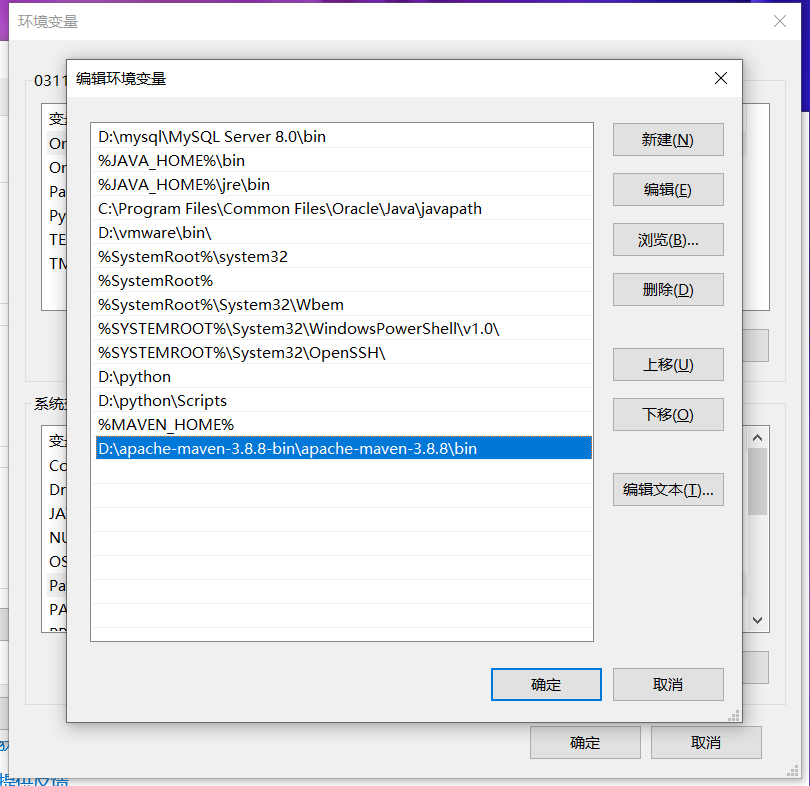
## Maven

下载安装

### 配置本地环境变量



### 配置maven配置文件

Config文件中的setting文件

1. 配置本地仓库
2. <localRepository>D:\java-localrepository</localRepository>

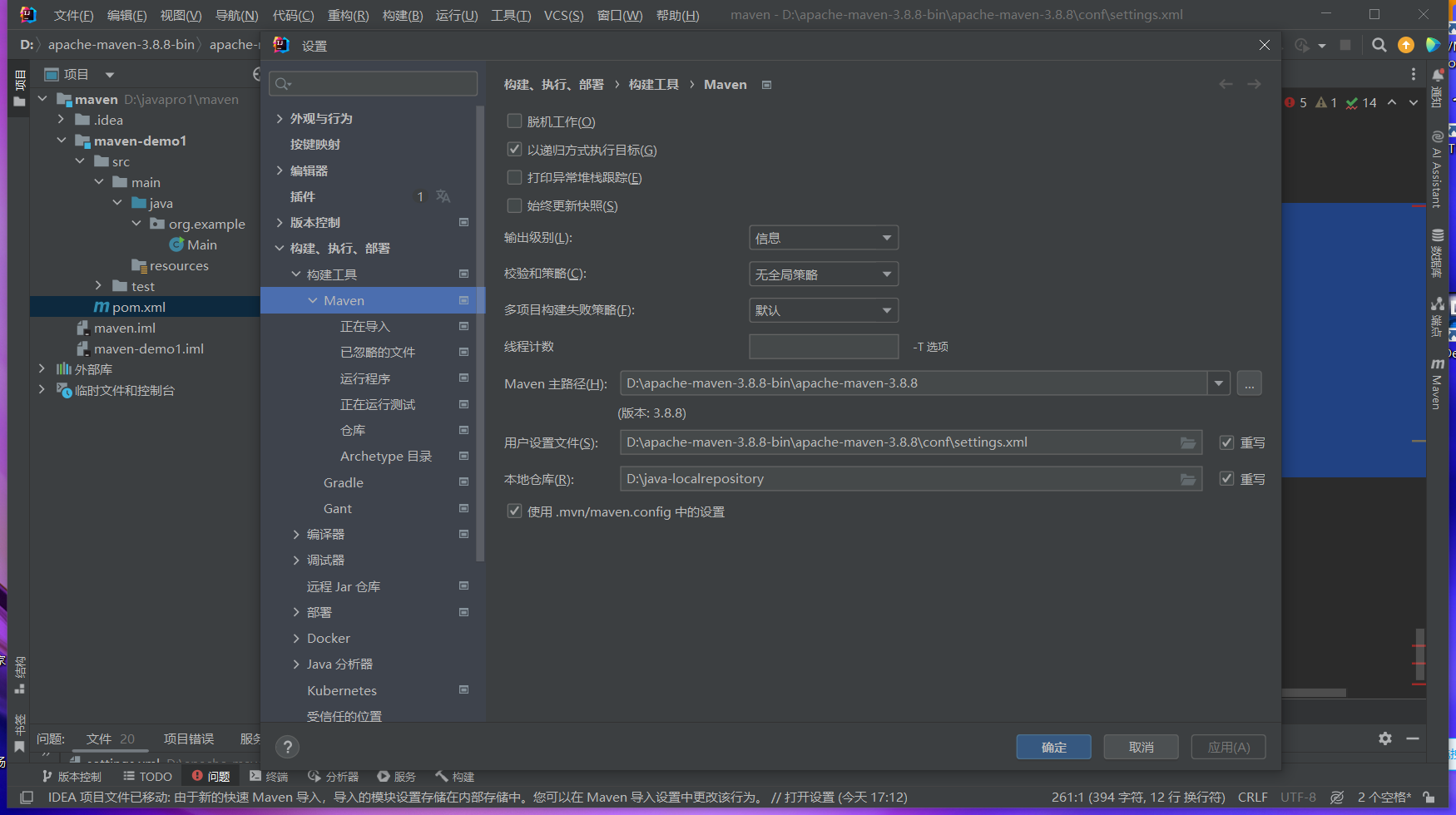
2，配置ali镜像

1. <mirror>  
    <id>alimaven</id>  
    <mirrorOf>central</mirrorOf>  
    <name>aliyun.maven</name>  
    <url>http://maven.aliyun.com/nexus/comtent/groups/public</url>  
    <!--<blocked>true</blocked>-->  
   </mirror>

3，配置jdk版本

<profile>  
 <id>jdk-17</id>  
 <activation>  
 <activeByDefault>true</activeByDefault>  
 <idk>17</idk>  
 </activation>  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <maven.compiler.compilerVersion>17</maven.compiler.compilerVersion>  
 </properties>  
 </profile>  
</profiles>

### 配置idea中的maven



### 构建命令



### 依赖范围

Compile，可编译可测试可运行（打成包）

text，可测试

provided，可编译可测试

optional标签 中止依赖传递



依赖的Build标签

打包方式packing

名称filename

配置打包的名称和特定位置文件，相关插件

依赖冲突排除exclusion标签

路径短>配置早

Maven继承

Packing标签为pom，在父类中新建立子类

父类maven

依赖管理

Dependencymanagement标签子maven选择父中的依赖

Maven聚合（打包，一键管理）

Module标签为路径

## Spring

## Mybatis

Sql动态

### 创建maven工程

导入以下依赖

Mybatis

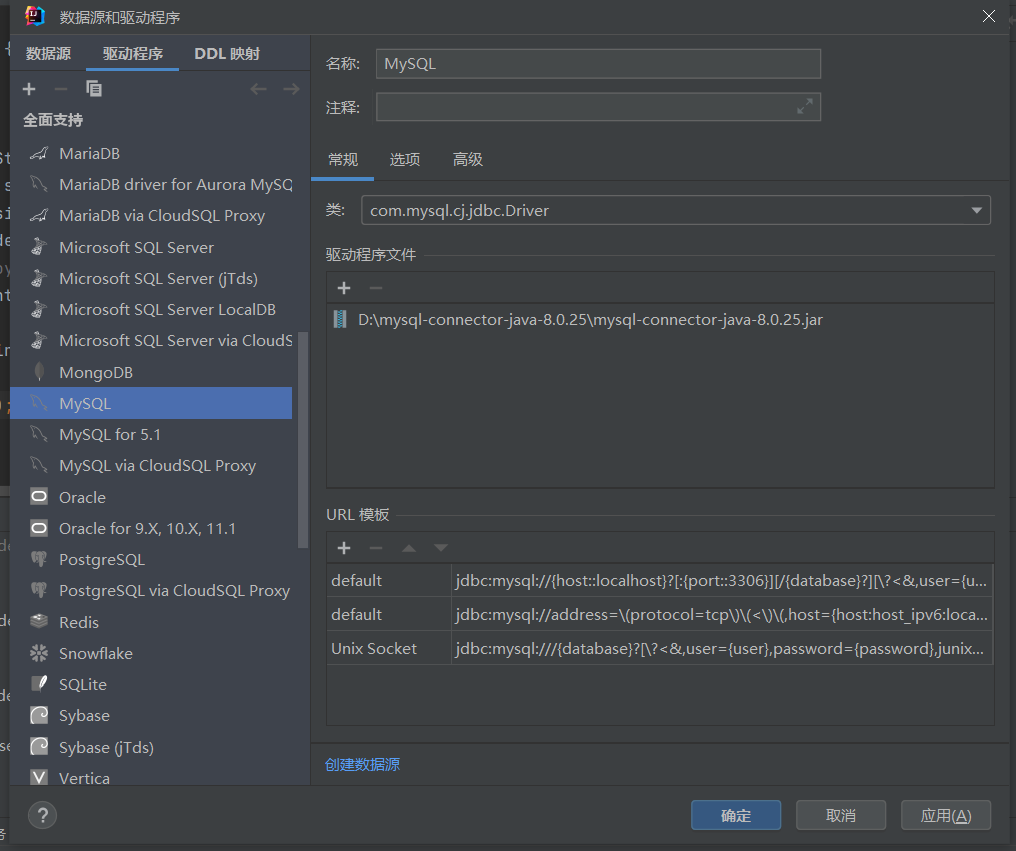
Mysql连接

Text测试

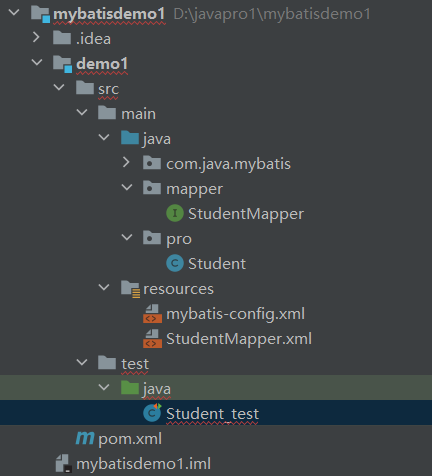
<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.5.11</version>  
</dependency>  
  
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.26</version>  
</dependency>  
<dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-api</artifactId>  
 <version>5.3.1</version>  
</dependency>

### 数据库连接

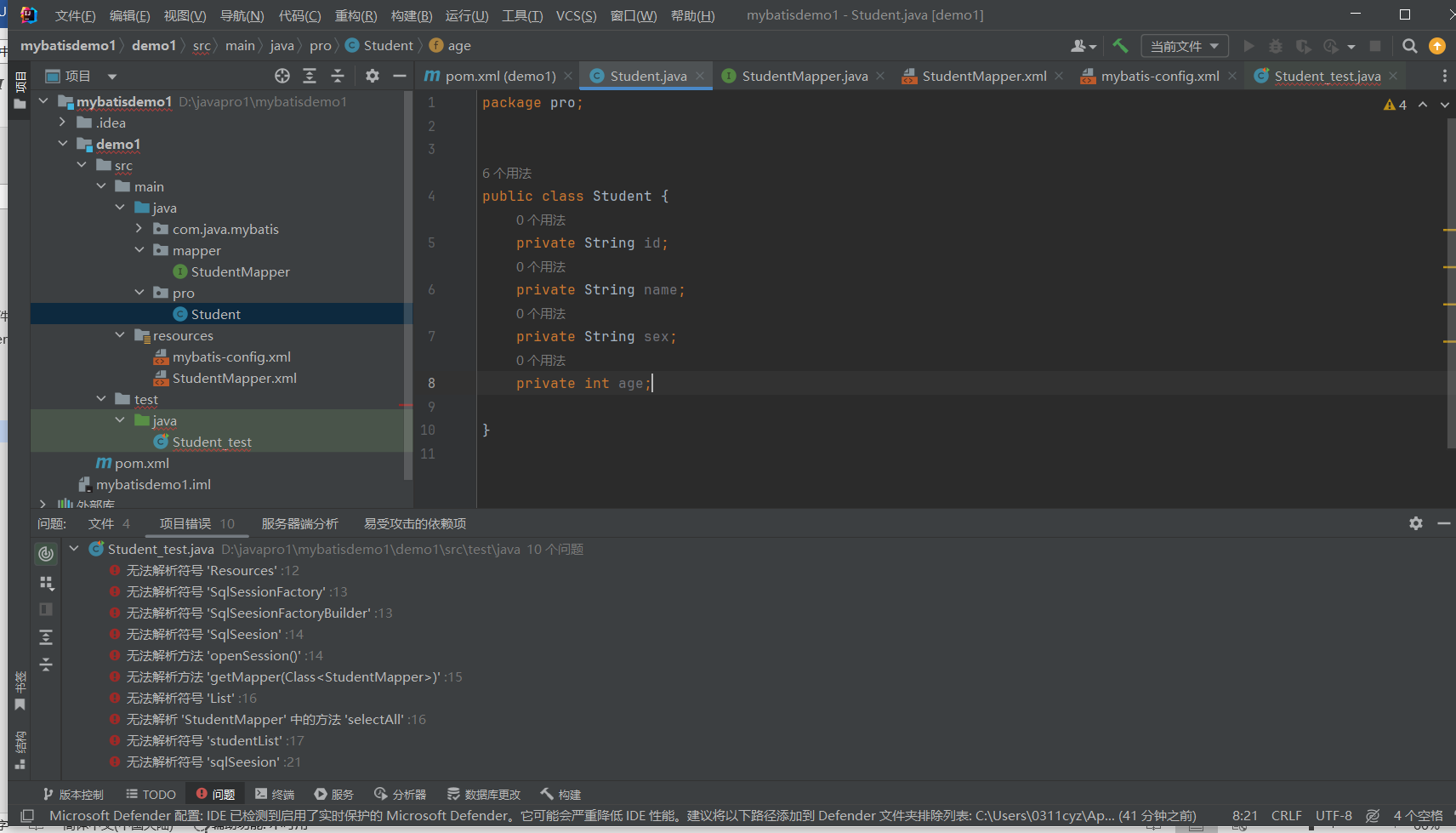
Idea自带数据库连接，驱动可从官网下载再导入



### 文件配置

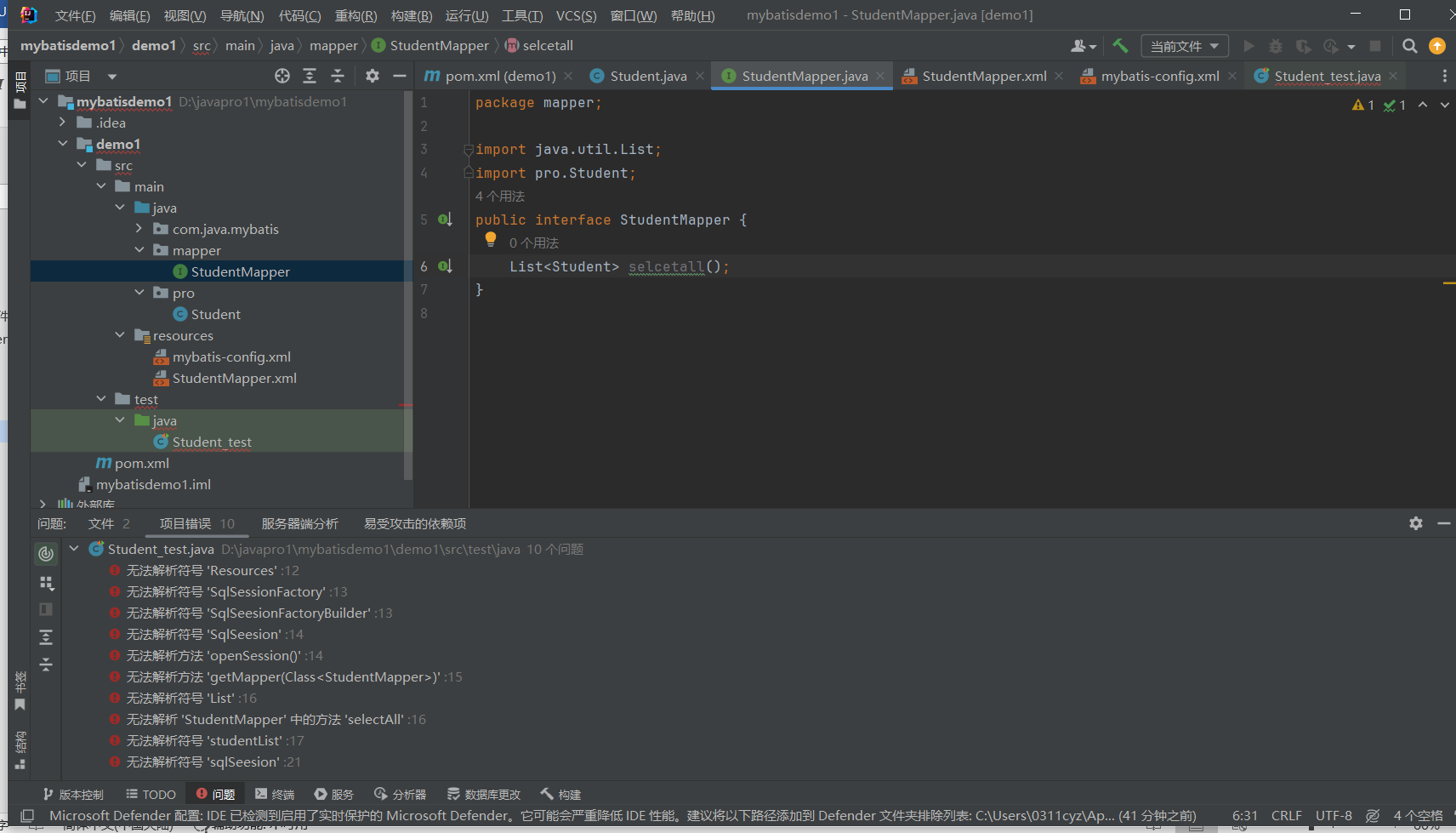


### 类



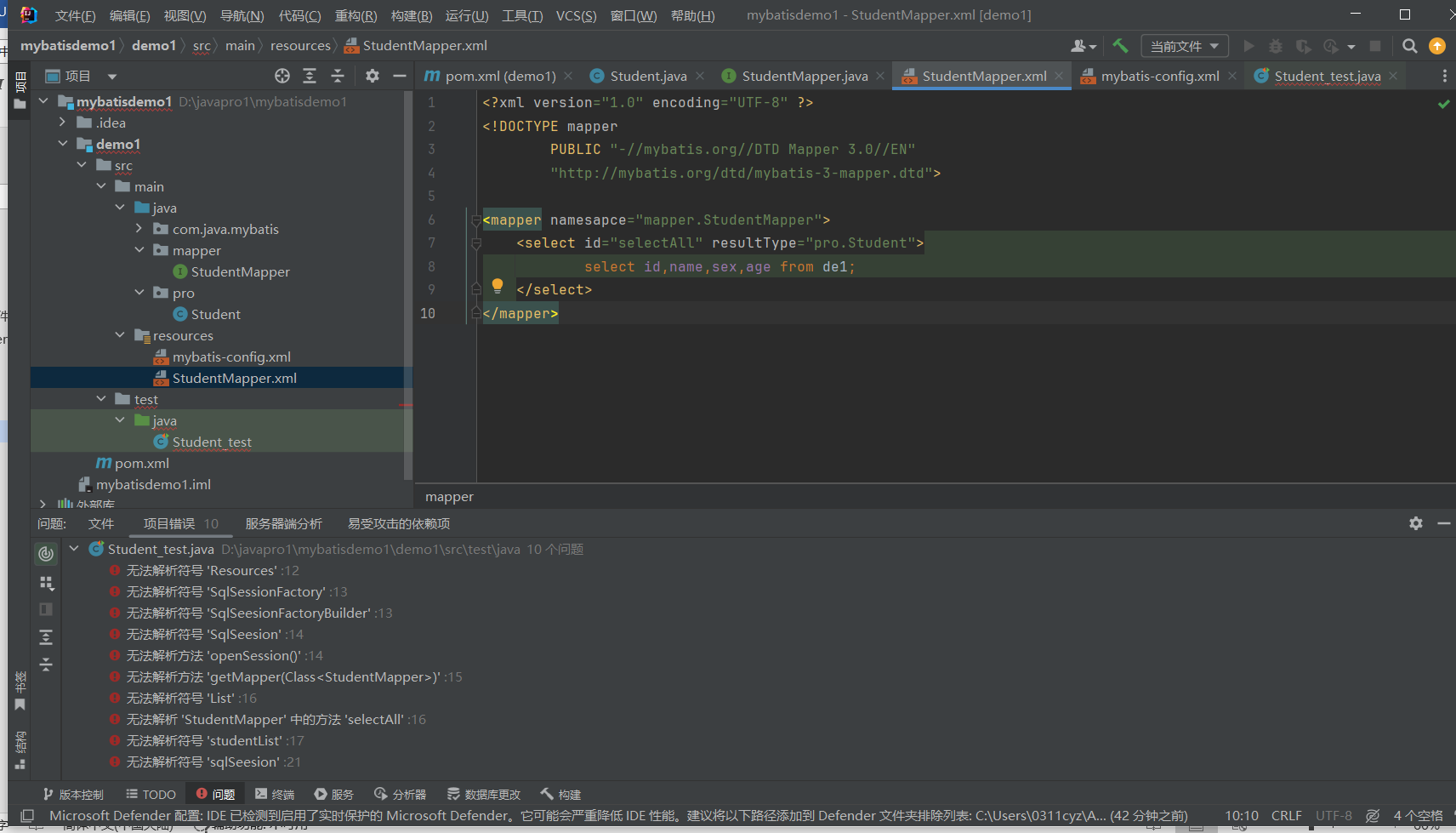
package pro;  
  
  
public class Student {  
 private String id;  
 private String name;  
 private String sex;  
 private int age;  
  
}

### 接口



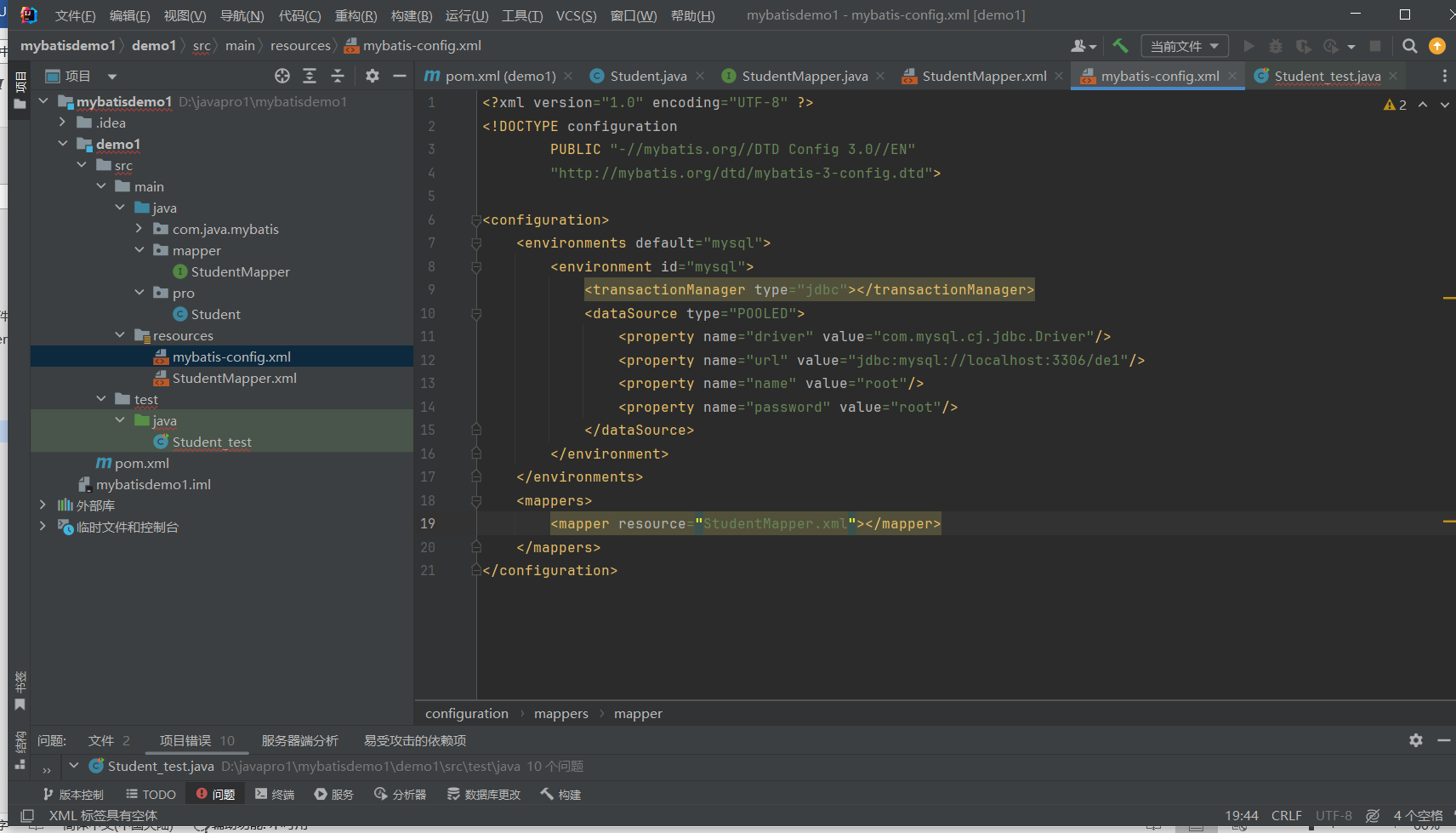
package mapper;  
  
import java.util.List;  
import pro.Student;  
public interface StudentMapper {  
 List<Student> selcetall();  
}

### 映射文件



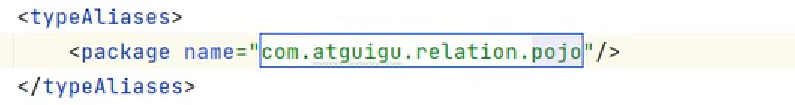
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper  
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
  
<mapper namesapce="mapper.StudentMapper">  
 <select id="selectAll" resultType="pro.Student">  
 select id,name,sex,age from de1;  
 </select>  
</mapper>

### 配置文件

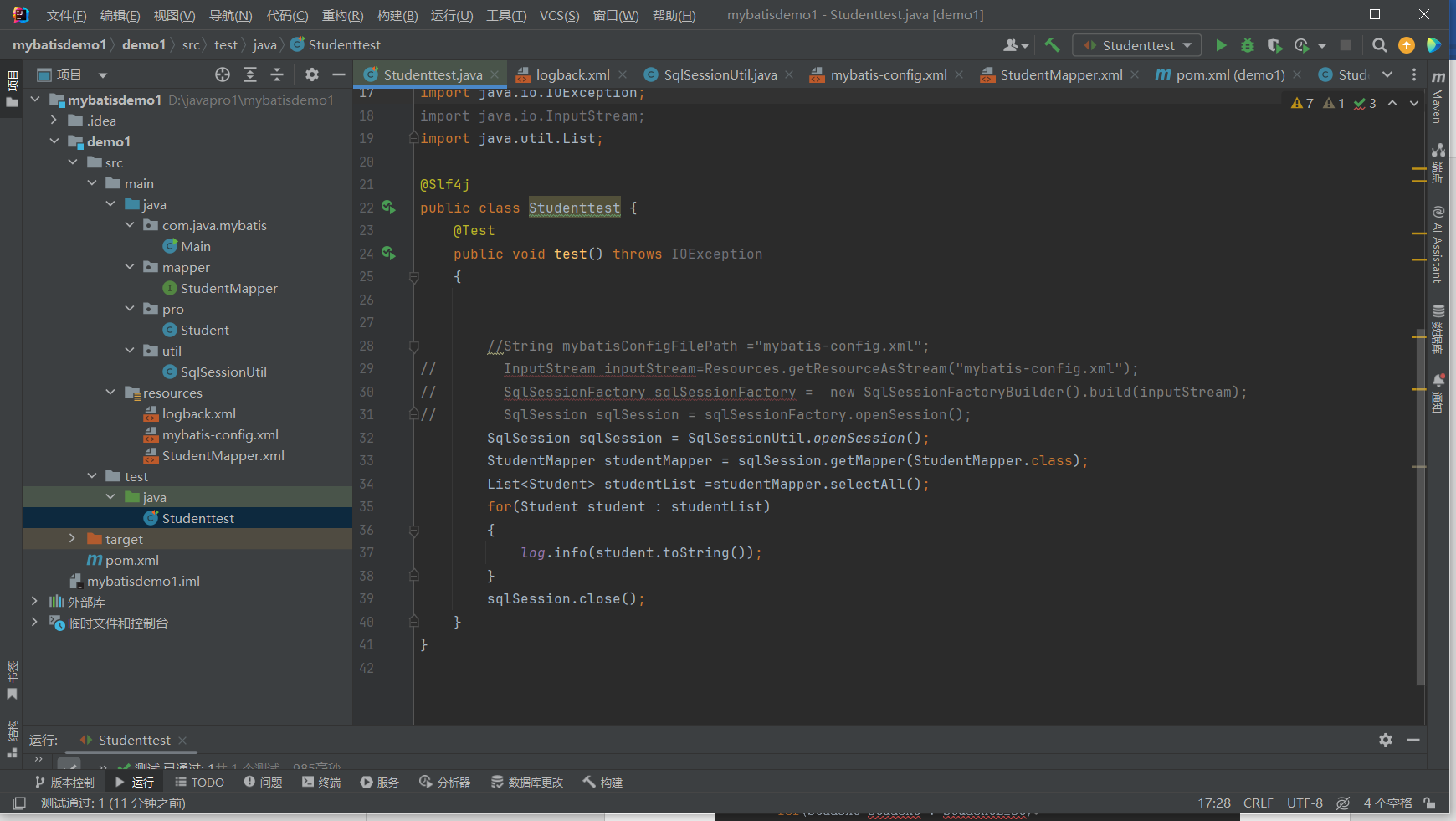


<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
  
<configuration>  
 <environments default="mysql">  
 <environment id="mysql">  
 <transactionManager type="jdbc"></transactionManager>  
 <dataSource type="POOLED">  
 <property name="driver" value="com.mysql.cj.jdbc.Driver"/>  
 <property name="url" value="jdbc:mysql://localhost:3306/demo"/>  
 <property name="name" value="root"/>  
 <property name="password" value="root"/>  
 </dataSource>  
 </environment>  
 </environments>  
 <mappers>  
 <mapper resource="StudentMapper.xml"></mapper>  
 </mappers>  
</configuration>

别名



### Text测试



import lombok.extern.slf4j.Slf4j;  
import mapper.StudentMapper;  
import org.apache.ibatis.session.SqlSession;  
import org.apache.ibatis.session.SqlSessionFactory;  
import org.apache.ibatis.session.SqlSessionFactoryBuilder;  
import org.junit.Test;  
import pro.Student;  
import org.apache.ibatis.io.Resources;  
import util.SqlSessionUtil;  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.List;  
  
@Slf4j  
public class Studenttest {  
 @Test  
 public void test() throws IOException  
 {  
  
  
 String mybatisConfigFilePath ="mybatis-config.xml";  
 InputStream inputStream=Resources.getResourceAsStream("mybatis-config.xml");  
// SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);  
// SqlSession sqlSession = sqlSessionFactory.openSession();  
 SqlSession sqlSession = SqlSessionUtil.*openSession*();  
 StudentMapper studentMapper = sqlSession.getMapper(StudentMapper.class);  
 List<Student> studentList =studentMapper.selectAll();  
 for(Student student : studentList)  
 {  
 *log*.info(student.toString());  
 }  
 sqlSession.close();  
 }  
}

1读取MyBatis的配置文件

2 基于MyBatis的配置文件构建SqlSessionFactory工厂

3.通过SqlSessionFactory获取SglSession(connection)

4.根据接口的类对象，获取接口的实现类对象(动态代理)

5.以接口的角度调用方法

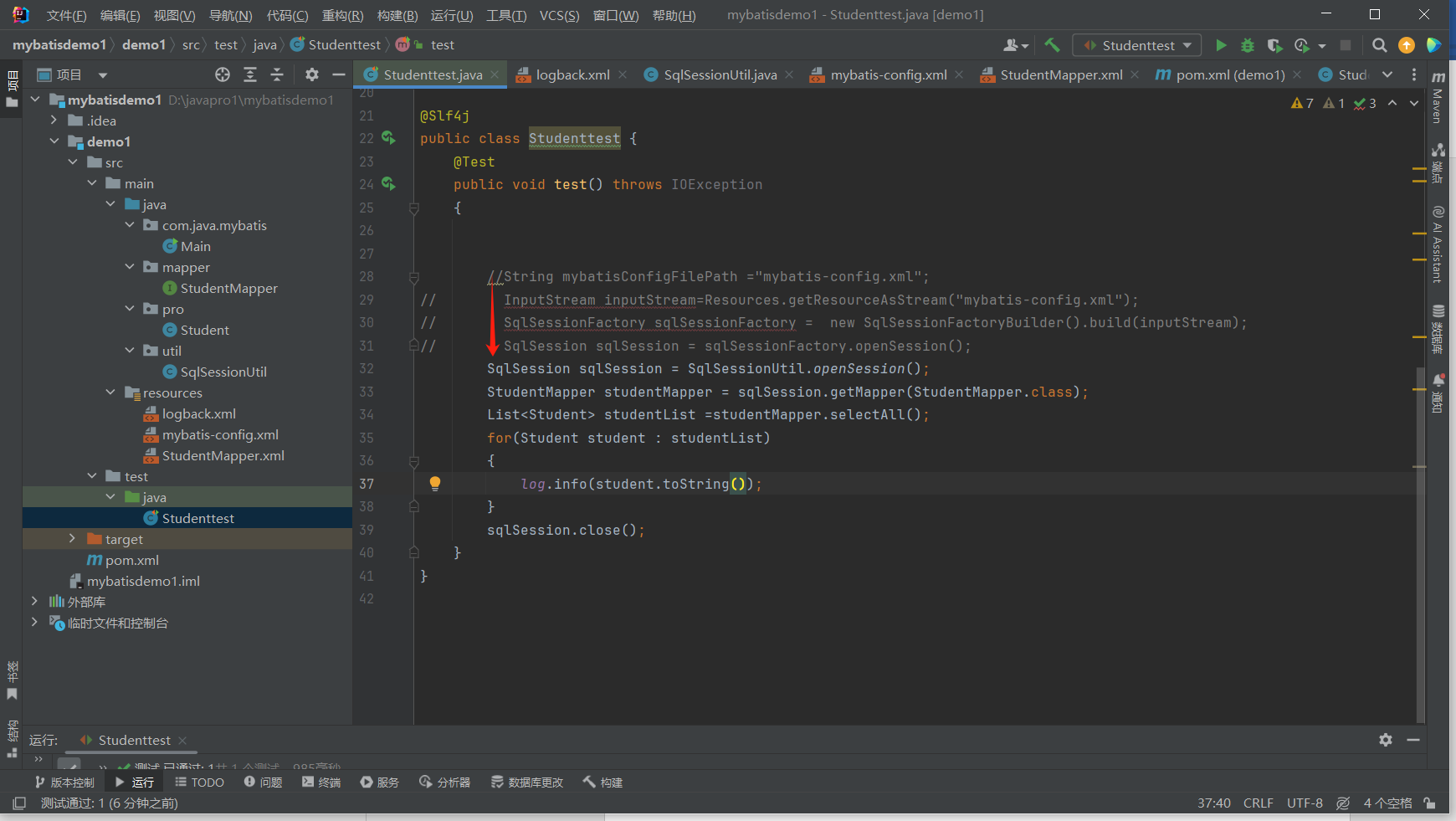
6.处理结果

7.释放资源

### 封装工具类

创建工具类

package util;  
  
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 java.io.IOException;  
import java.io.InputStream;  
  
public class SqlSessionUtil {  
 private static SqlSessionFactory *sqlSessionFactory* =null;  
 static {  
 try {  
 InputStream inputStream= null;  
 inputStream = Resources.*getResourceAsStream*("mybatis-config.xml");  
 *sqlSessionFactory* = new SqlSessionFactoryBuilder().build(inputStream);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 }  
 public static SqlSession openSession()  
 {  
 return *sqlSessionFactory*.openSession();  
 }  
}

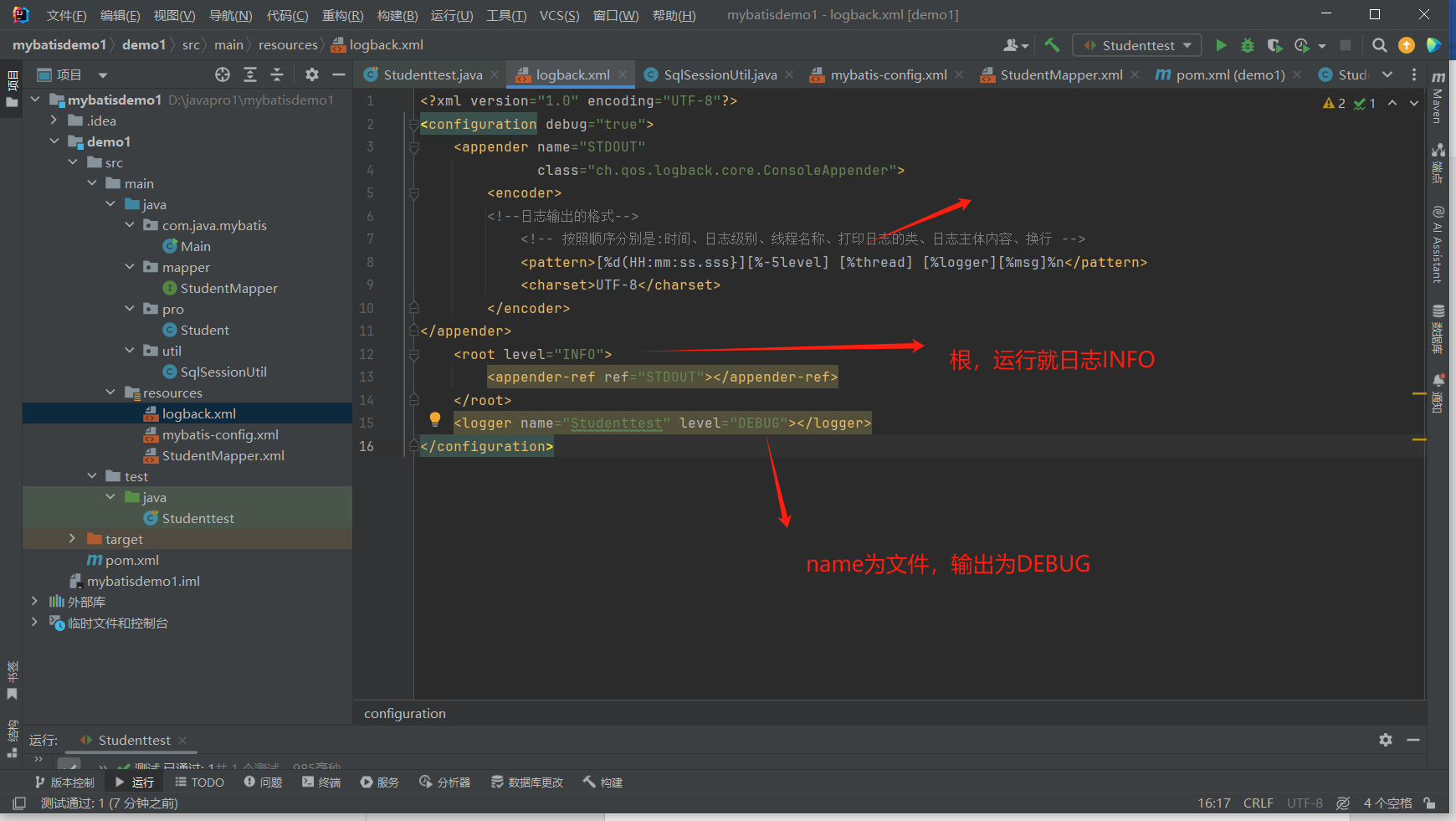


日志框架



日志配置文件

<?xml version="1.0" encoding="UTF-8"?>  
<configuration debug="true">  
 <appender name="STDOUT"  
 class="ch.qos.logback.core.ConsoleAppender">  
 <encoder>  
 <!--日志输出的格式-->  
 <!-- 按照顺序分别是:时间、日志级别、线程名称、打印日志的类、日志主体内容、换行 -->  
 <pattern>[%d(HH:mm:ss.sss}][%-5level] [%thread] [%logger][%msg]%n</pattern>  
 <charset>UTF-8</charset>  
 </encoder>  
</appender>  
 <root level="INFO">  
 <appender-ref ref="STDOUT"></appender-ref>  
 </root>  
 <logger name="Studenttest" level="DEBUG"></logger>  
</configuration>



### Setting

Loglmpl:<setting name=”logImpl” value=”SLF4J”/>//日志

MapunderscoretoCase:<setting name=” MapunderscoretoCase” value=”true”/>//驼峰映射

### Mybatis输入

#{}会在值打上引号

${}直接代入值

Map

Mapper

Integer insertdata(Map<String, Object>condition);

映射文件

<insert id="insertdata">  
 insert into de1(id,name,sex,age) VALUES(#{inid},#{inname},#{insex},#{inage})  
</insert>

Test

Map<String, Object> condition =new HashMap<>();  
condition.put("inid","rr");  
condition.put("inname","cyz");  
condition.put("insex","n");  
condition.put("inage",123);  
Integer showdata=dataMapper.insertdata(condition);  
*log*.info(showdata.toString());

实体类型



单参

Mapper

Data selectbyid(String dataid);

映射文件

<select id="selectbyid" resultType="data.Data">  
 select id,name,sex,age from de1 where id =#{dataid}  
</select>

Test

Data data=dataMapper.selectbyid("abc");  
*log*.info(data.toString());

多参

有注解@Param

Mapper

Integer updatedata(@Param("upid") String id, @Param("upname") String name);

映射文件

<update id="updatedata">  
 update de1 set name=#{upname} where id=#{upid}  
</update>

无注解

可在前面打上@Param注解，若无，则再映射文件需要用Param1，2，3代替其中的upname

Mapper

Integer updatedata(String id,String name);

映射文件

<update id="updatedata">  
 update de1 set name=#{param2} where id=#{param1}  
</update>

Test

Integer result=dataMapper.updatedata("rr","d");  
*log*.info(result.toString());

### Mybatis数据输出

单行单列

Mapper

Integer selecttotal();//单行单列

映射文件

<select id="selecttotal" resultType="int">  
 select count(\*) from de1  
</select>

Test

Integer result= dataMapper.selecttotal();  
*log*.info(result.toString());

多行单列

多行多列

单行多列（对象）

Mapper

Data selectbyname(String namekk);//单行多列

映射文件

<select id="selectbyname" resultType="data.Data">  
 select id id\_t,name name\_t,sex id\_t,age age\_t from de1 where name=#{namekk}  
</select>

Test

Data data=dataMapper.selectbyname("hei");  
*log*.info(data.toString());

单行多列（列元素不可放在同一个对象，map）

Mapper

Map<String,Object> selectMap();

映射文件

<select id="selectMap" resultType="map">  
 select max(age),min(age),avg(age) from de1  
</select>

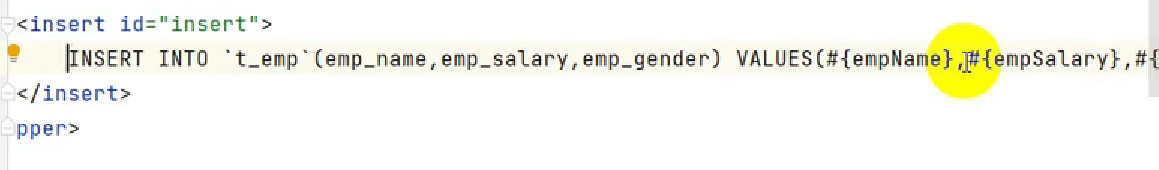
Test

Map<String,Object> datamap=dataMapper.selectMap();  
Set<Map.Entry<String, Object>> entries = datamap.entrySet();  
for (Map.Entry<String, Object> entry : entries) {  
 *log*.info(entry.toString());  
}

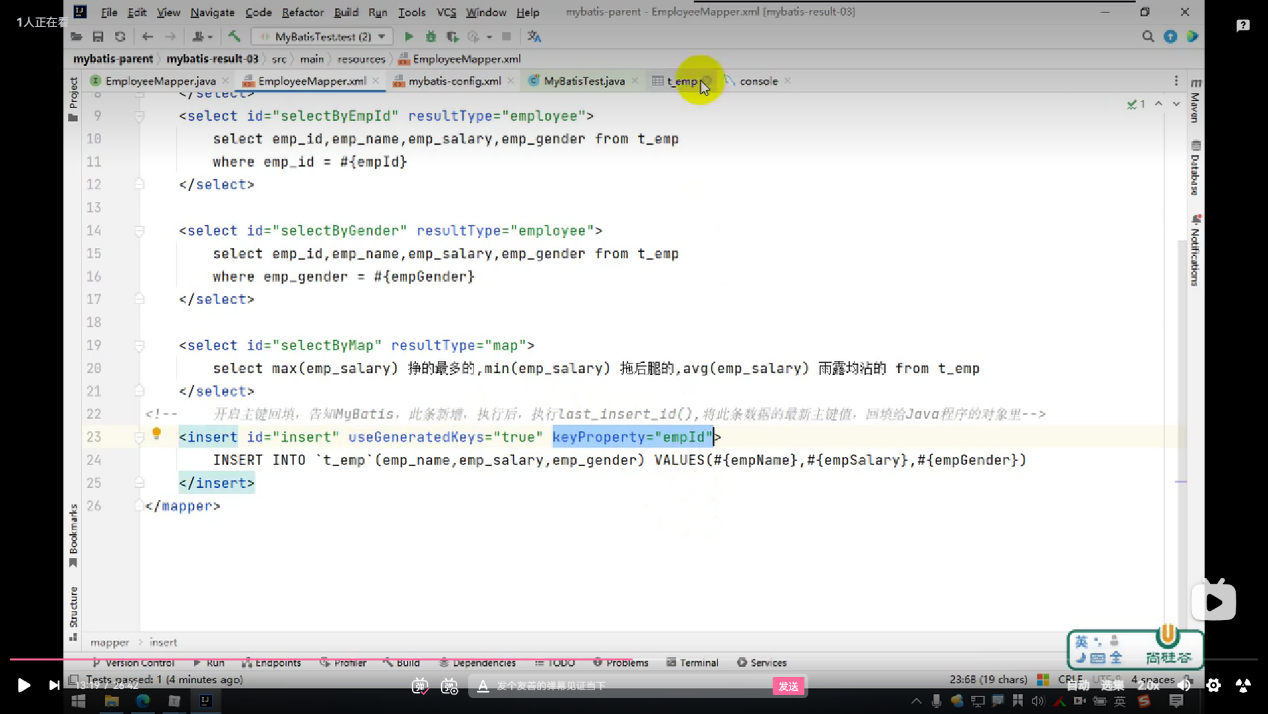
主键回传（自增，非自增）

自增

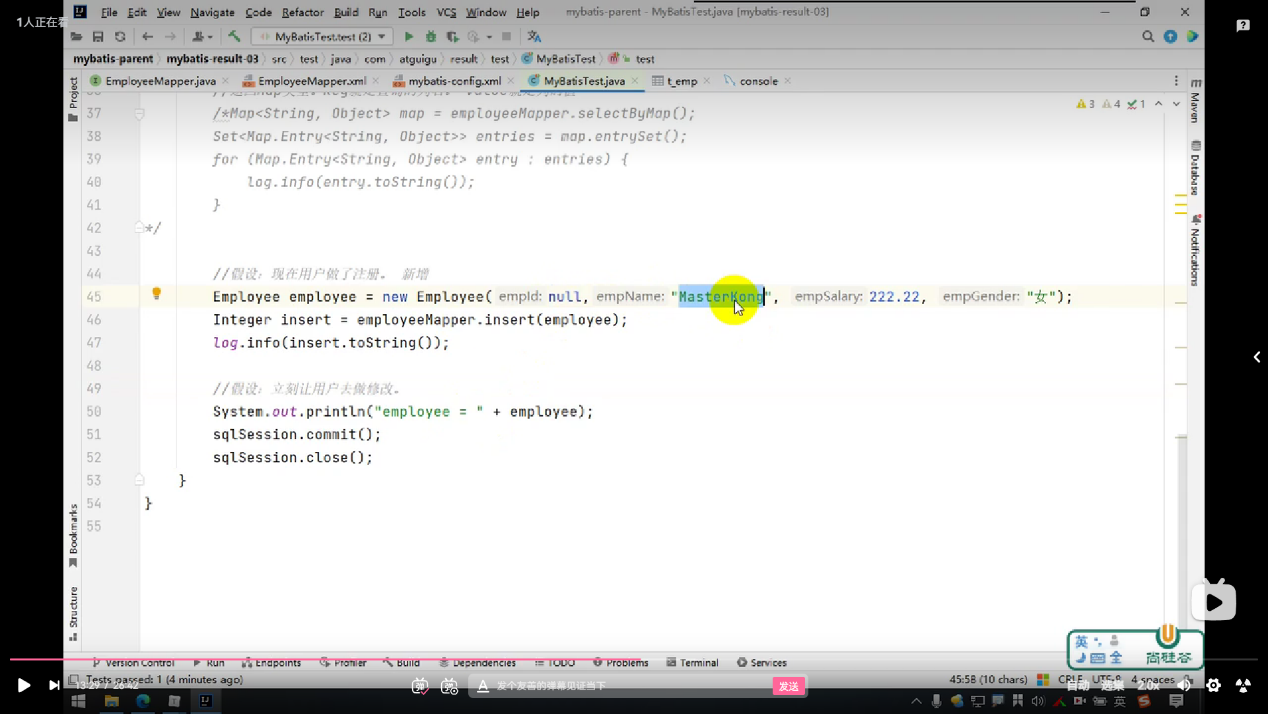
Mapper



映射文件

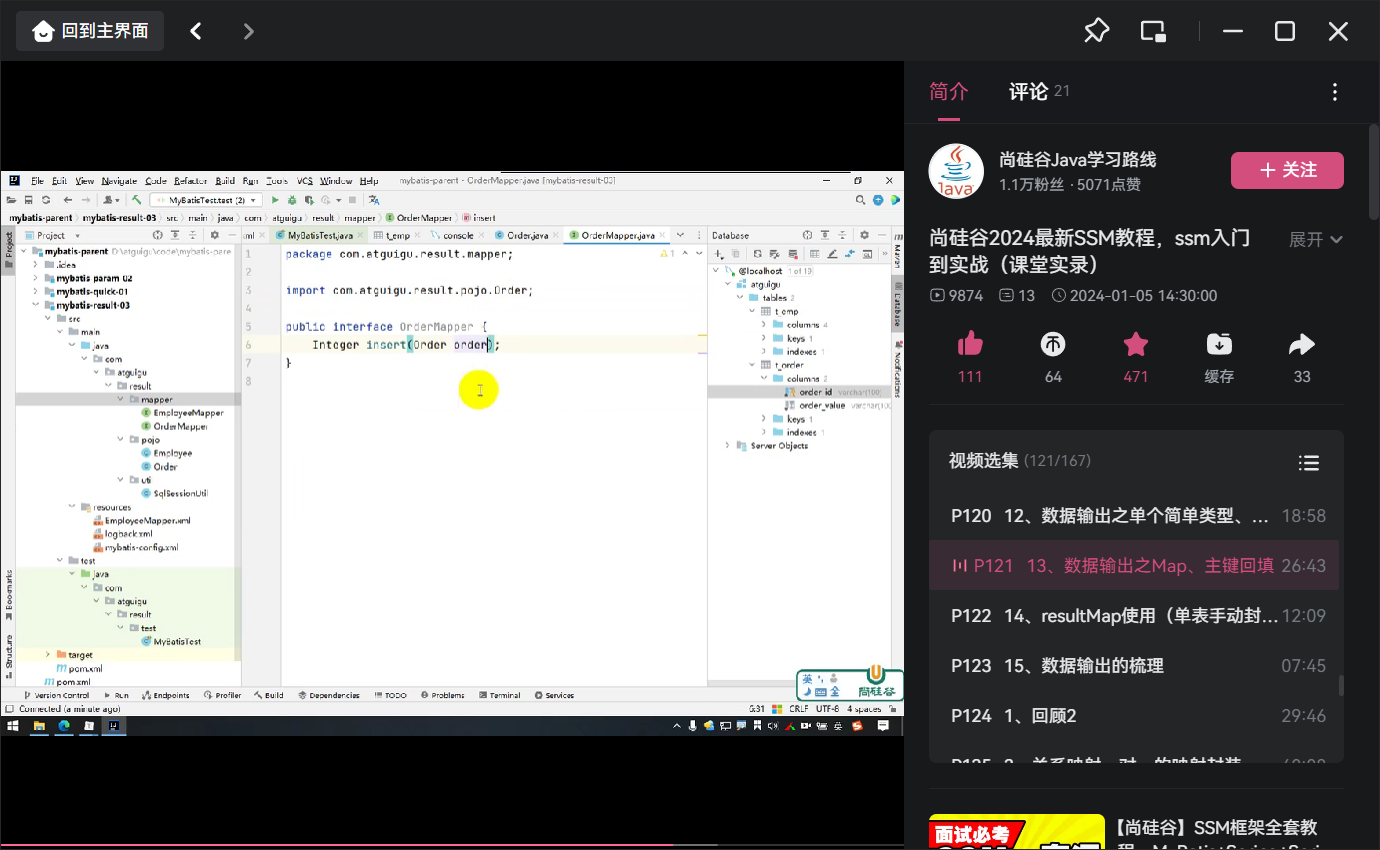


Test

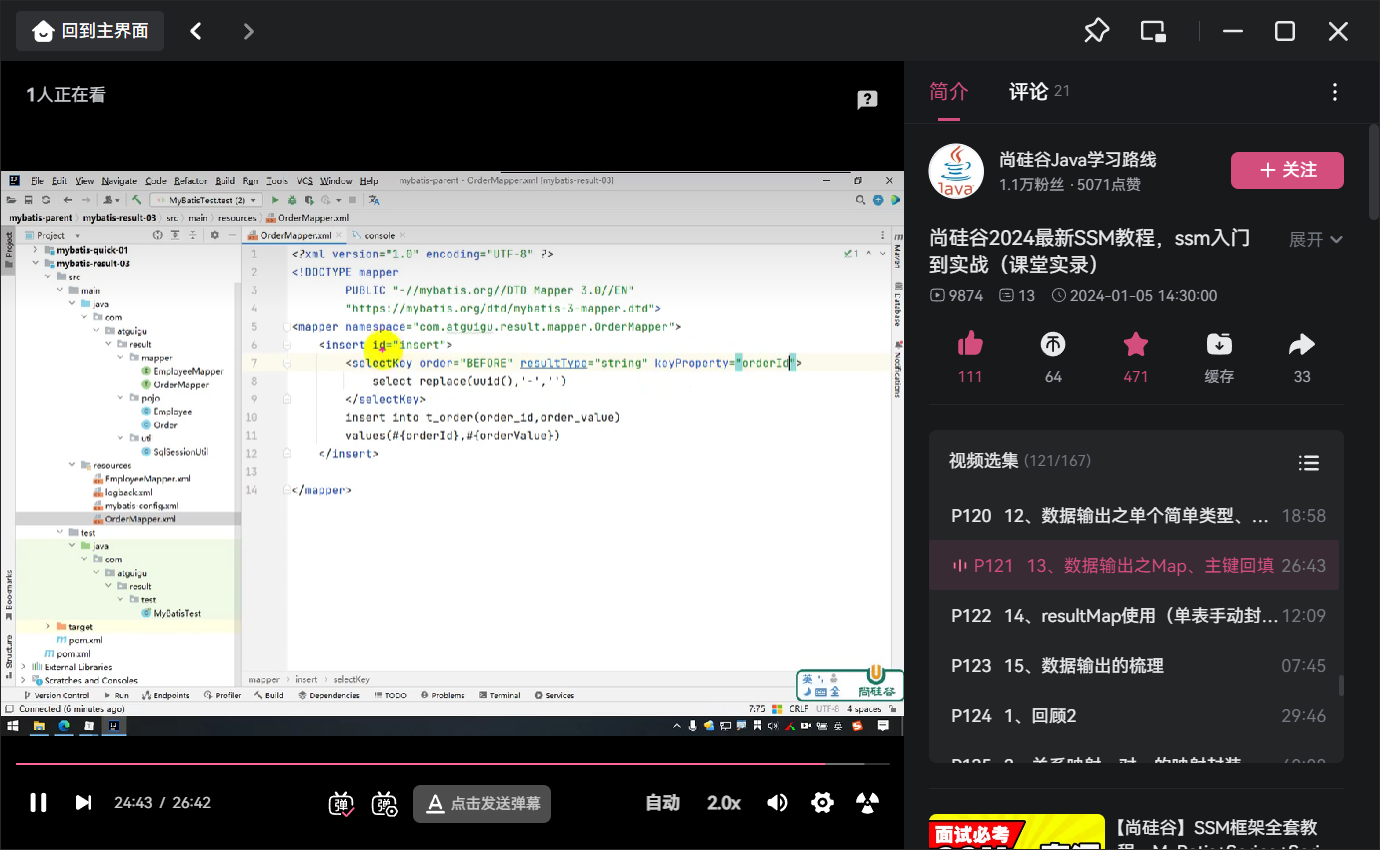


非自增

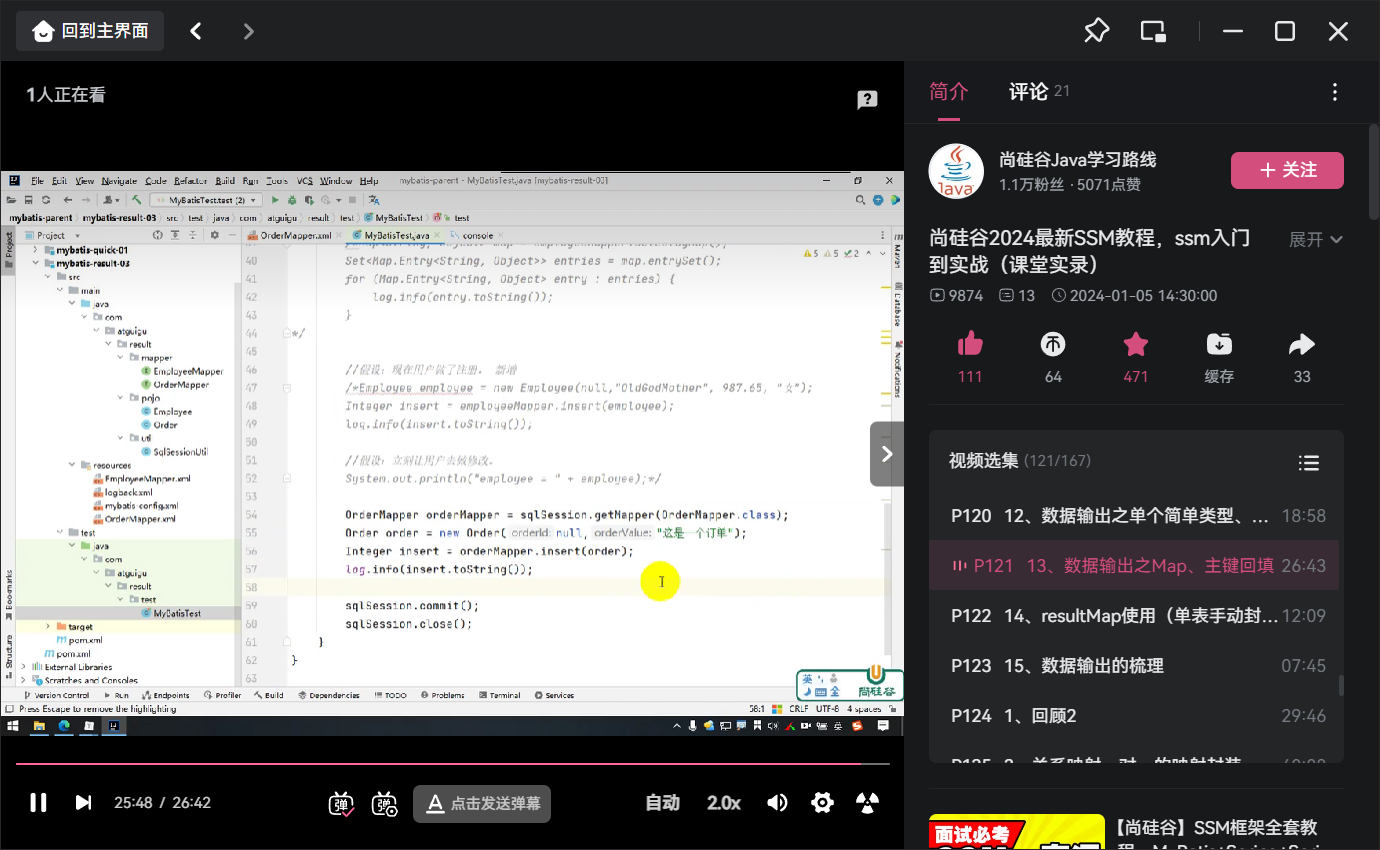
Mapper



映射文件



Test



Resultmap（查询时不用别名，不用驼峰映射）

<resultMap id="datamap" type="data.Data">  
 <id column="id" property="id\_t"></id>  
 <result column="name" property="name\_t"></result>  
 <result column="sex" property="sex\_t"></result>  
 <result column="age" property="age\_t"></result>  
</resultMap>  
<select id="selectbyname" resultMap="datamap">  
 select id,name,sex,age from de1 where name=#{namekk}  
</select>

### 多表映射

对一

映射文件

<resultMap id="ordermap" type="Order">  
 <id column="orderid" property="orderid\_t"></id>  
 <result column="ordername" property="ordername\_t"></result>  
 <association property="customer" javaType="Customer">  
 <id column="customerid" property="customerid\_t"></id>  
 <result column="customername" property="customername\_t"></result>  
 </association>  
</resultMap>  
<select id="selectOrder" resultMap="ordermap">  
 select orderid,ordername,demo3.customerid,customername from demo2  
 inner join demo3 on demo2.customerid=demo3.customerid  
 where orderid=#{orderidk}  
</select>

Mapper

Order selectOrder(Integer orderidk);

Text

OrderMapper orderMapper = sqlSession.getMapper(OrderMapper.class);  
Order order = orderMapper.selectOrder(6);  
*log*.info(order.toString());

对多

Mapper

<mapper namespace="mapper.CustomerMapper" >  
 <resultMap id="customermap" type="Customer">  
 <id column="customerid" property="customerid\_t"></id>  
 <result column="customername" property="customername\_t"></result>  
 <collection property="orderList" ofType="data.Order">  
 <id column="orderid" property="orderid\_t"></id>  
 <result column="ordername" property="ordername\_t"></result>  
 </collection>  
 </resultMap>  
 <select id="selectCustomer" resultMap="customermap">  
 select demo2.customerid,customername,orderid,ordername from demo2  
 inner join demo3 on demo2.customerid=demo3.customerid  
 where demo2.customerid=#{customeridm}  
 </select>  
</mapper>

接口

public interface CustomerMapper {  
 Customer selectCustomer(Integer customeridm);  
}

test

Customer customer = customermapper.selectCustomer(5);  
*log*.info(customer.toString());  
sqlSession.commit();  
sqlSession.close();

多对多

## Spring boot

## Vue

## Javascript