一对多查询

一、@Many注解

二、步骤

在基于xml的一对多查询基础上，删掉Mapper.xml文件（IUserDao.xml)

1、仍需要配置映射Account

```
public class Account implements Serializable {

    private Integer id;
    private Integer uid;
    private Double money;

    //多对一（mybatis中称之为一对一）的映射：一个账户只能属于一个用户
    private User user;

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Integer getUid() {
        return uid;
    }
```

```
    public void setUid(Integer uid) {
        this.uid = uid;
    }

    public Double getMoney() {
        return money;
    }

    public void setMoney(Double money) {
        this.money = money;
    }

    @Override
    public String toString() {
        return "Account{" +
                "id=" + id +
                ", uid=" + uid +
                ", money=" + money +
                '}';
    }
}
```

2、配置IUserDao 接口

```
@CacheNamespace(blocking = true)
public interface IUserDao {

    /**
     * 查询所有用户
     * @return
     */
    @Select("select * from user")
    @Results(id="userMap",value={
            @Result(id=true,column = "id",property = "userId"),
            @Result(column = "username",property = "userName"),
            @Result(column = "address",property = "userAddress"),
            @Result(column = "sex",property = "userSex"),
            @Result(column = "birthday",property = "userBirthday"),
            @Result(property = "accounts",column = "id",
                    many = @Many(select =
"com.itheima.dao.IAccountDao.findAccountByUid",
                            fetchType = FetchType.LAZY))
    })
    List<User> findAll();
```

```
    /**
     * 根据id查询用户
     * @param userId
     * @return
     */
    @Select("select * from user  where id=#{id} ")
    @ResultMap("userMap")
    User findById(Integer userId);

    /**
     * 根据用户名称模糊查询
     * @param username
     * @return
     */
    @Select("select * from user where username like #{username} ")
    @ResultMap("userMap")
    List<User> findUserByName(String username);


}
```

3、测试

```
public class AnnotationCRUDTest {
    private InputStream in;
    private SqlSessionFactory factory;
    private SqlSession session;
    private IUserDao userDao;

    @Before
    public  void init()throws Exception{
        in = Resources.getResourceAsStream("SqlMapConfig.xml");
        factory = new SqlSessionFactoryBuilder().build(in);
        session = factory.openSession();
        userDao = session.getMapper(IUserDao.class);
    }

    @After
    public  void destroy()throws  Exception{
        session.commit();
        session.close();
        in.close();
    }

    @Test
    public  void  testFindAll(){
```

```
        List<User> users = userDao.findAll();
//        for(User user : users){
//            System.out.println("---每个用户的信息----");
//            System.out.println(user);
//            System.out.println(user.getAccounts());
//        }
    }

    @Test
    public void testFindOne(){
        User user = userDao.findById(57);
        System.out.println(user);
    }


    @Test
    public  void testFindByName(){
        List<User> users = userDao.findUserByName("%mybatis%");
        for(User user : users){
            System.out.println(user);
        }
    }
}
```