# 本篇是本人看完所有Mybatis教程视频后总结的内容

# Mybatis框架学习2——Mybatis与Sql

## 主要学习内容：

Mybatis框架sql语句的传参与结果封装

Mybatis框架下的传统Dao实现方法

SqlMapConfig.xml配置文件的信息

## 基于代理dao实现CRUD

在第一部分中只用了一个查询所有用户的例子，在这个部分中会详细的学习其他CRUD的实现方法。

1.1 通过id查询用户（sql语句需要传入基本类型参数）

当通过id查询用户时，需要传入int类型的用户id，映射文件如下

<select id="findById" parameterType="INT" resultType=" "com.itheima.domain.User" ">

select \* from user where id = #{uid}

</select>

新加了parameterType属性表示参数类型，sql中的#{ }表示占位符，执行时被替换为实际的数据。

1.2 保存新用户（ sql语句需要传入类的对象）

当执行插入一个新用户操作时，需要传入一个user对象。

<insert id="saveUser" parameterType="com.itheima.domain.User">

insert into user(username,address,sex,birthday)values(#{userName},#{userAddress},#{userSex},#{userBirthday});

</insert>

此时传入User类，因此parameterType中写全限定类名。而#{ }中的名词要写User对象中的属性名称（因为在parameterType指定了类名，因此可以只写属性名）。

**注意：在对数据库实现增删改时，需要使用session.commit()来实现对事务的提交，下同。**

<selectKey keyProperty="userId" keyColumn="id" resultType="int" order="AFTER">

select last\_insert\_id();

</selectKey>

**另外，可以添加该配置来获取新增用户的id，使实体用户id同步。**

**1.3用户更新**

<update id="updateUser" parameterType="USER">

update user set username=#{userName},address=#{userAddress},sex=#{userAex},birthday=#{userBirthday} where id=#{userId}

</update>

**使用<update>标签，没太大区别。**

**1.4 用户删除**

<delete id="deleteUser" parameterType="java.lang.Integer">

delete from user where id = #{uid}

</delete>

**使用<delete>标签，没太大区别。**

**1.5 模糊查询**

**①sql语句占位符为#{ }时，查询方法传入参数时**就需要给定模糊查询的标识%。即

List users = userDao.findByName("%王%");

**②当占位符为’%${value}%’时（写法固定），此时传入的参数不需要表示%。**

List users = userDao.findByName("王");

#{}表示一个占位符号 通过#{}可以实现 preparedStatement 向占位符中设置值，自动进行 java 类型和 jdbc 类型转换， **#{}可以有效防止 sql 注入。** #{}可以接收简单类型值或 pojo 属性值。 如果 parameterType 传输单个简单类 型值，#{}括号中可以是 value 或其它名称。

${}表示拼接 sql 串 通过${}可以将 parameterType 传入的内容拼接在 sql 中且不进行 jdbc 类型转换， ${}可以接收简 单类型值或 pojo 属性值，如果 parameterType 传输单个简单类型值**，${}括号中只能是 value。**

**1.6 pojo包装对象**

**若将user作为pojo类的一个属性，在传参的时候用pojo类，则sql语句中的参数带上user类名即可。**

<select id="findUserByVo" parameterType="com.itheima.domain.QueryVo" resultType=" "com.itheima.domain.User" ">

select \* from user where username like #{user.username}

</select>

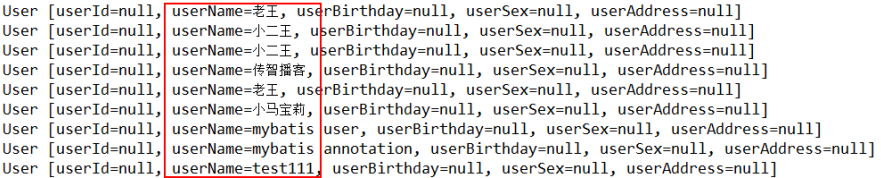
**1.7 小结**

**Mybatis与Jdbc相比，首先mybatis可通过在SqlMapConfig.xml中配置数据链接池来管理数据库连接（后面会学习），其次，mybatis将sql语句写在配置文件中，使之与代码分离。而且，mybatis自动将java对象映射至sql语句，并自动将sql执行结果映射至java对象。**

## 2、Mybatis 的输出结果封装

**2.1数据库列名和实体属性名称不一致**

当数据库列名和实体属性名称不一致时，执行查询所有操作，得到的结果如下。



只有userName有值的原因是，mysql数据库不区分大小写，因此username可以转为userName，但是可以看到其他名称不同的属性都无法转化,有以下两种解决方法。

**2.2 使用别名查询**

<select id="findAll" resultMap="userMap">

<!--select id as userId,username as userName,address as userAddress,sex as userSex,birthday as userBirthday from user;-->

select \* from user;

</select>

**比如在使用查询语句的时候将数据库中名称和实体属性名对应起来，解决无法转换的问题。**

**2.3 配置resultMap**

<resultMap id="userMap" type=" com.itheima.domain.User">

<!-- 主键字段的对应 -->

<id property="userId" column="id"></id>

<!--非主键字段的对应-->

<result property="userName" column="username"></result>

<result property="userAddress" column="address"></result>

<result property="userSex" column="sex"></result>

<result property="userBirthday" column="birthday"></result>

</resultMap>

**建立如上resultMap，建立列名和实体属性名的一一对应关系，将resultType标签替换为resultMap标签，即可通过resultMap将查询结果转化为实体类。**

id 标签：用于指定主键字段

result 标签：用于指定非主键字段

column 属性：用于指定数据库列名

property 属性：用于指定实体类属性名称

**2.4 小结**

**在这一部分，学习了用两种方法来实现数据库结果到实体类的转换，不仅如此，resultMap还可配置一对一、一对多的映射关系。**

## 3、Mybatis的DAO传统开发方式

在这个部分，将会学习基于传统编写 Dao 实现类的开发方式。但是由于是了解部分内容，不会写的太详细。

**3.1 IUserDao接口**

**这个接口即普通的持久层接口，包含了普通的CRUD方法。**

**3.2 UserDaoImpl类**

**这个类实现了IUserDao接口，有一个SqlSessionFactory类的对象，在每个接口的方法实现中，通过factory创建一个SqlSession,直接通过session的select、insert、update、delete等等方法来完成CRUD操作，结束后再关闭session。**

**3.3 执行测试**

**在执行时，直接通过相同的方法获得SqlSessionfactory，并将这个factory传入UserDaoImpl类创建一个Dao接口的实现类，再用这个实现类来完成CRUD操作即可。**

in = **Resources**.getResourceAsStream("SqlMapConfig.xml");

**SqlSessionFactory** factory = **new** **SqlSessionFactoryBuilder**().build(in);

sqlSession = factory.openSession();

//4.获取dao的代理对象

userDao = sqlSession.getMapper(**IUserDao**.class);

## 4、SqlMapConfig.xml配置文件

**在**SqlMapConfig.xml 中有很多可以配置的信息，如属性、别名、环境、映射器等等，视频中主要讲解了三个部分。

**4.1** properties(属性)

在使用 properties 标签配置时，我们可以采用两种方式指定属性配置。

第一种则是视频中使用的<property>标签，将名称和值一一对应起来。

<properties>

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

<property name="url" value="jdbc:mysql://localhost:3306/eesy\_mybatis"></property>

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

<property name="password" value="1234"></property>

</properties>

第二种则是通过读取一个属性配置文件来获取properties信息。在resource中创建一个properties文件比在其中写下配置信息，再用url来定位。

<properties url="file:///D:/IdeaProjects/day02\_eesy\_01mybatisCRUD/src/main/resources/jdbcConfig.properties">

</properties>

4.2 typeAliases(类型别名)

在 SqlMapConfig.xml 中配置：

<typeAliases>

<typeAlias type="com.itheima.domain.User" alias="user"></typeAlias>

<package name="com.itheima.domain"></package>

</typeAliases>

typeAlias用于配置别名。type属性指定的是实体类全限定类名。alias属性指定别名，当指定了别名就不再区分大小写

package用于指定要配置别名的包，当指定之后，该包下的实体类都会注册别名，并且类名就是别名，不再区分大小写

4.3 mappers（映射器）

4.3.1 <mapper resource=” ”/>

使用配置文件方法来映射Dao接口，配置的内容为配置文件的路径。

<mapper resource=”com/itheima/dao/IUserDao/xml”/>

4.3.2 <mapper class=” ”/>

使用注解的方法，写入内容指定被注解的dao**全限定类名**

<mapper class=”com.itheima.dao.IUserDao”/>

4.3 <package name=” ”/>

注册指定包下的所有 mapper 接口

<package name=”cn.itcast.mybatis.mapper”/>

**注意：使用**配置文件映射都要求 mapper 接口名称和 mapper 映射文件名称相同，且放在同一个目录中。

## 5、总结

总的来说，这部分学习内容主要讲的还是使用sql语句时的参数传入、类型封装以及结果转换等等实现方法，中间插入了一部分额外的内容，简单的学习了一下怎么使用传统的Dao开发方式。最后简单的对SqlMapConfig.xml配置文件的内容和流程进行了学习。