java基础(面向oop)---->java高级特性---->web前端(html、css、js、jquery)---->mysql---->
jsp/servlet(easyui、dbutils、redis)--->oracle---->ssm(mybatis、spring、springmvc)--->maven--->
ssh(struts2、spring、hibernate)------>svn--------->(springboot、linux)等

面向对象---->面向接口---->面向切面(面向sm)--->面向组件---->面向服务(dubbo)

持久层框架：mybatis(半自动化框架) ：操作数据库时需要使用sql
hibernate(全自动化框架) :操作数据库时不需要书写sql,需要书写hql

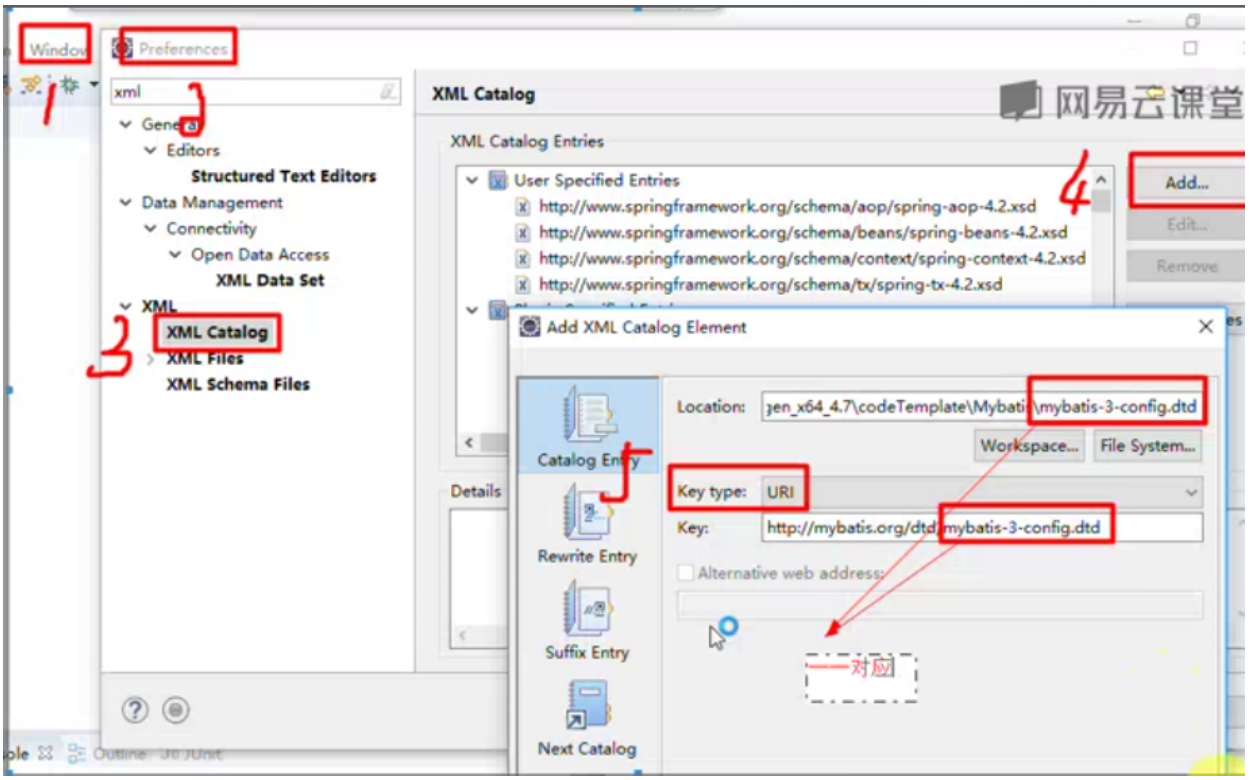
mybatis是Apache的一个开源项目，是一个基于java的持久层框框，一个半自动化持久层框架，支持定制SQL。操作数据库时，需要使用SQL语句。

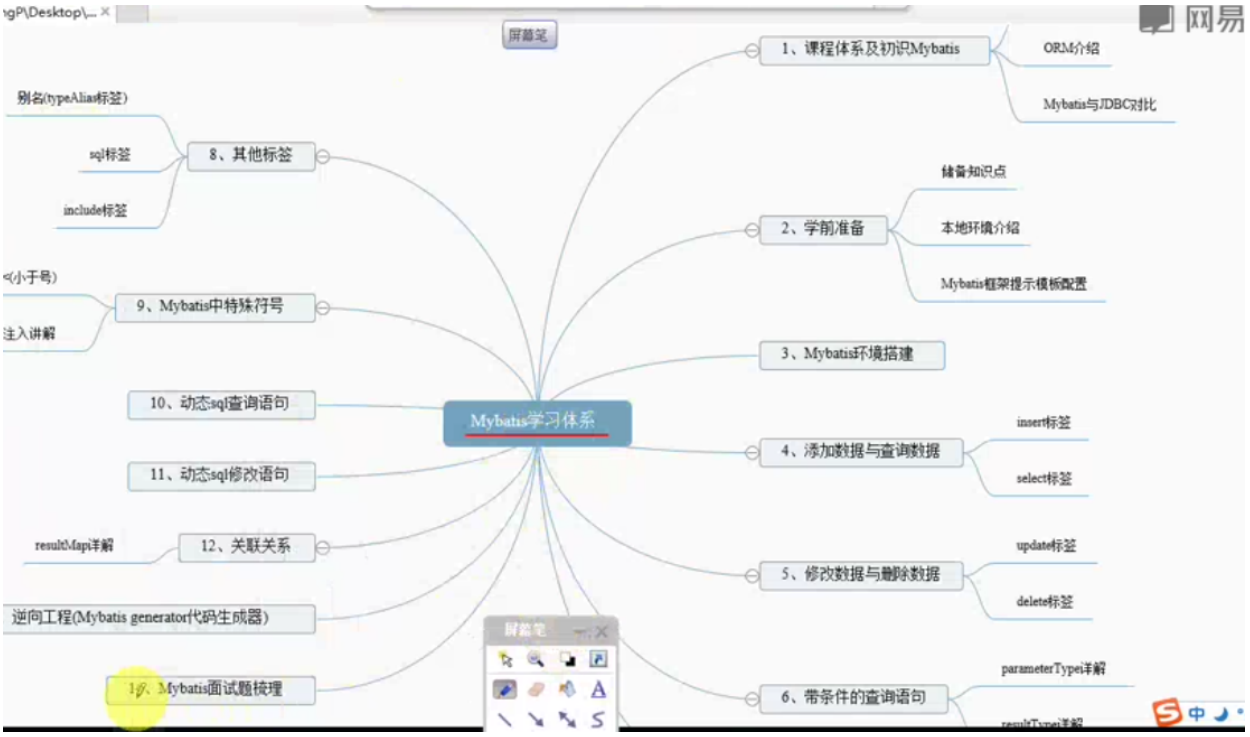
是一款操作数据库的框架。

MD5加密会变成32位。

java bean 的条件：属性私有，提供公有的属性设置方法，提供无参的构造函数

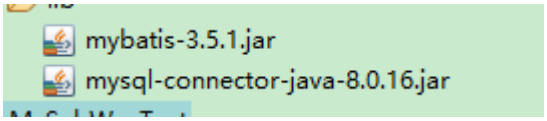
一，mybatis框架提示模板





二，环境搭建

1，导入mybatis需要的jar包

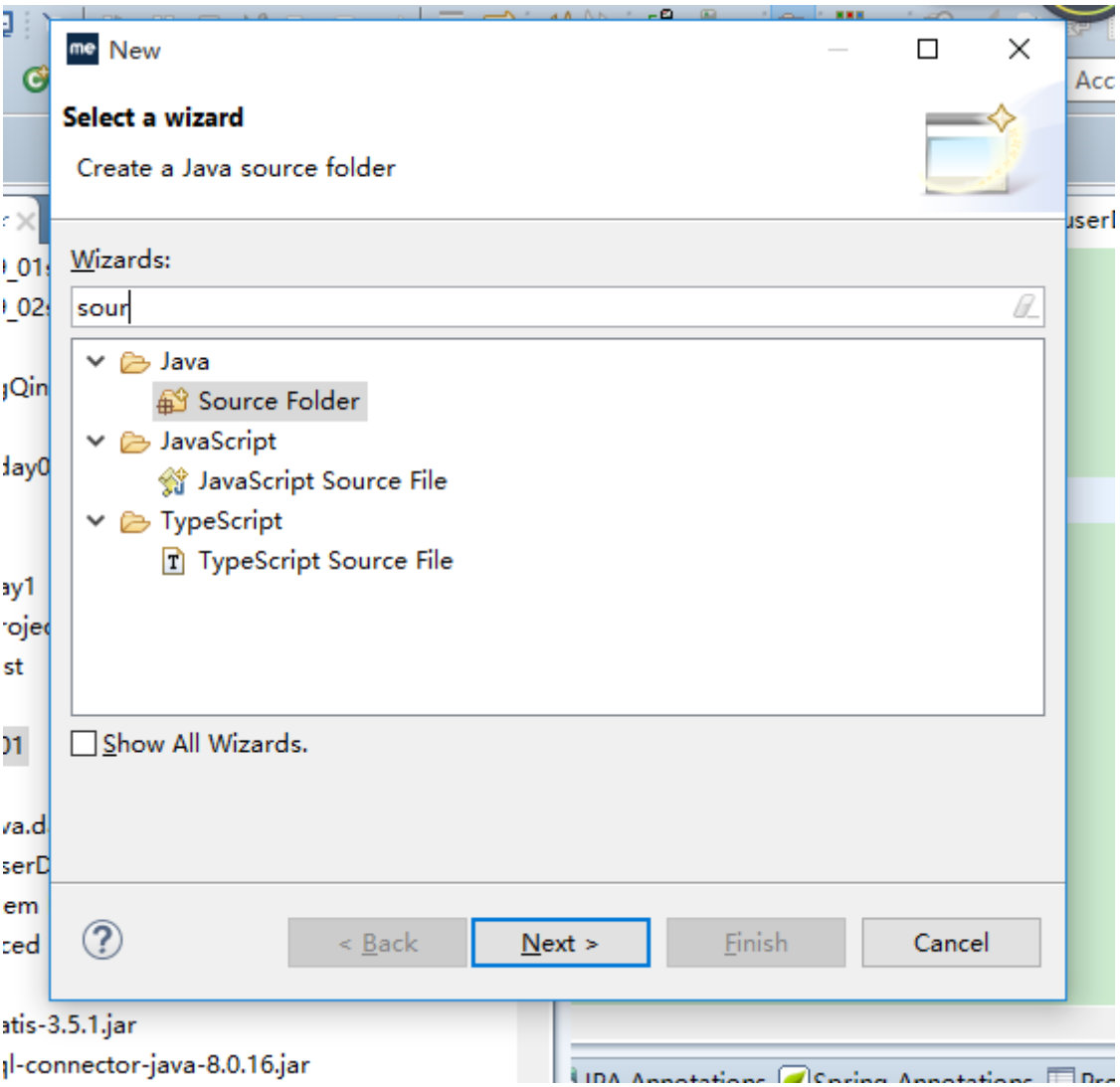


2，书写mybatis主配置文件（.properties、.xml，在框架中主要使用.xml类型的配置文件，其中mybatis的主配置文件是mybatis.xml）

（多条数据放在list<bean>或者list<map<string,object>>）

```
<?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>
<!-- 配置连接数据库的连接环境 -->
        <environments default="mysql">
        <!-- 开始配置mysql 可以配置多个<environment id="值">，如mysql、Oracle等，若实际应用中需要默认使用哪个数据库，则在上面的default写上mysql或者Oracle即可-->
                <environment id="mysql">
                                <!-- 配置事务 -->
```

```
                <transactionManager type="JDBC">
</transactionManager>
                <!-- 配置数据源 -->
                <dataSource type="POOLED">
                        <property name="driver"
value="com.mysql.cj.jdbc.Driver"/>
                        <property name="url"
value="jdbc:mysql://localhost:3306/db_mybatis?
useSSL=false&amp;serverTimezone=UTC&amp;characterEncoding=utf8"/>
                        <property name="username" value="root"/>
                        <property name="password"
value="AQL271422"/>
                </dataSource>
            </environment>
        </environments>
        <!-- 关联局部配置文件 -->
        <mappers>
        <mapper resource="cn/java/dao/impl/userDaoImpl.xml"/>
        </mappers>
</configuration>
```



3，书写局部配置文件（xxx.xml）---->放置SQL语句，名字与dao层java的类名一致，并将其放置在该类所在的java文件同一个目录下。

（除java文件使用.隔开，其它文件的路径使用/）存放SQL语句。

```
<?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">
<!-- namespace表示命名空间，一个局部配置文件只有一个命名空间。其值是某一个dao层类的具体路径 -->
<mapper namespace="cn.java.dao.impl.userDaoImpl">
<!-- sql语句保存在mybatis的局部配置文件中 ，增删改查的SQL语句需要放置在相对应的标签内部-->
        <!-- select存放查询语句，id属性在整个局部配置文件中必须唯一，其值SQL查询在dao层的方法名 -->
        <!-- resultType指定当前SQL查询返回的数据类型 ，存放某一条数据的数据类型
```

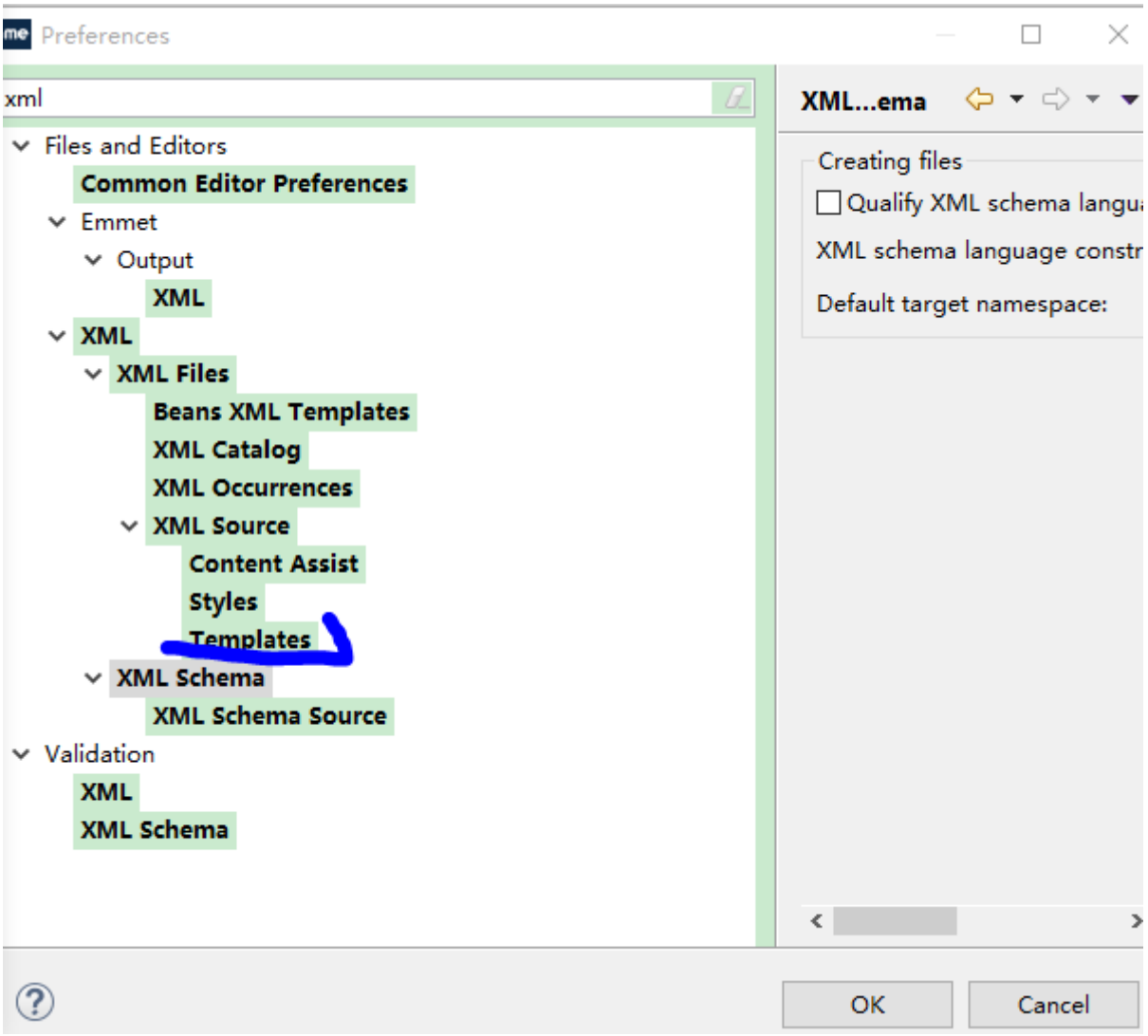
```
        类型不是SQL语句的最终类型，而是某一条数据的类型。其id值必须唯一，与dao层
类中的方法名保持一致
        -->
        <select id="getAllUser" resultType="cn.java.entity.user">
        <!-- sql语句后面的分号可要可不要 -->
        select * from users;
        </select>
</mapper>
```

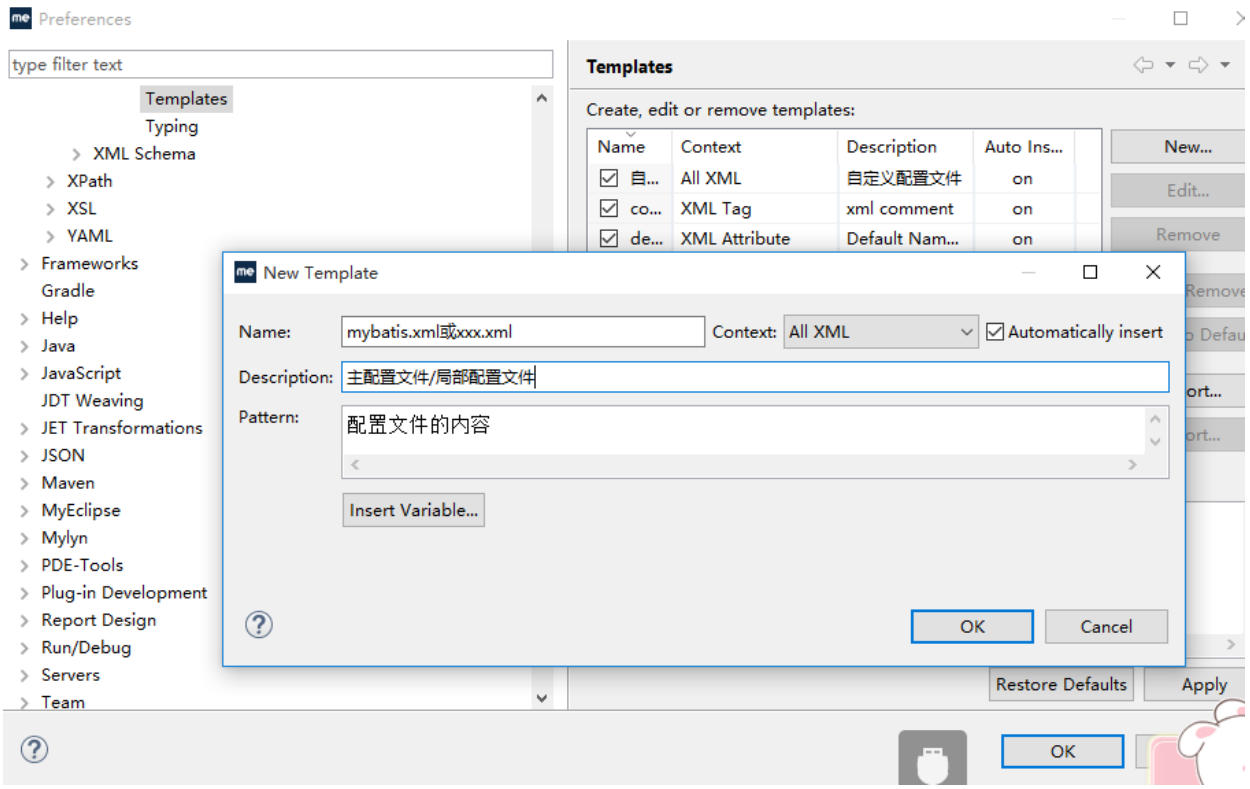
4，启动mybatis框架，测试

```
package cn.java.dao.impl;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;
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.java.entity.user;

public class userDaoImpl {
        @Test
        /*
         * 获取user中的所有数据记录
         */
public void  getAllUser() throws IOException {
        //1,启功mybatis框架SqlSession--->SqlSessionFactory---
>SqlSessionFactoryBuilder
        SqlSessionFactoryBuilder sdfBuilder = new SqlSessionFactoryBuilder();
        //将mybatis.xml文件转化为流
        InputStream ins = Resources.getResourceAsStream("mybatis.xml");
        SqlSessionFactory ssf = sdfBuilder.build(ins);
        SqlSession session = ssf.openSession();
```

```
        //2.调用局部配置文件中的SQL语句
        List<user> userlistList =
session.selectList("cn.java.dao.impl.userDaoImpl.getAllUser");
        for(user user:userlistList)
        {
                System.out.println(user.toString());
        }

}
}
```

5，配置模板（如图新建一个mybatis.xml和xxx.xml文件，将上述的俩个代码复制到输入框中，完成主配置文件和局部配置文件的配置）





1,排错的时候永远找第一个caused by.....

2,局部配置文件存放我们的SQL语句。

3,字段需要与数据库中的字段保持一致，否则会报错。

4,在mybatis尽量使用map，而不用bean。

5,查询语句只能放置在select语句中。

如果是多条记录的话，使用session.selectList（“该方法所在的类的路径+该方法的名称”），返回一个list，里面装的是map；如果是只返回一条记录的话，使用session.selectOne（“放置该方法所在的类的路径+该方法的名称”）

第一课时总结：

1，查询多条记录（不带任何参数）操作数据库的都放在dao层。

```
<!--
select语句返回list<map<string,object>>，一个map存放一个记录，
resultType的值可以是Map或map或java.util.map
 -->
<select id="getAllUser1" resultType="Map">
<!-- sql语句后面的分号可要可不要 -->
select * from users;
</select>
```

2，一般在查询的过程中，将查询的结果保存在map集合中，这样的话，在启动项目的时候，就不需要再进行创建一个新的对象了，每次创建新的实体类对象很麻烦，并且如果使用实体对象的话，就强烈要求实体类中的属性字段要和数据库的表中的字段列名保持一致。（一条记录可以放在map中，也可以放在list<bean>中）

3，单条记录查询（注意Junit中不可以有参数）

```
<select id="getNameById" resultType="map">
select *from users where id =1;
</select>
```

```
@Test
/*
 * 按照条件查询一条记录
 */
public void getNameById() throws IOException {
    SqlSessionFactoryBuilder ssf = new SqlSessionFactoryBuilder();
    InputStream ins = Resources.getResourceAsStream("mybatis.xml");
    SqlSessionFactory sqlSessionFactory = ssf.build(ins);
    SqlSession session = sqlSessionFactory.openSession();
    List<Map<String, Object>> list = session.selectList("cn.java.dao.impl.userDaoImpl.getNameById");
    for (Map<String, Object> map : list) {
        System.out.println(map);
    }
}
```

4，带一个参数的条件查询：

```
@Test
/*
 * 根据传递的参数查询数据信息
 */
public void getUsersByTiaojian() throws IOException {
    SqlSessionFactoryBuilder ssf = new SqlSessionFactoryBuilder();
    InputStream ins = Resources.getResourceAsStream("mybatis.xml");
    SqlSessionFactory sqlSessionFactory = ssf.build(ins);
    SqlSession session = sqlSessionFactory.openSession();
    Long id = 1L;
    System.out.println("带条件的查询：");
    List<Map<String, Object>> list = session.selectList("cn.java.dao.impl.userDaoImpl.getUsersByTiaojian", id);
    for (Map<String, Object> map : list) {
        System.out.println(map);
    }
}
```

在局部配置文件中写如下代码：

```
<!--
parameterType：指定参数类型
 -->
<select id="getUsersByTiaojian" resultType="map" parameterType="long">
<!-- #{角标}其中角标从0开始，0表示第一个参数 -->
select *from users where id =#{0};
</select>
```

角标相当于java中的数组。parameter：用于指定参数的数据类型。并且当数据需要动态传入的时候，一定需要写明其数据类型是什么样的？