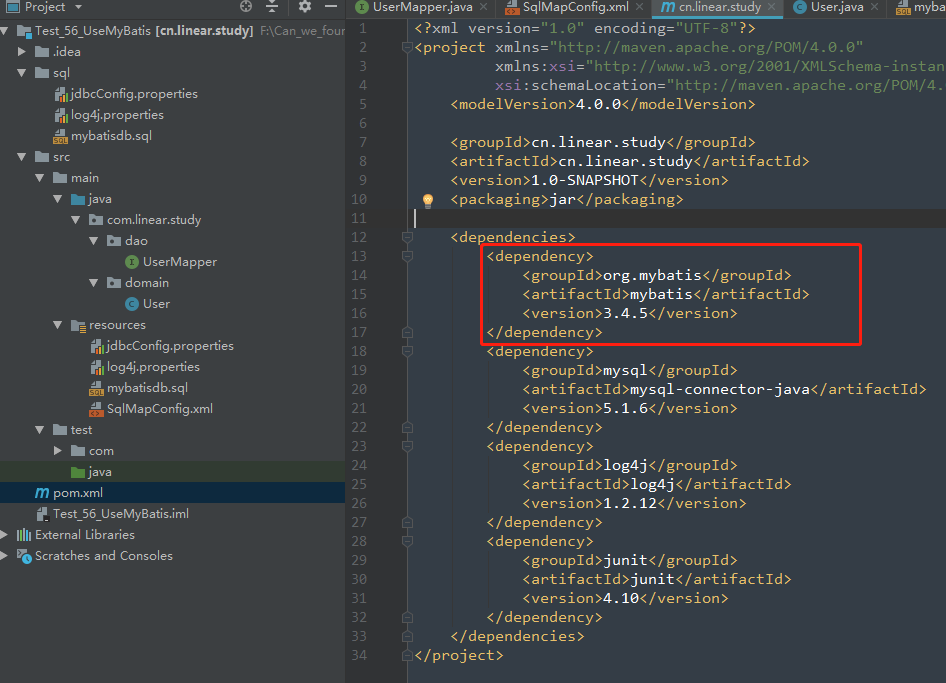
# 56.Mybatis

## 一、如何使用mybatis

### 1、pom.xml，引入相关包



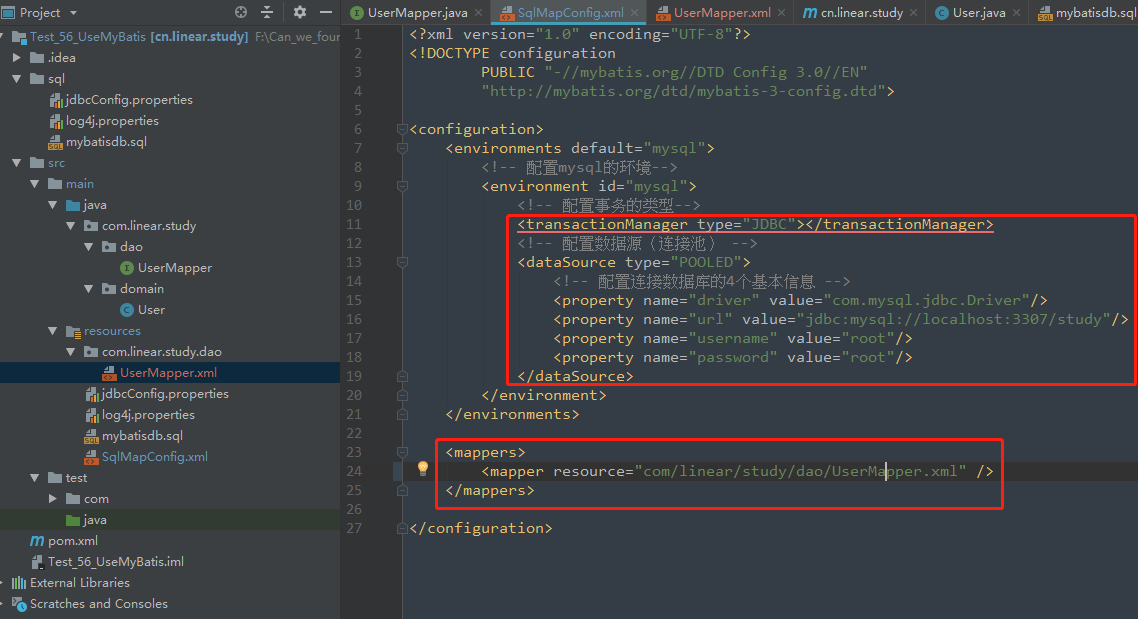
### 2、有两种配置方式，

第一种是xml配置方式，

第二种是注解配置方式

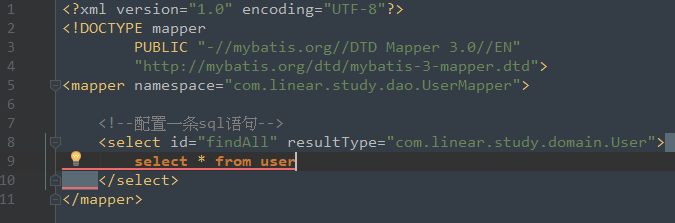
#### 2.1、XML配置

先创建一个mybatis主配置文件**/src/main/resources/SqlMapConfig.xml**：



然后创建一个**/src/main/resources/com/linear/study/dao/UserMapper.xml**，

也是在src目录下，路径名跟包名一致：



然后，此时的**/src/main/java/com/linear/study**目录下，

**/dao/UserMapper.java**和**/domain/User.java**就被绑定到这条sql语句中了。

显然，现在只需要一个UserMapper的interface，以及一个User.java的类，

而implements interface的操作，以及将查询结果封装到对象，则是根据xml配置，由mybatis自动完成。

接下来就可以开始用了！

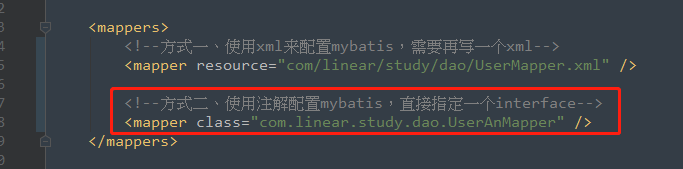
|  |
| --- |
| // 1、读取mybatis主配置 InputStream in = Resources.*getResourceAsStream*("SqlMapConfig.xml");  // 2、使用SqlSessionFactoryBuilder，创建一个builder，然后用builder创建一个工厂 SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder(); SqlSessionFactory factory = builder.build(in);  // 3、使用工厂生产SqlSession对象 SqlSession session = factory.openSession();  // 4、利用SqlSession对象，创建UserMapper的代理对象 UserMapper userMapper = session.getMapper(UserMapper.class);  // 5、可以直接用这个代理对象，去调用dao的方法 List<User> users = userMapper.findAll();  // 6、将获取到的数据打印出来 for(User user:users){  System.*out*.println(user); }  // 7、执行完毕，关闭连接 session.close(); in.close(); |

#### **2.2、使用注解来配置mybatis完成sql查询**

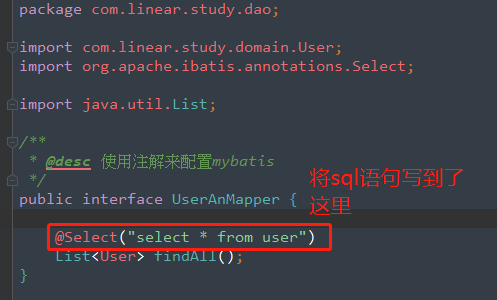
不同于2.1，**此时不需要UserMapper.xml了**，

首先修改一下**SqlMapConfig.xml**，

mapper标签配置class属性，指定一个UserAnMapper的interface



然后修改一下**UserAnMapper.class**：



接着就可以写测试，和上面的代码基本一致，有变化的是第4项，用的类的名字有变化。

|  |
| --- |
| // xml方式，引入UserMapper // 4、利用SqlSession对象，创建UserMapper的代理对象 UserMapper userMapper = session.getMapper(UserMapper.class);  // 注解方式，使用UserAnMapper // 4、利用SqlSession对象，创建UserMapper的代理对象 UserAnMapper userMapper = session.getMapper(UserAnMapper.class); |

## 二、自己实现一个mybatis

### **1、技术要点**

* 工厂模式（Factory）
* 构造者模式（Builder）
* 代理模式（Proxy）
* 反射
* 自定义注解
* 注解的反射
* xml解析
* 数据库元数据
* 元数据反射

### **2、思路**

先以xml配置方式入手。

（1）根据mybatis的xml文件，解析xml，

获取数据库连接的相关配置，创建Connection对象，注册驱动，获取数据库连接。

（2）数据库查询的语句也在xml里面，读取这个dao的xml，获取配置的sql语句。

获取预处理对象PreparedStatement，将sql语句传入。

然后执行查询，

|  |
| --- |
| ResultSet resultSet = preparedStatement.executeQuery(); |

然后遍历resultSet，将其add到一个List集合中。

最后把这个list集合return出去。

|  |
| --- |
| List<E> list = new ArrayList();  while(resultSet.next()){  E element = (E) Class.forName(xml里面的全限定类名).newInstance();  list.add(element);  }  return list; |

### **3、具体实现**

有点复杂啊...细节有点多...