一对多查询(（多对一)

一、主要标签

collection

1、属性

property

ofType

二、示例

在一对一示例的基础上：

需求：

查询所有用户User信息及用户关联的账户Acount信息。

分析：

用户信息和他的账户信息为一对多关系，并且查询过程中如果用户没有账户信息，此时也要将用户信息查询出来，我们想到了左外连接查询比较合适。

代码实现

1、User 类加入 List<Account>获取账户信息

```
public class User implements Serializable {

    private Integer id;
    private String username;
    private String address;
    private String sex;
    private Date birthday;

    //一对多关系映射：主表实体应该包含从表实体的集合引用
    private List<Account> accounts;

    public List<Account> getAccounts() {
        return accounts;
    }

    public void setAccounts(List<Account> accounts) {
        this.accounts = accounts;
    }

    public Integer getId() {
        return id;
    }

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

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getSex() {
        return sex;
    }

    public void setSex(String sex) {
        this.sex = sex;
    }

    public Date getBirthday() {
        return birthday;
    }

    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                ", address='" + address + '\'' +
                ", sex='" + sex + '\'' +
                ", birthday=" + birthday +
                '}';
    }
}
```

2、用户持久层Dao 接口中加入查询方法

```
List<User> findAll();
```

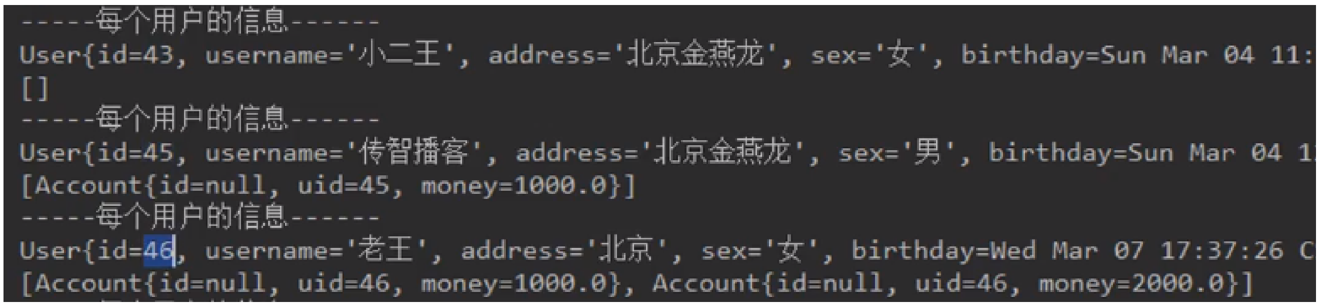
3、用户持久层UserDao.xml映射文件配置

```
<?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 namespace="com.itheima.dao.IUserDao">

    <!-- 定义User的resultMap-->
    <resultMap id="userAccountMap" type="user">
        <id property="id" column="id"></id>
        <result property="username" column="username"></result>
        <result property="address" column="address"></result>
        <result property="sex" column="sex"></result>
        <result property="birthday" column="birthday"></result>

        <!-- 配置user对象中accounts集合的映射 -->
        <collection property="accounts" ofType="account">
            <id column="aid" property="id"></id>
            <result column="uid" property="uid"></result>
            <result column="money" property="money"></result>
        </collection>
    </resultMap>

    <!-- 查询所有 -->
    <select id="findAll" resultMap="userAccountMap">
        select * from user u left outer join account a on u.id = a.uid
    </select>

    <!-- 根据id查询用户 -->
    <select id="findById" parameterType="INT" resultType="user">
        select * from user where id = #{uid}
    </select>

</mapper>
```

4、测试

```
public class UserTest {

    private InputStream in;
    private SqlSession sqlSession;
    private IUserDao userDao;

    @Before//用于在测试方法执行之前执行
    public void init()throws Exception{
        //1.读取配置文件，生成字节输入流
        in = Resources.getResourceAsStream("SqlMapConfig.xml");
        //2.获取SqlSessionFactory
        SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(in);
        //3.获取SqlSession对象
        sqlSession = factory.openSession(true);
        //4.获取dao的代理对象
        userDao = sqlSession.getMapper(IUserDao.class);
    }

    @After//用于在测试方法执行之后执行
    public void destroy()throws Exception{
        //提交事务
        // sqlSession.commit();
        //6.释放资源
        sqlSession.close();
        in.close();
    }

    /**
     * 测试查询所有
     */
    @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());
        }
    }
}
```

5、测试结果



```
-----每个用户的信息------
User{id=43, username='小二王', address='北京金燕龙', sex='女', birthday=Sun Mar 04 11:
[]
-----每个用户的信息------
User{id=45, username='传智播客', address='北京金燕龙', sex='男', birthday=Sun Mar 04 1
[Account{id=null, uid=45, money=1000.0}]
-----每个用户的信息------
User{id=46, username='老王', address='北京', sex='女', birthday=Wed Mar 07 17:37:26 C
[Account{id=null, uid=46, money=1000.0}, Account{id=null, uid=46, money=2000.0}]
```

可以看到，id=46的用户有2个账号，且会显示在一条记录里，这是mybatis的作用