2019

Java EE框架 ---MyBatis

Java EE framework --MyBatis

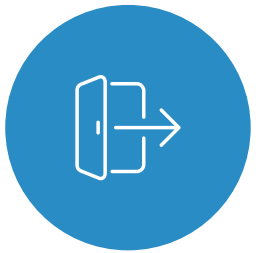
王磊

计算机工程学院

CONTENTS

创建javabean

编写Mybatis、user.xml



创建MybatisUtil工具类

Mapper动态代理方式

一、创建Java项目

1、新建Java Project mybatisCRUD

2、引入Mybatis jar包，构建路径

3、编写javabean

4、编写Mybatis核心配置文件sqlMapConfig.xml

5、创建log4j日志

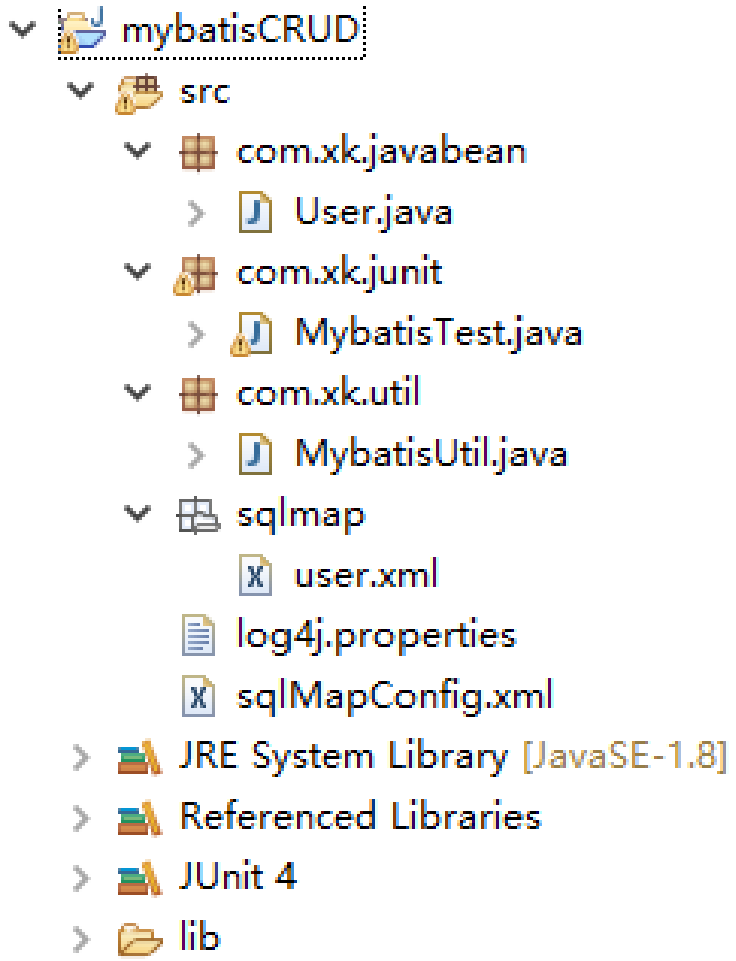
6、创建user.xml

7、在sqlMapConfig.xml中注册user.xml

8、创建MybatisUtil.java工具类

9、创建junit测试单元

10、逐个进行单元测试



二、创建javabean

创建javabean包，编写javabean

```
package com.xk.javabean;
public class User {
    private String id;
    private String username;
    private String password;
    private String grade;
    private String email;
    public String getId() {
    public void setId(String id) {
    public String getUsername() {
    public void setUsername(String username) {
    public String getPassword() {
    public void setPassword(String password) {
    public String getGrade() {
    public void setGrade(String grade) {
    public String getEmail() {
    public void setEmail(String email) {
    @Override
    public String toString() {
        return "User [id=" + id + ", username=" + username + ", password="
                + password + ", grade=" + grade + ", email="+ email + "]";
    }
```

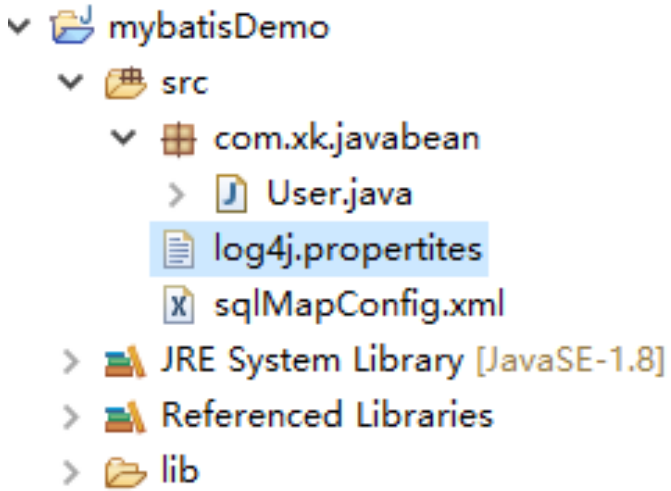
三、编写Mybatis核心配置文件

编写Mybatis核心配置文件sqlMapConfig.xml

```xml
<?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">
            <!-- 使用jdbc事务管理 -->
            <transactionManager type="JDBC" />
            <!-- 数据库连接池 -->
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver" />
                <property name="url"
                    value="jdbc:mysql://localhost:3306/usermanager?characterEncoding=utf-8" />
                <property name="username" value="root" />
                <property name="password" value="root" />
            </dataSource>
        </environment>
    </environments>
</configuration>
```

四、在src下创建log4j日志

在src下创建log4j日志




```
# Global logging configuration
log4j.rootLogger=DEBUG, stdout
# Console output...
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n
```

五、创建sqlmap文件夹，创建user.xml

在项目的src文件夹下添加一个“sqlmap”文件夹，然后添加**user.xml文件**

```xml
<?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">
<!-- 写Sql语句 -->
<mapper namespace="sqlmap">
    <!-- 通过ID查询一个用户 -->
    <select id="getUser" parameterType="Integer" resultType="com.xk.javabean.User">
        select * from tuserlogin where id = #{id}
    </select>
    <insert id="addUser" parameterType="com.xk.javabean.User">
        insert into tuserlogin(username,password,grade,email) values(#{username},#{password},#{grade},#{email})
    </insert>
    <delete id="deleteUser" parameterType="Integer">
        delete from tuserlogin where id = #{id}
    </delete>
    <update id="updateUser" parameterType="String">
        update tuserlogin set username=#{username},password=#{password},grade=#{grade},email=#{email}
        where id = #{id}
    </update>
    <select id="getAllUser" resultType="com.xk.javabean.User">
        select * from tuserlogin
    </select>
</mapper>
```

六、在sqlMapConfig.xml中注册user.xml

打开“sqlMapConfig.xml”，添加如下代码：

```
<?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">
            <!-- 使用jdbc事务管理 -->
            <transactionManager type="JDBC" />
            <!-- 数据库连接池 -->
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver" />
                <property name="url"
                    value="jdbc:mysql://localhost:3306/usermanager?characterEncoding=utf-8" />
                <property name="username" value="root" />
                <property name="password" value="root" />
            </dataSource>
        </environment>
    </environments>
    <!-- mappers位置 -->
        <mappers>
            <mapper resource="sqlmap/user.xml"/>
        </mappers>
</configuration>
```

七、创建MybatisUtil工具类

```java
package com.xk.util;

import java.io.IOException;

public class MybatisUtil {

    public static SqlSessionFactory getSqlSessionFactory() throws IOException {

        String resource = "sqlMapConfig.xml";
        InputStream in = Resources.getResourceAsStream(resource);
        SqlSessionFactory  sqlSessionFactory = new SqlSessionFactoryBuilder().build(in);
        return sqlSessionFactory;
    }
    public static SqlSession getSqlSession() throws IOException {
        return getSqlSessionFactory().openSession();
    }
    //true 表示创建的SqlSession对象在执行完SQL之后会自动提交事务
    //false 表示创建的SqlSession对象在执行完SQL之后不会自动提交事务，这时就需要我们手动调用sqlSession.commit()提交事务
    public static SqlSession getSqlSession(boolean autoCommit) throws IOException {
        return getSqlSessionFactory().openSession(autoCommit);
    }
}
```

八、testAdd单元

测试testAdd

```
@Test
public void testAdd() throws Exception {

    //创建SqlSession
    SqlSession sqlSession = MybatisUtil.getSqlSession(true);
    String statement = "sqlmap.addUser";
    User user = new User();
    user.setUsername("hadoop");
    user.setPassword("h123");
    user.setGrade("1");
    user.setEmail("hadoop@126.com");
    //执行sql语句
    int result = sqlSession.insert(statement, user);
    System.out.println(user);
}
```

观察运行结果

```
DEBUG [main] - Logging initialized using 'class org.apache.ibatis.logging.slf4j.Slf4jImpl' adapter.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 1852584274.
DEBUG [main] - ==>  Preparing: insert into tuserlogin(username,password,grade,email) values(?,?,?,?)
DEBUG [main] - ==> Parameters: hadoop(String), h123(String), 1(String), hadoop@126.com(String)
DEBUG [main] - <==    Updates: 1
User [id=null, username=hadoop, password=h123, grade=1, email=hadoop@126.com]
```

51	test	123	1	test@123.com
52	test	123	1	test@123.com
53	hadoop	h123	1	hadoop@126.com

SELECT * FROM `tuserlogin` LIMIT 0, 1000

九、testDelete单元

测试testDelete

```
@Test
public void testDelete() throws IOException {
    SqlSession sqlSession = MybatisUtil.getSqlSession(true);
    String statement = "sqlmap.deleteUser";
    int result = sqlSession.delete(statement, 46);
    sqlSession.close();
    System.out.println(result);

}
```

观察运行结果

```
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [Finalizer] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 1568059495.
DEBUG [main] - ==>  Preparing: delete from tuserlogin where id = ?
DEBUG [main] - ==> Parameters: 47(Integer)
DEBUG [main] - <==    Updates: 1
DEBUG [main] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@5d76b067]
DEBUG [main] - Returned connection 1568059495 to pool.
1
```

十、testUpdate单元

测试testUpdate

```
@Test
public void testUpdate() throws IOException {
    SqlSession sqlSession = MybatisUtil.getSqlSession(true);
    String statement = "sqlmap.updateUser";
    User user = new User();
    user.setId("50");
    user.setUsername("hive");
    user.setPassword("123");
    user.setGrade("1");
    user.setEmail("hive@126.com");
    int result = sqlSession.update(statement, user);
    sqlSession.close();
    System.out.println(result);
}
```

观察运行结果

```
DEBUG [main] - Logging initialized using 'class org.apache.ibatis.logging.slf4j.Slf4jImpl' adapter.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 1852584274.
DEBUG [main] - ==>  Preparing: update tuserlogin set username=?,password=?,grade=?,email=? where id = ?
DEBUG [main] - ==> Parameters: hive(String), 123(String), 1(String), hive@126.com(String), 50(String)
DEBUG [main] - <==    Updates: 1
DEBUG [main] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@6e6c3152]
DEBUG [main] - Returned connection 1852584274 to pool.
1
```

十一、testGetAll单元

测试testGetAll

```
@Test
public void testGetAll() throws IOException {
    SqlSession sqlSession = MybatisUtil.getSqlSession(true);
    String statement = "sqlmap.getAllUser";
    List<User> users = sqlSession.selectList(statement);
    sqlSession.close();
    System.out.println(users);
}
```

观察运行结果

```
DEBUG [main] - Logging initialized using 'class org.apache.ibatis.logging.slf4j.Slf4jImpl' adapter.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 215145189.
DEBUG [main] - ==>  Preparing: select * from tuserlogin
DEBUG [main] - ==> Parameters:
DEBUG [main] - <==      Total: 38
DEBUG [main] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@cd2dae5]
DEBUG [main] - Returned connection 215145189 to pool.
[User [id=8, username=司徒, password=123456, grade=1, email=sit@126.com], User [id=9, username=wang,
```

十二、原始DAO开发

1、创建com.xk.dao.UserDao接口

```
package com.xk.dao;
import java.io.IOException;
import java.util.List;
import com.xk.javabean.User;

public interface UserDao {
    public User getUserById(Integer id)throws IOException;
    public List<User> getAllUser() throws IOException;
    public Integer deleteUserById(Integer id) throws IOException;
    public Integer modifyUser(User user) throws IOException;
    public Integer addUser(User user) throws IOException;
}
```

十二、原始DAO开发

2、创建com.xk.dao.UserDaoImpl实现类
2.1 getUserById(Integer id)方法

```
package com.xk.dao;
import java.io.IOException;
import java.util.List;
import org.apache.ibatis.session.SqlSession;
import com.xk.javabean.User;
import com.xk.util.MybatisUtil;
public class UserDaoImpl implements UserDao {
//通过用户ID查询一个用户
public User getUserById(Integer id) throws IOException{
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        User user = sqlSession.selectOne("sqlmap.getUser", id);
        sqlSession.close();
        return user;
}
```

十二、原始DAO开发

2、创建com.xk.dao.UserDaoImpl实现类
2.2 getAllUser()方法

```
//获取全体用户信息
public List<User> getAllUser() throws IOException{
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        List<User> users = sqlSession.selectList("sqlmap.getAllUser");
        sqlSession.close();
        return users;
}
```

十二、原始DAO开发

2、创建com.xk.dao.UserDaoImpl实现类
2.3 deleteUserById(Integer id)方法

```
//通过id删除一个用户
public Integer deleteUserById(Integer id) throws IOException {
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        int result = sqlSession.delete("sqlmap.deleteUser", id);
        sqlSession.close();
        return result;
}
```

十二、原始DAO开发

2、创建com.xk.dao.UserDaoImpl实现类
2.4 modifyUser(User user)方法

```
//修改用户
public Integer modifyUser(User user) throws IOException {
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        int result = sqlSession.update("sqlmap.updateUser", user);
        sqlSession.close();
        return result;
}
```

十二、原始DAO开发

2、创建com.xk.dao.UserDaoImpl实现类
2.5 addUser(User user)方法

```
//增加用户
public Integer addUser(User user) throws IOException {
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        int result = sqlSession.insert("sqlmap.addUser", user);
        sqlSession.close();
        return result;
}
```

十二、原始DAO开发

3、创建com.xk.junit.MybatisDaotest测试类

```
package com.xk.junit;
import java.io.IOException;
import java.util.List;
import org.junit.Test;
import com.xk.dao.UserDao;
import com.xk.dao.UserDaoImpl;
import com.xk.javabean.User;
public class MybatisDaotest{
private UserDao userDao = new UserDaoImpl();
        @Test
        public void testAddUser() throws IOException {
        User user = new User();
        user.setUsername("spark");
        user.setPassword("123456");
        user.setGrade("1");
        user.setEmail("spark@139.com");
        userDao.addUser(user);
}
```

十二、原始DAO开发

3、创建com.xk.junit.MybatisDaotest测试类

```
@Test
public void testDeleteUser() throws IOException    {
        userDao.deleteUserById(63);
}
@Test
public void testUpdateUser() throws IOException {
        User user = new User();
        user.setId("62");
        user.setUsername("HUAWEI");
        user.setPassword("123456");
        user.setGrade("11");
        user.setEmail("HUAWEI@hw.com");
        userDao.modifyUser(user);
}
```

十二、原始DAO开发

3、创建com.xk.junit.MybatisDaotest测试类

```
@Test
public void testGetUserById() throws Exception {
        User user = userDao.getUserById(15);
        System.out.println(user);
}
@Test
public void testGetAllUser() throws IOException{
        List<User> user = userDao.getAllUser();
        System.out.println(user);
}
}
```

十二、原始DAO开发

4、junit单元测试结果

```
DEBUG [main] - Logging initialized using 'class org.apache.ibatis.logging.slf4j.Slf4jImpl' adapter.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 1852584274.
DEBUG [main] - ==>  Preparing: insert into tuserlogin(username,password,grade,email) values(?,?,?,?)
DEBUG [main] - ==> Parameters: spark(String), 123456(String), 1(String), spark@139.com(String)
DEBUG [main] - <==    Updates: 1
DEBUG [main] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@6e6c3152]
DEBUG [main] - Returned connection 1852584274 to pool.
DEBUG [main] - Logging initialized using 'class org.apache.ibatis.logging.slf4j.Slf4jImpl' adapter.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 1852584274.
DEBUG [main] - ==>  Preparing: delete from tuserlogin where id = ?
DEBUG [main] - ==> Parameters: 63(Integer)
DEBUG [main] - <==    Updates: 0
DEBUG [main] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@6e6c3152]
DEBUG [main] - Returned connection 1852584274 to pool.
```

十二、原始DAO开发

4、junit单元测试结果

```
DEBUG [main] - Logging initialized using 'class org.apache.ibatis.logging.slf4j.Slf4jImpl' adapter.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 1852584274.
DEBUG [main] - ==>  Preparing: update tuserlogin set username=?,password=?,grade=?,email=? where id = ?
DEBUG [main] - ==> Parameters: HUAWEI(String), 123456(String), 11(String), HUAWEI@hw.com(String), 62(String)
DEBUG [main] - <==    Updates: 1
DEBUG [main] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@6e6c3152]
DEBUG [main] - Returned connection 1852584274 to pool.
DEBUG [main] - Logging initialized using 'class org.apache.ibatis.logging.slf4j.Slf4jImpl' adapter.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 215145189.
DEBUG [main] - ==>  Preparing: select * from tuserlogin where id = ?
DEBUG [main] - ==> Parameters: 15(Integer)
DEBUG [main] - <==      Total: 1
DEBUG [main] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@cd2dae5]
DEBUG [main] - Returned connection 215145189 to pool.
User [id=15, username=JavaEE, password=123456, grade=1, email=java@163.com]
```

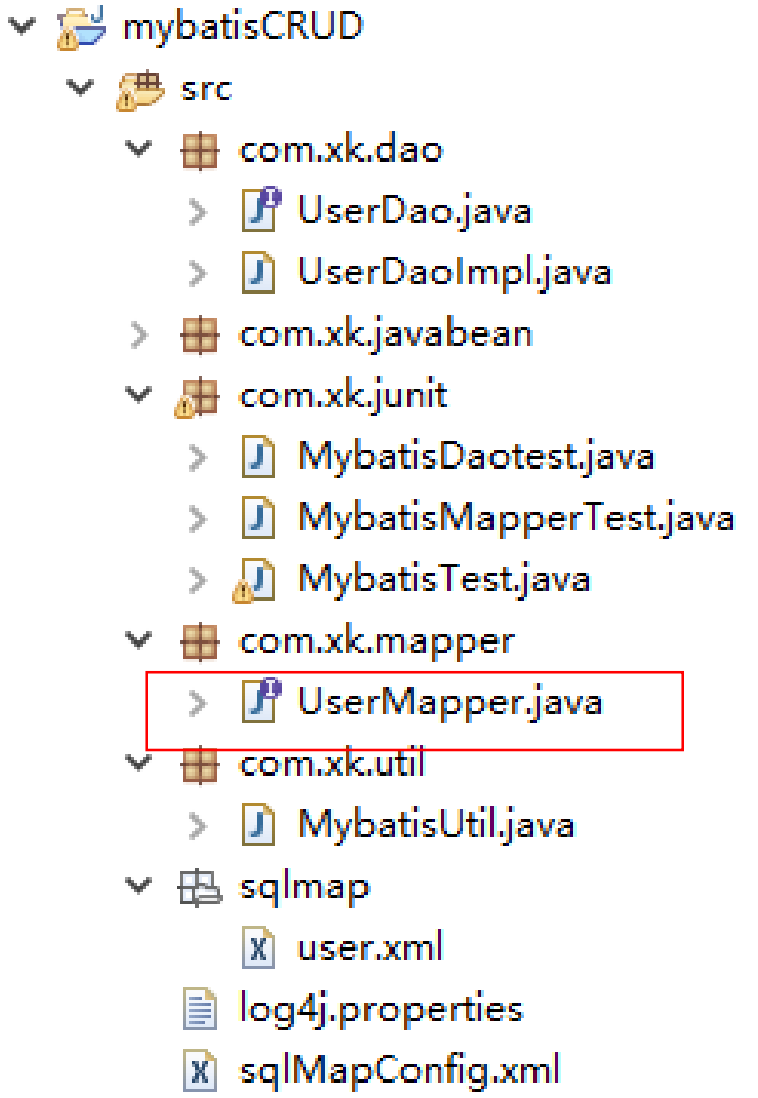
十二、原始DAO开发

4、junit单元测试结果

```
DEBUG [main] - Logging initialized using 'class org.apache.ibatis.logging.slf4j.Slf4jImpl' adapter.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - PooledDataSource forcefully closed/removed all connections.
DEBUG [main] - Opening JDBC Connection
DEBUG [main] - Created connection 215145189.
DEBUG [main] - ==>  Preparing: select * from tuserlogin
DEBUG [main] - ==> Parameters:
DEBUG [main] - <==      Total: 40
DEBUG [main] - Closing JDBC Connection [com.mysql.jdbc.JDBC4Connection@cd2dae5]
DEBUG [main] - Returned connection 215145189 to pool.
[User [id=8, username=司徒, password=123456, grade=1, email=sit@126.com], User [id=9, username=wang, pass
```

十三、Mapper动态代理方式

1、创建com.xk.mapper.UserMapper接口



十三、Mapper动态代理方式

2、com.xk.mapper.UserMapper接口代码如下

```
package com.xk.mapper;

import java.util.List;

import com.xk.javabean.User;

public interface UserMapper {

    public User getUser(Integer id);
    public List<User> getAllUser();
    public void deleteUser(Integer id);
    public void addUser(User user);
}
```

Mapper接口开发需要遵循以下规范：
1、Mapper.xml文件中的namespace与mapper接口的类路径相同。
2、Mapper接口方法名和Mapper.xml中定义的每个statement的id相同
3、Mapper接口方法的输入参数类型和mapper.xml中定义的每个sql 的parameterType的类型相同
4、Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同

十三、Mapper动态代理方式

3、修改user.xml中的namespace

```
<mapper namespace="com.xk.mapper.UserMapper">
    <!-- 通过ID查询一个用户 -->
    <select id="getUser" parameterType="Integer" resultType="com.xk.javabean.User">
        select * from tuserlogin where id = #{id}
    </select>
    <insert id="addUser" parameterType="com.xk.javabean.User">
        insert into tuserlogin(username,password,grade,email) values(#{username},#{password},
        #{grade},#{email})
    </insert>
    <delete id="deleteUser" parameterType="Integer">
        delete from tuserlogin where id = #{id}
    </delete>
    <update id="updateUser" parameterType="String">
        update tuserlogin set username=#{username},password=#{password},grade=#{grade},email=#{email}
        where id = #{id}
    </update>
    <select id="getAllUser" resultType="com.xk.javabean.User">
        select * from tuserlogin
    </select>
</mapper>
```

十三、Mapper动态代理方式

3、创建MybatisMapperTest测试类

```
package com.xk.junit;
import java.io.IOException;
...
public class MybatisMapperTest {
        @Test
        public void testGetUser() throws IOException {
                SqlSession sqlSession = MybatisUtil.getSqlSession(true);
                UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
                User user = userMapper.getUser(10);
                System.out.println(user);
        }
        @Test
        public  void testGetAllUser() throws IOException{
                SqlSession sqlSession = MybatisUtil.getSqlSession(true);
                UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
                List<User> users = userMapper.getAllUser();
                System.out.println(users);
        }
```

十三、Mapper动态代理方式

3、创建MybatisMapperTest测试类

```
@Test
public void testDeleteUser() throws IOException {
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
        userMapper.deleteUser(62);
}
@Test
public void testAddUser() throws IOException {
        SqlSession sqlSession = MybatisUtil.getSqlSession(true);
        UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
        User user = new User();
        user.setUsername("jack");
        user.setPassword("123456");
        user.setGrade("1");
        user.setEmail("jack@126.com");
        userMapper.addUser(user);
}
```

官方推荐

MyBatis官方推荐使用mapper代理方法开发mapper接口，程序员不用编写mapper接口实现类，使用mapper代理方法时，输入参数可以使用pojo包装对象或map对象，保证dao的通用性。