map"**>

Select \* from user

</select>

7、修改测试代码，将查询所有用户信息的返回结果保存到 Map 中，并记录运行

结果。其中，Map 的 key 时 select 语句查询的字段名（必须完全一致），而

Map 的 value 时查询返回结果中字段对应的值，一条记录映射到一个 Map 对

象中。代码片段如下所示：

List<Map<String, Object>> lmp=**ss.selectList**

**("com.mybatismapper.UserMapper.selectAllUser")**;

For(MyUser myUser : list){

System.out.println(myUser);

}

8、在 select 语句中也可以使用 POJO 存储结果，此时可以使用 resultType 属性进

行自动映射，也可以使用 resultMap 属性进行更为复杂的映射或级联。此时，

需要在 com.pojo 包中创建一个 POJO 类 MapUser，代码片段如下：

package com.pojo;

public class MapUser {

private Integer m\_uid;

private String m\_uname；

private String m\_usex;

//省略 setters/getters 方法

public String toString(){

return "User [uid=" + m\_uid + ", uname=" + m\_uname +

", usex=" + m\_usex + "]";

}

}

9、在 SQL 映射文件 UserMapper.xml 中配置<resultMap>元素，其属性 type 引用

POJO 类，代码片段如下；

<resultMap **type="com.pojo.MapUser" id="myResult"**>

<!-- property 是 com.pojo.MapUser 类中的属性--><!-- column 是查询结果的列名，可以来自不同的表 -->

<id property="m\_uid" column="uid"/>

<result property="m\_uname" column="uname"/>

<result property="m\_usex" column="usex"/>

</resultMap>

10、修改映射文件 UserMapper.xml，在<select>元素中使用 resultMap 属性，代

码片段如下：

<select id="selectResultMap" **resultMap="myResult**">

Select \* from user

</select>

11、修改测试代码，使用 id 为 selectResultMap 的 select 操作完成查询，并记录

运行结果；

12、<insert>、<update>和<delete>元素用于映射插入、更新和删除语句，MyBatis

执行完此类语句后，将返回一个整数表示其影响的行数。尝试使用 resultMap

属性和 parameterMap 属性，完成增删改操作，并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）结合实验过程，总结 MyBatis 实现查询时返回的结果集由集中常见的存储

方式；

（3）碰到的问题及解决方案或思考；

（4）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**三、扩展实验——级联查询**

（一）实验目的

1、掌握 MyBatis 关联关系映射的基本概念，理解关联的方向和数量，重点理解

双向一对多/多对一的关联关系，及其在实际应用中的体现；

2、学习 MyBatis 框架处理一对多/多对一关联关系的方法，掌握相应 MyBatis 映

射文件的配置方案；

3、能在实际应用中通过 MyBatis 建立正确的一对多/多对一关联关系映射，并进

行数据库访问，特别是级联查询的相关实现。（二）基本知识与原理

1、MyBatis 中，通常通过<resultMap>元素的子元素<association>处理级联关系。

在<association>元素中，通常使用以下属性：

 property：指定映射到实体类的对象属性。

 column：指定表中对应的字段（即查询返回的列名）。

 javaType：指定映射到实体对象属性的类型。

 select：指定引入嵌套查询的子 SQL 语句，用于关联映射中的嵌套查询。

（三）实验内容及步骤

1、一对一级联查询：在数据库中创建两张数据表：身份证表 idcard，个人信息

表 person，创建代码如下：

2、创建对应的持久化类 Idcard 和 Person；

3、在 MyBatis 的核心配置文件 mybatis-config.xml 中打开延迟加载开关，代码片

段如下：

4、创建映射文件 IdCardMapper.xml，具体代码如下：5、创建映射文件 PersonMapper.xml，并在其中以 3 中方式实现“根据 id 查询个

人信息”的功能，具体代码如下：6、创建 POJO 类 com.pojo.SelectPersonById，代码片段如下：

7、创建数据操作接口 IdCardDao 以及其实现类，接口的具体代码如下：

8、创建数据操作接口 PersonDao 以及其实现类，接口的具体代码如下：

9、创建测试类 TestOneToOne 并记录运行结果，代码片段如下：10、尝试以用户和订单之间的一对多关系为例，实现一对多级联查询的处理过

程，并记录过程。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）碰到的问题及解决方案或思考；

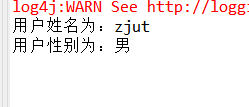
（3）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**一、基础实验——MyBatis 框架搭建**

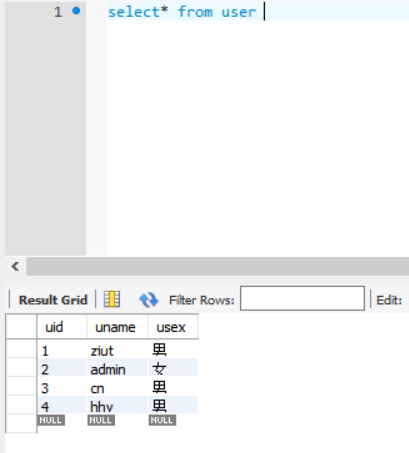
**（1）运行结果截图；**

1.查询一个用户（查询uid为1的用户信息）

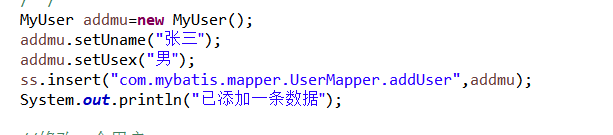


1. 添加一个用户

初始四个用户：



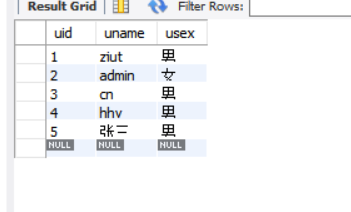
添加一个用户：



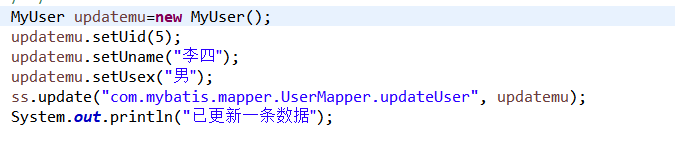
（工作台）

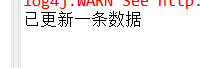


（数据库）



1. 修改一个用户

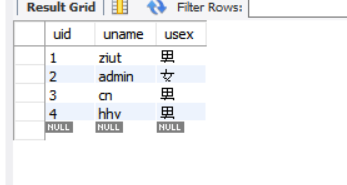






1. 删除一个用户

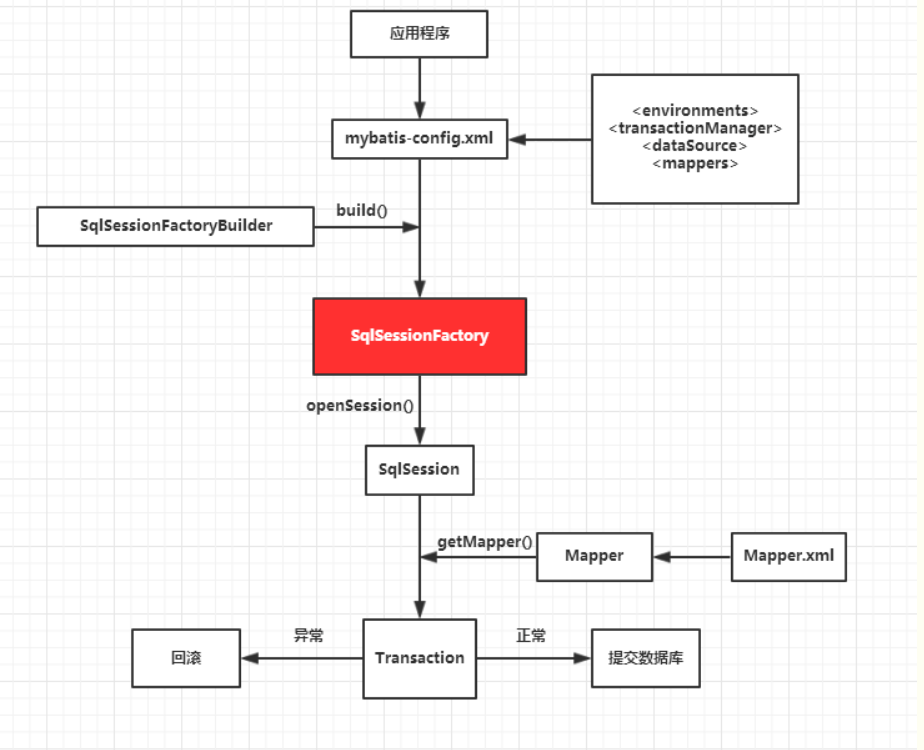




1. 查询所有用户



1. **简述 MyBatis 的工作原理；**

1）读取 MyBatis 配置文件：mybatis-config.xml 为 MyBatis 的全局配置文件，配置了 MyBatis 的运行环境等信息，例如数据库连接信息。  
2）加载映射文件。映射文件即 SQL 映射文件，该文件中配置了操作数据库的 SQL 语句，需要在 MyBatis 配置文件 mybatis-config.xml 中加载。mybatis-config.xml 文件可以加载多个映射文件，每个文件对应数据库中的一张表。  
3）构造会话工厂：通过 MyBatis 的环境等配置信息构建会话工厂 SqlSessionFactory。  
4）创建会话对象：由会话工厂创建 SqlSession 对象，该对象中包含了执行 SQL 语句的所有方法。  
5）Executor 执行器：MyBatis 底层定义了一个 Executor 接口来操作数据库，它将根据 SqlSession 传递的参数动态地生成需要执行的 SQL 语句，同时负责查询缓存的维护。  
6）MappedStatement 对象：在 Executor 接口的执行方法中有一个 MappedStatement 类型的参数，该参数是对映射信息的封装，用于存储要映射的 SQL 语句的 id、参数等信息。  
7）输入参数映射：输入参数类型可以是 Map、List 等集合类型，也可以是基本数据类型和 POJO 类型。输入参数映射过程类似于 JDBC 对 preparedStatement 对象设置参数的过程。  
8）输出结果映射：输出结果类型可以是 Map、 List 等集合类型，也可以是基本数据类型和 POJO 类型。输出结果映射过程类似于 JDBC 对结果集的解析过程。

**（3）简述 MyBatis 和 Hibernate 的异同点和优缺点；**

相同点：1.mybatis和hibernate都是持久层框架。  
不同点：1.hibernate是一个全ORM框架，mybatis是半ORM框架。   
2.就结构上来说，hibernate的sql是封装的，适合处理些简单逻辑， 对复杂的业务处理能力较弱。相对来说，mybatis 就比较 灵活，能 够使用 sql 处理一些复杂的业务逻辑。   
3.就学习上来说，hibernate理解起来比较复杂，学习速度慢，mybatis容易理解，学习快。  
4.就生产上，对有一些逻辑比较简单的产品来说，是有hibernate 能 够更好的,更快的完成开发。   
5.就缓存上来讲，hibernate有一级 缓存session和 二级缓存 sessionFactory。  
Hibernate 优势  
lDAO 层开发比较简单  
对对象的维护和缓存较。  
数据库移植性好（MyBatis在不同的数据库需要写不同 SQL）  
有更好的二级缓存机制，可以使用第三方缓存  
Mybatis优势  
可以进行更为细致的SQL优化，可以减少查询字段  
lMyBatis容易掌握，而Hibernate门槛较高

**（4）碰到的问题及解决方案或思考；**

1、 有不对应的表及属性，一直报错没有输出。

解决方案：仔细核对相应的代码，找到不对应的进行修改。

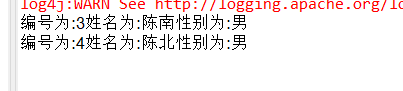
2.思考：初步了解了mybatis是如何正确运行和工作的，以及他的工作流程及原理，了解了他的优点与缺点。

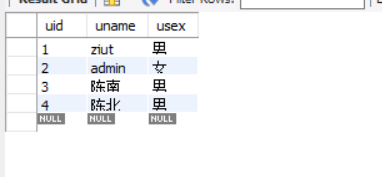
1. **实验收获及总结。**
2. 了解并实践了mybatis的操作流程及实现形式。
3. 对mybatis和hibernate之间的区别有了一定的认识和理解。

**二、提高实验——映射器**

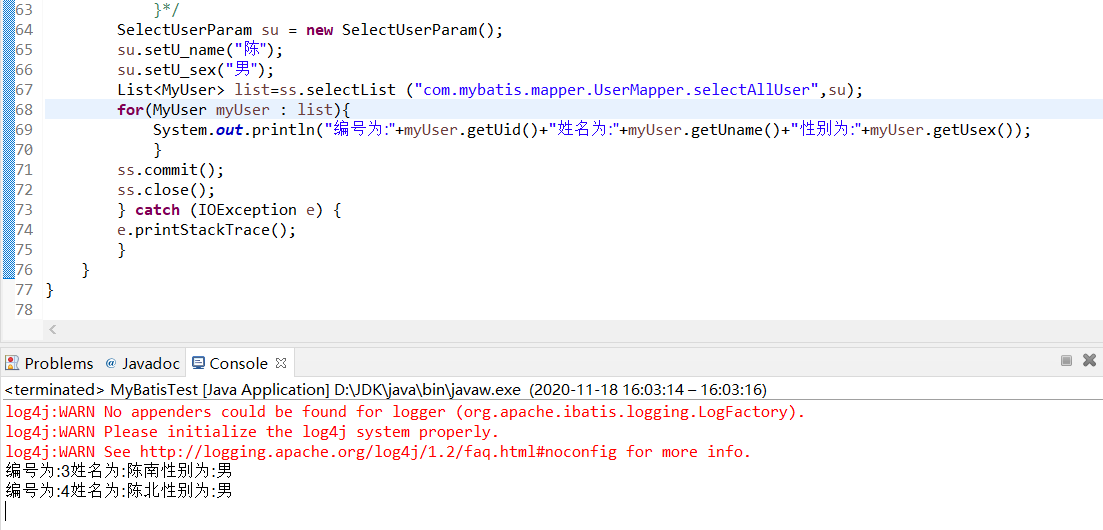
**（1）运行结果截图；**

**1.第二小题**





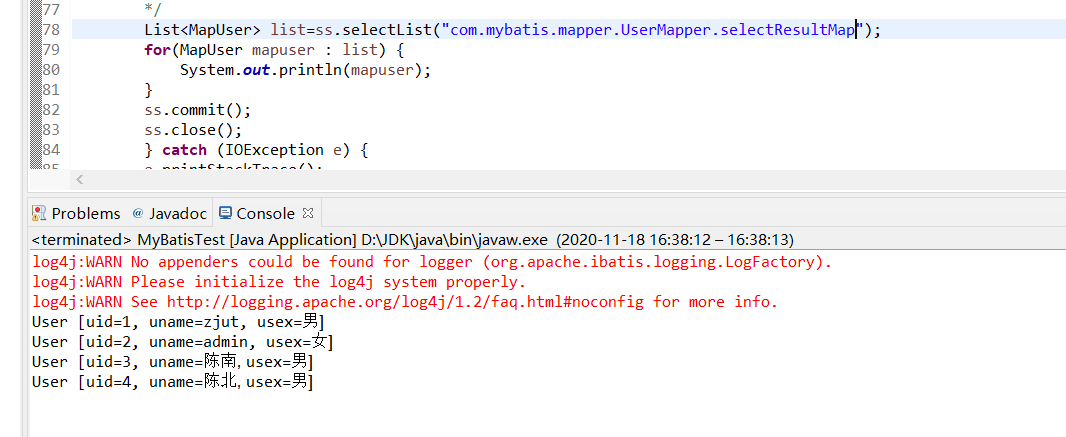
2.第五小题



1. 第七小题

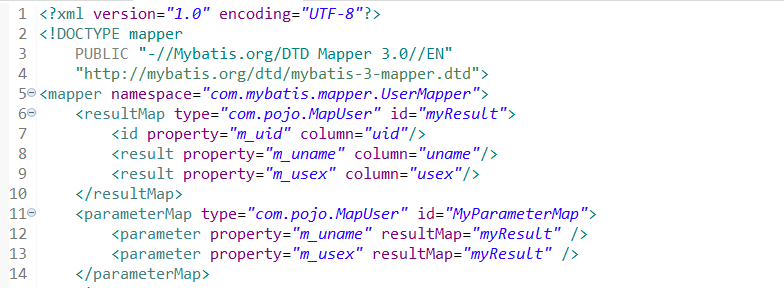


1. 第十一小题

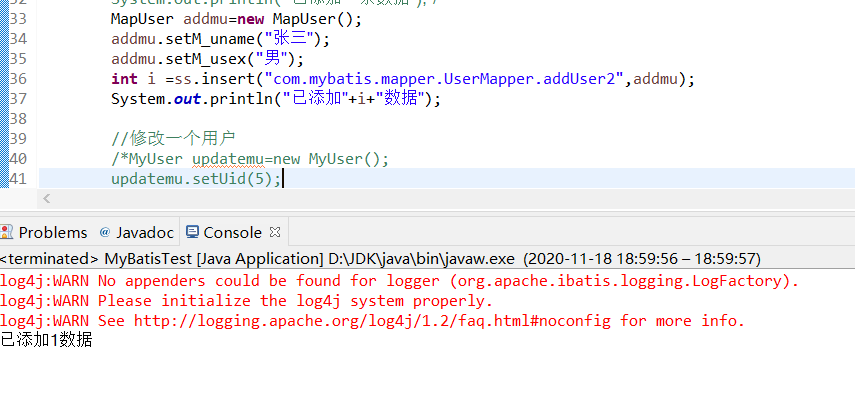


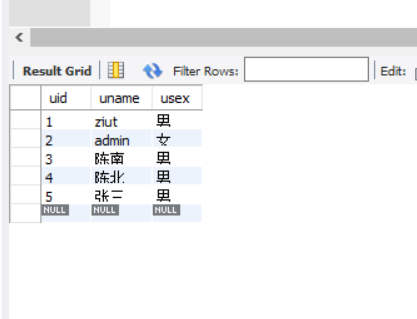
5.12小题

UserMapper.xml配置

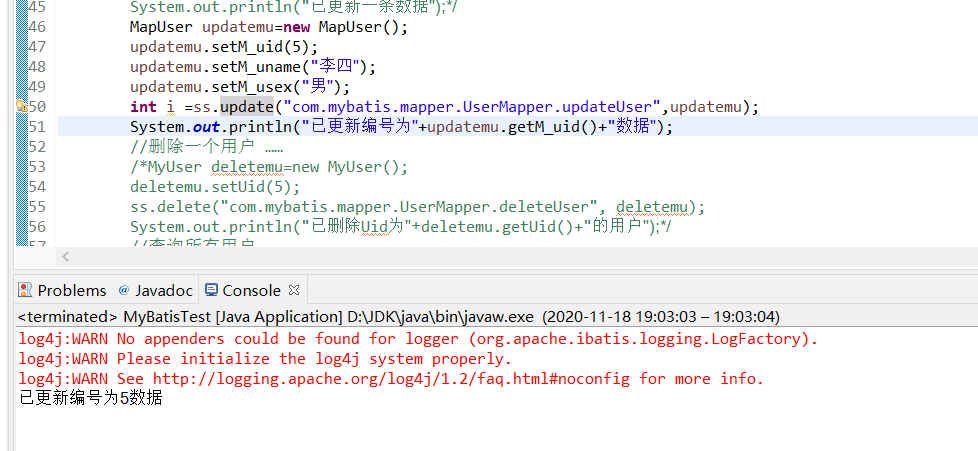


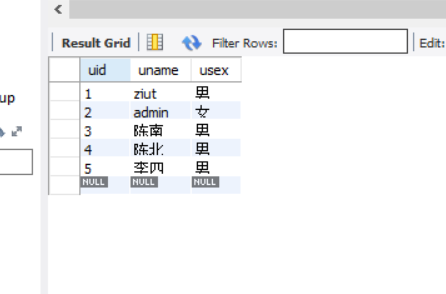
添加一条数据：





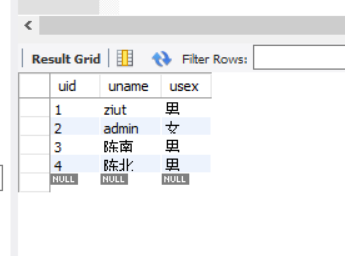
更新一条数据：





删除一条数据：





**（2）结合实验过程，总结 MyBatis 实现查询时返回的结果集由集中常见的存储**

**方式；**

1. 返回一个String
2. 返回一个int

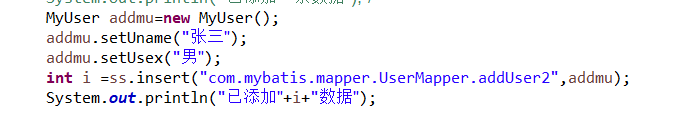
3.返回在map集中，key为select语句查询的字段名，value为查询结果  
返回pojo结果集得到一个pojo的list集合

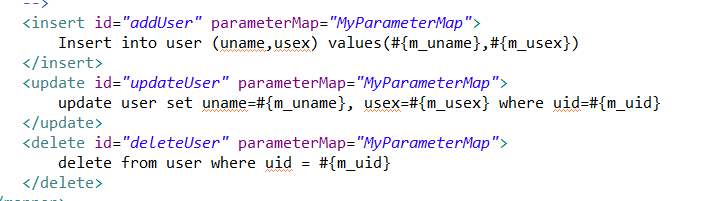
**（3）碰到的问题及解决方案或思考；**

1.写代码的时候有很多的小错误，关于大小写，中英文的引号，名称对应等等。

解决方案：根据报错信息，仔细的核对代码。

1. 在用parameterMap报错说无法在MyUser中找到m\_name的getter方法，在xml文件中找了半天错误，发现没有地方用到了关于MyUser的配置，最后在测试文件中发现错误。





解决方案：将xml文件中的m\_uname改成MyUser中的属性uname，或在测试文件中新建一个MapUser的对象，本次用第二种方式解决。

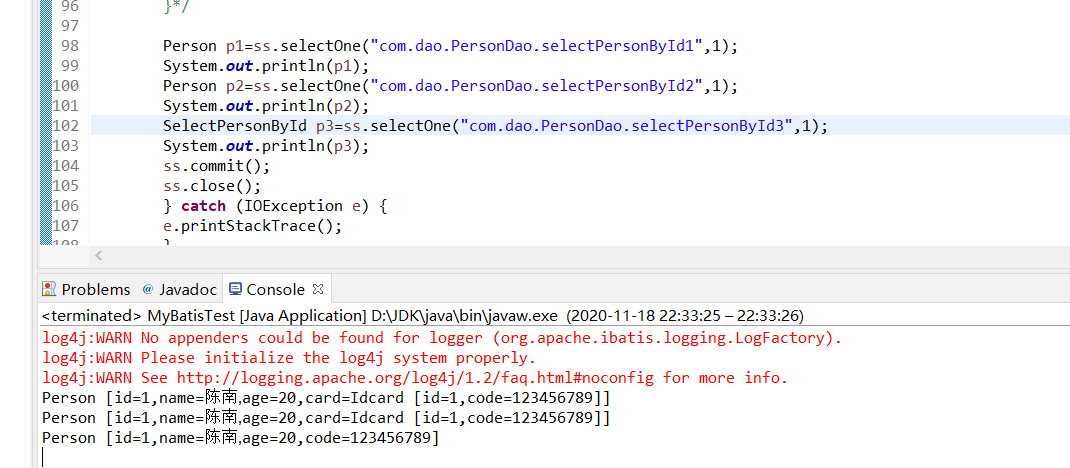
1. **实验收获及总结。**

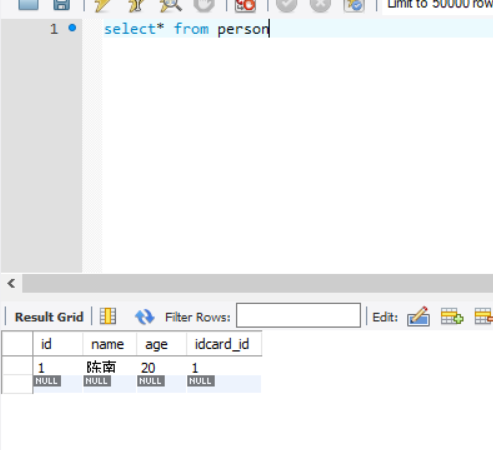
1. 了解了mybatis查询时所返回的结果集与对应的关系操作。

**三、扩展实验——级联查询**

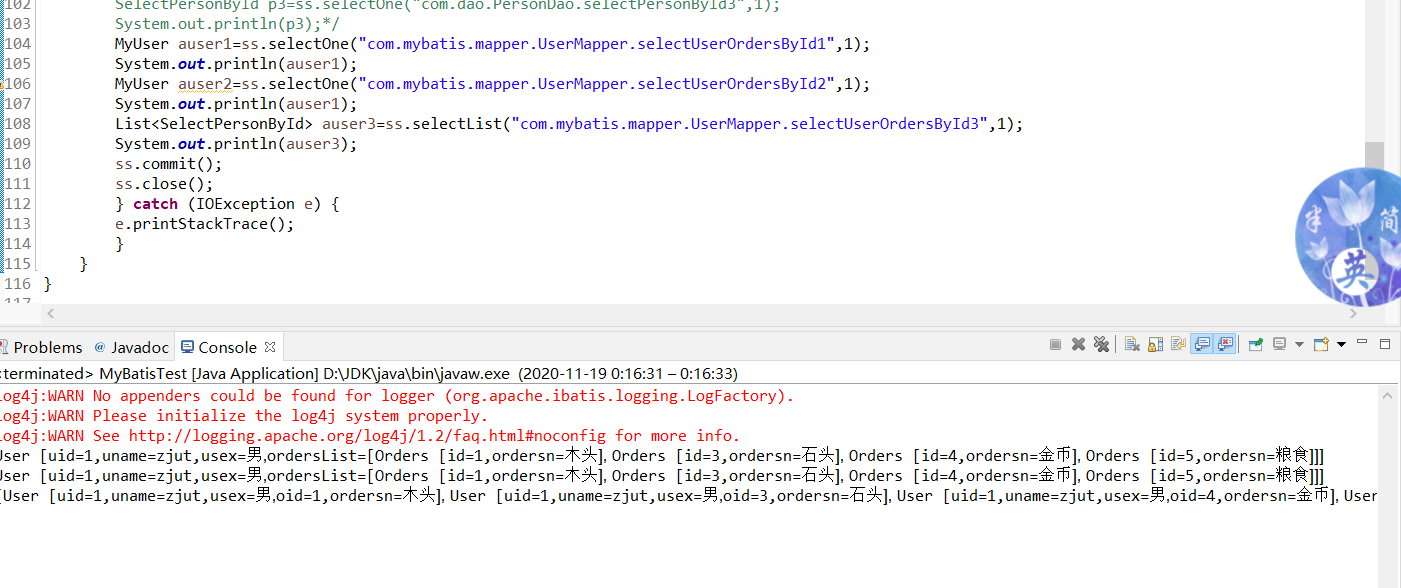
**（1）运行结果截图；**

1.第九小题



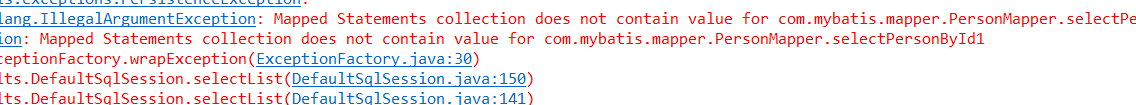


2.第十小题



**（2）碰到的问题及解决方案或思考；**

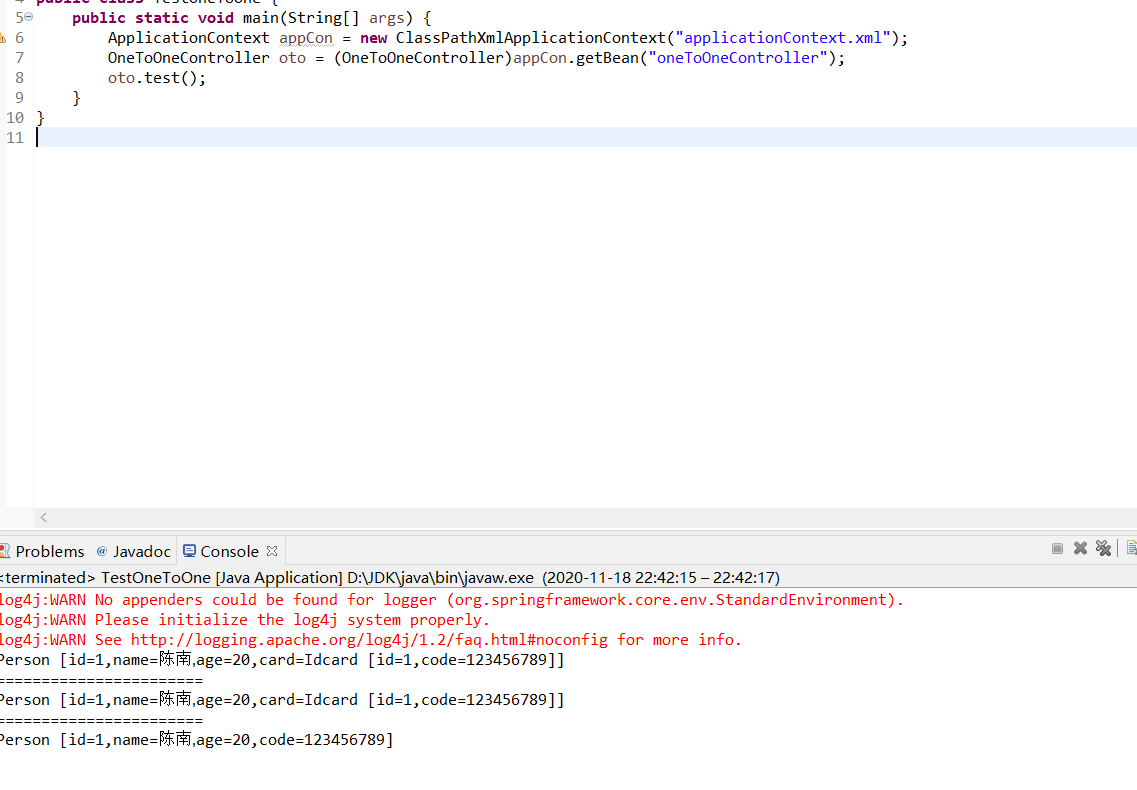
1.不熟悉如何配置，一开始写Personp1=ss.selectOne("com.dao.mybatis.mapper.PersonMapper.selectPersonById1",1);报错发现找不到



解决方案：改成Person p1=ss.selectOne("com.dao.PersonDao.selectPersonById1",1);发现selectPersonById前面那一串与xml文件中的namespace是相互对应的。

2.记得在mybatis-config.xml文件中加入相关的mapper，否则会痛苦很久。

3.尝试了书上的加入spring之后的结构模式。自己书写了一遍书上的方法，用com.controller的方式实现测试。也能成功运行。



**（3）实验收获及总结。**

1.了解并实践了mybatis中的级联操作。