Maven项目中使用MyBatis

在maven项目中使用mybatis实现查询数据库库中的一张表
步骤：
1、在对应目录中创建一个空工程：idea-maven-mybatis
2、添加新的maven模块：a-maven-mybatis01，保存在空工程的目录下
3、在pom文件中添加mybatis的依赖坐标和mysql的依赖坐标

```
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.2</version>
</dependency>

<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.49</version>
</dependency>
```

4、创建实体类com.studymyself.entity.User--作用是mybatis用来创建对象保存查询结果集中表中的一行数据
5、创建持久层的dao接口com.studymyself.dao.UserDao其中定义操作数据库的方法--mybatis是在持久层起作用的
6、在dao包下创建sql映射文件--一般一个表对应一个映射文件,文件名和接口名一样
由于该sql映射文件不在resources目录中，所以要在pom文件中配置资源位置的插件信息，使该文件编译时拷贝到target
对应目录中

```
<build>
 <resources>
   <resource>
     <!--表示编译时在src/main/java目录下的-->
     <directory>src/main/java</directory>
     <!--所有.properties、.xml类型的文件可以被扫描到-->
     <includes>
       <include>**/*.properties</include>
       <include>**/*.xml</include>
     </includes>
     <!--filtering的值false表示不启用过滤器，上面include已经是过滤操作了-->
     <filtering>false</filtering>
   </resource>
 </resources>
</build>
```

7、在resources目录中创建mybatis核心目录文件
8、在resources目录下创建保存数据库连接信息的属性文件--jdbc.properties
9、创建使用的mybatis类，通过mybatis访问数据库的代码。--在studymyself包下创建MyApp.java

出现的一些错误：

--首先我们要知道的是什么叫项目的类的根路径：

maven项目中，点击右边的maven会出现maven窗口，双击其中的compile，开始项目的编译，而后生成一个target文件目录。
打开target里面是一个classes目录，存放的是编译后的class文件和resources目录中的资源配置文件。
当我们不是maven项目时，直接运行项目，也是先编译后运行，也会生成target目录，和上面一样。
所以运行时JVM是到classes目录中找class文件和配置资源文件来运行的，这样就可以理解为target\classes目录就是项目类的根路径，对应的是项目中的src目录，在maven项目中src目录和src\main是相当于一样一个src目录，所以maven项目中resources目录中的文件被拷贝到类的根路径下，而java文件的class文件存放在原先一样的层次包中，只不过是根目录换了个名字而已，项目的目录结构还是没变，resources中被提取出来了。
就是说：
maven项目中存放在resources目录中的资源配置文件，编译后存到的是类根路径下，所以上面第八步中代码的获取mybatis的核心配置文件时该文件的路径只需填文件名加文件类型就可以了。
而sql映射文件通过资源插件的配置拷贝到的是target/classes/com/studymyself/dao目录中，mybatis是从根路径中扫描文件，所以在mybatis核心配置文件中配置sql映射文件的位置信息就是这样：com/studymyself/dao/UserDao.xml
--注意是左斜杠，不是点，还要保证resources目录一定要是资源文件夹
有时候我们的项目配置都没有问题，但生成的target目录中不一定有相关的文件，
我们就得clean，然后再编译。
或者直接Rebuild Project。
或者手工复制资源配置文件到target对应目录中。
或者File->Invalidata Caches/Restart，表示删除idea之前项目的缓存，重启idea

在上面的基础上添加添加数据的方法测试

首先在UserDao添加添加数据的方法

```
package com.studymyself.dao;

import com.studymyself.entity.User;

import java.util.List;

//接口操作t_user表
public interface UserDao {

    //该方法查询t_user表中所有数据
    public List<User> getAll();

    //添加数据的方法，要和映射文件中sql语句的id一致，namespace的值也要是本接口的全限定名
    //才能将方法和sql映射文件中的某个id的sql语句绑定，而方法中传递的参数就和sql语句的parameterType属性
    //的作用一样
    public int addUser(User user);

}
```

然后添加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">
        <!--命名空间属性：里面的值是dao接口的全限定名，虽然可以自定义字符串，但不能这样用-->
        <!--作用是区分有多个sql映射文件时，下面sql语句的唯一标识id可能重复，
```

```
            这时就用接口全限定名.id 来区别执行哪个映射文件的同id的sql语句了
            com.studymyself.dao.UserDao.getAll
        -->
<mapper namespace="com.studymyself.dao.UserDao">

    <!--id中的值是这条sql语句在本映射文件中的唯一标识
        而com.studymyself.dao.UserDao.getAll是这条sql语句在本项目中的唯一标识
    -->
    <select id="getAll" resultType="com.studymyself.entity.User">
        select
            id,loginName,loginPwd,realName
        from
            t_user
    </select>

    <!--下面需要传参数，但是我们并没有设置parameterType属性却仍然可以执行添加数据的操作
        原因是namespace属性绑定的UserDao接口中添加数据的方法addUser需要的参数是User对象
        本句sql语句的id和接口中的方法名一致，所以下面占位符中传的值就是接口方法中传递
        的User对象中的值，也就不在多余设置parameterType属性为User类了，但是还是写上更好
    -->
    <insert id="addUser">
        insert into t_user
            (loginName,loginPwd,realName)
        values
            (#{loginName},#{loginPwd},#{realName})
    </insert>

</mapper>
```

核心配置文件添加一个全局配置，添加输出控制台日志的配置

```
<?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>

    <!--注意这些属性设置从上往下是有顺序的，写反就会报错-->

    <!--配置属性资源文件-->
    <properties resource="jdbc.properties"/>

    <!--该属性的作用：控制mybatis全局行为，
        顺序要在上面的properties属性下面-->
    <settings>
        <!--设置mybatis输出日志-->
        <setting name="logImpl" value="STDOUT_LOGGING"/>
    </settings>

    <!--environments是环境配置，是复数说明下面可以配置多个环境
        所以default属性的值表示这个配置文件用下面的哪一套数据库配置信息
        这里使用development1
    -->
    <environments default="development1">
```

```
        <!--第一套数据库连接信息配置，唯一标识id叫development-->
        <environment id="development1">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="${jdbc.driver}"/>
                <property name="url" value="${jdbc.url}"/>
                <property name="username" value="${jdbc.username}"/>
                <property name="password" value="${jdbc.password}"/>
            </dataSource>
        </environment>
        <!--第二套数据库连接信息配置，唯一标识id叫development-->
        <environment id="development2">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="${jdbc.driver}"/>
                <property name="url" value="${jdbc.url}"/>
                <property name="username" value="${jdbc.username}"/>
                <property name="password" value="${jdbc.password}"/>
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <!--这里也可以添加多个sql映射文件，因为项目中不止一个表
            这就是映射文件中namespace属性的作用了-->
        <!-- <mapper resource="StudentDao.xml"/>-->
        <mapper resource="com/studymyself/dao/UserDao.xml"/>
    </mappers>
</configuration>
```

数据库信息的属性配置文件如下

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/mysqlstudy
jdbc.username=root
jdbc.password=rong195302
```

pom文件

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.studymyself</groupId>
  <artifactId>a-maven-mybatis01</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
```

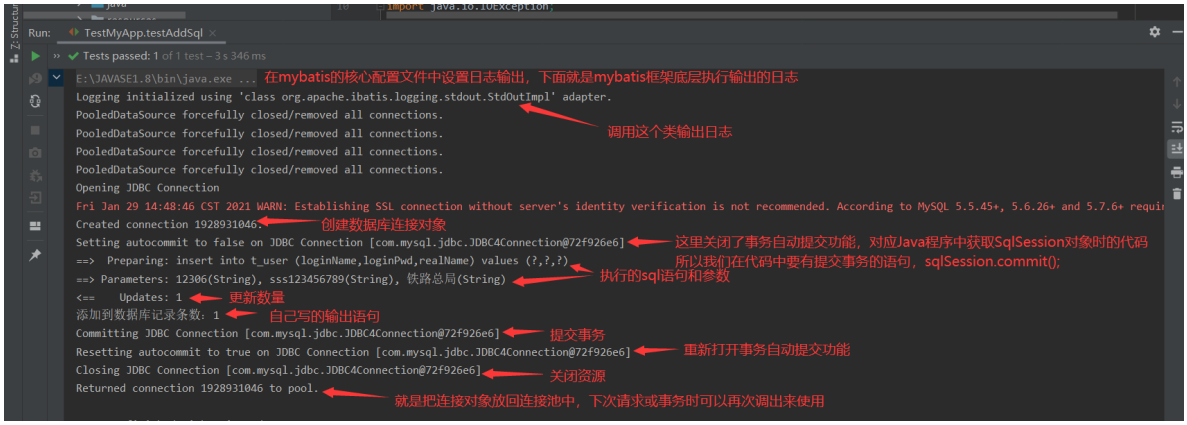
```
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
      <version>3.5.2</version>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>5.1.47</version>
    </dependency>
  </dependencies>

  <build>
    <resources>
      <resource>
        <!--表示编译时在src/main/java目录下的-->
        <directory>src/main/java</directory>
        <!--所有.properties、.xml类型的文件可以被扫描到-->
        <includes>
          <include>**/*.properties</include>
          <include>**/*.xml</include>
        </includes>
        <!--filtering的值false表示不启用过滤器，上面include已经是过滤操作了-->
        <filtering>false</filtering>
      </resource>
    </resources>
  </build>
</project>
```

在test目录中新建mabatis测试类com.studymyself.TestMyApp

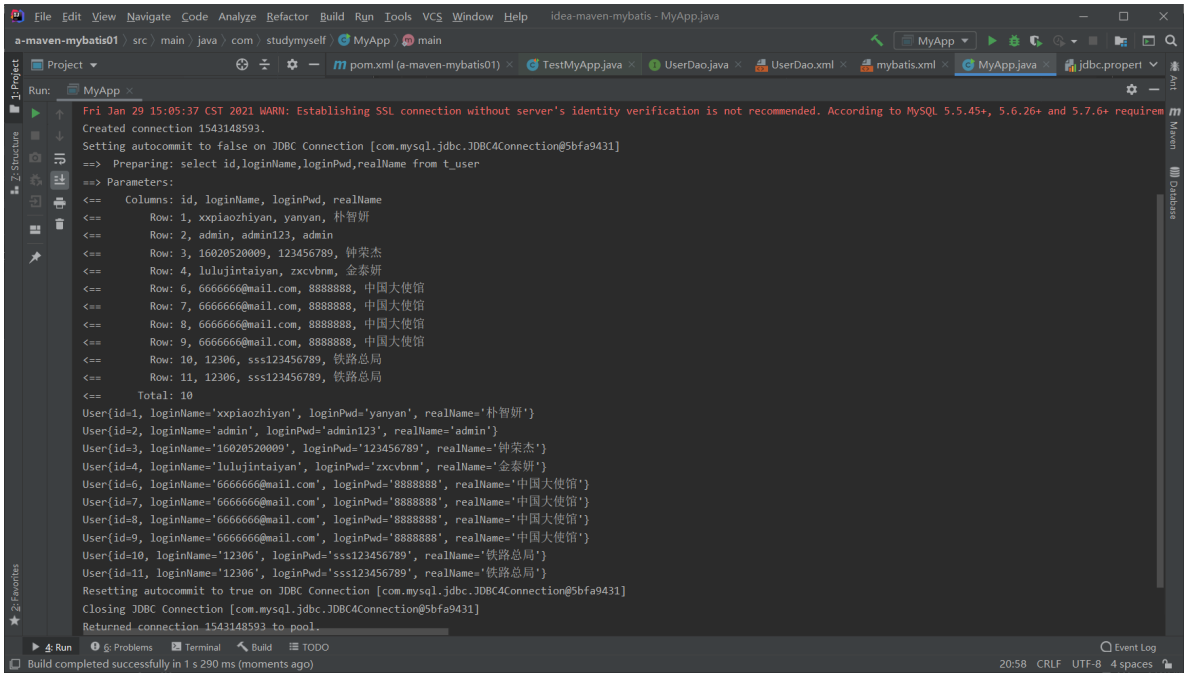
```
package com.studymyself;

import com.studymyself.entity.User;
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 java.io.IOException;

public class TestMyApp {

    @Test
    public void testAddSql(){

        SqlSession sqlSession = null;
```

```
        try {
            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(Resources.getResourceAsStream("mybatis.xml"));
            sqlSession = sqlSessionFactory.openSession();

            //将UserDao.xml文件中namespace的值放进一个变量中
            String userId = "com.studymyself.dao.UserDao";

            //插入数据
            User user1 = new User();
            user1.setLoginName("12306");
            user1.setLoginPwd("sss123456789");
            user1.setRealName("铁路总局");
            int count = sqlSession.insert(userId+".addUser",user1);
            System.out.println("添加到数据库记录条数: "+count);

            sqlSession.commit();
        } catch (IOException e) {
            if (sqlSession != null) {
                sqlSession.rollback();
            }
            e.printStackTrace();
        }finally {
            if (sqlSession != null) {
                sqlSession.close();
            }
        }

    }

}
```

执行测试方法，控制台输出的日志和结果信息如下：



执行查询的输出信息如下


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
idea-maven-mybatis - MyApp.java
a-maven-mybatis01 src main java com studymyself MyApp main
MyApp
Project
pom.xml (a-maven-mybatis01)
TestMyApp.java
UserDao.java
UserDao.xml
mybatis.xml
MyApp.java
jdbc.propert
Run: MyApp
Fri Jan 29 15:05:37 CST 2021 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirem
Created connection 1543148593.
Setting autocommit to false on JDBC Connection [com.mysql.jdbc.JDBC4Connection@5bfa9431]
==> Preparing: select id,loginName,loginPwd,realName from t_user
==> Parameters:
<== Columns: id, loginName, loginPwd, realName
<== Row: 1, xxpiaozhiyan, yanyan, 朴智妍
<== Row: 2, admin, admin123, admin
<== Row: 3, 16020520009, 123456789, 钟荣杰
<== Row: 4, lulujintaiyan, zxcvbnm, 金泰妍
<== Row: 6, 6666666@mail.com, 8888888, 中国大使馆
<== Row: 7, 6666666@mail.com, 8888888, 中国大使馆
<== Row: 8, 6666666@mail.com, 8888888, 中国大使馆
<== Row: 9, 6666666@mail.com, 8888888, 中国大使馆
<== Row: 10, 12306, sss123456789, 铁路总局
<== Row: 11, 12306, sss123456789, 铁路总局
<== Total: 10
User{id=1, loginName='xxpiaozhiyan', loginPwd='yanyan', realName='朴智妍'}
User{id=2, loginName='admin', loginPwd='admin123', realName='admin'}
User{id=3, loginName='16020520009', loginPwd='123456789', realName='钟荣杰'}
User{id=4, loginName='lulujintaiyan', loginPwd='zxcvbnm', realName='金泰妍'}
User{id=6, loginName='6666666@mail.com', loginPwd='8888888', realName='中国大使馆'}
User{id=7, loginName='6666666@mail.com', loginPwd='8888888', realName='中国大使馆'}
User{id=8, loginName='6666666@mail.com', loginPwd='8888888', realName='中国大使馆'}
User{id=9, loginName='6666666@mail.com', loginPwd='8888888', realName='中国大使馆'}
User{id=10, loginName='12306', loginPwd='sss123456789', realName='铁路总局'}
User{id=11, loginName='12306', loginPwd='sss123456789', realName='铁路总局'}
Resetting autocommit to true on JDBC Connection [com.mysql.jdbc.JDBC4Connection@5bfa9431]
Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@5bfa9431]
Returned connection 1543148593 to pool.
4: Run
6: Problems
Terminal
Build
TODO
Event Log
Build completed successfully in 1 s 290 ms (moments ago)
20:58 CRLF UTF-8 4 spaces