实验四：MyBatis 的关联映射

一 、**实验目的：**熟练掌握实体之间的各种映射关系。

二 、**预习要求：**预习数据库原理中所讲过的一对一、一对多和多对多关系

三、**实验内容：**

1. 查询所有订单信息，关联查询下单用户信息(注意：因为一个订单信息只会是一个人下的订单，所以从查询订单信息出发关联查询用户信息为一对一查询。如果从用户信息出发查询用户下的订单信息则为一对多查询，因为一个用户可以下多个订单。)

2. 查询所有用户信息及用户关联的订单信息(用户信息和订单信息为一对多关系)。

四、**实验方法和步骤：**

1. 查询所有订单信息，关联查询下单用户信息(注意：因为一个订单信息只会是一个人下的订单，所以从查询订单信息出发关联查询用户信息为一对一查询。如果从用户信息出发查询用户下的订单信息则为一对多查询，因为一个用户可以下多个订单。)

sql 语句：

```
SELECT
o.id,
o.user_id    userId,
o.number,
o.createtime,
o.note,
u.username,
u.address
FROM
`order` o
LEFT JOIN `user` u ON o.user_id = u.id
```

⑴方法一：使用 resultType

使用 resultType，改造订单 pojo 类，此 pojo 类中包括了订单信息和用户信息,这样返回对象的时候，mybatis 自动把用户信息也注入进来了

①创建 pojo 类--创建 Order 订单类

```
public class Order {
    // 订单 id
    private int id;
    // 用户 id
    private Integer userId;
    // 订单号
    private String number;
    // 订单创建时间
    private Date createtime;
    // 备注
    private String note;
get/set。。。
toString();
}
```

创建包含用户的订单类，OrderUser 类继承 Order 类后 OrderUser 类包括了 Order 类的所有字段，只需定义用户的信息字段即可，如下图：

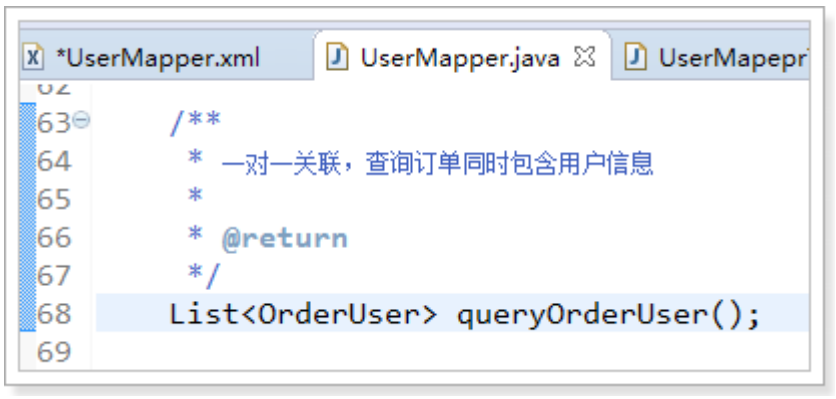
```
public class OrderUser extends Order {
    private String username;
    private String password;
set/get…
}
```

②Mapper.xml,在 UserMapper.xml 添加 sql，如下

```
<!-- 查询订单，同时包含用户数据 -->
```

```
<select id="queryOrderUser" resultType="com.haust.pojo.OrderUser">
    SELECT
    o.id,
    o.user_id    userId,
    o.number,
    o.createtime,
    o.note,
    u.username,
    u.address
    FROM
    `order` o
    LEFT JOIN `user` u ON o.user_id = u.id
</select>
```

③Mapper 接口在 UserMapper 接口添加方法，如下图：



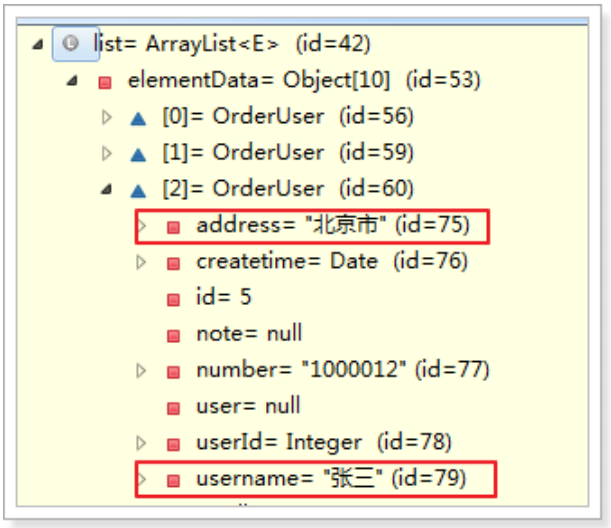
④测试方法：

在 UserMapperTest 添加测试方法，如下：

```
@Test
public void testQueryOrderUser() {
    // mybatis 和 spring 整合，整合之后，交给 spring 管理
    SqlSession sqlSession = this.sqlSessionFactory.openSession();
    // 创建 Mapper 接口的动态代理对象，整合之后，交给 spring 管理
```

```
        UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
        // 使用 userMapper 执行根据条件查询用户
        List<OrderUser> list = userMapper.queryOrderUser();
        for (OrderUser ou : list) {
            System.out.println(ou);
        }
        // mybatis 和 spring 整合，整合之后，交给 spring 管理
        sqlSession.close();
}
```

⑤实验效果测试结果如下图：

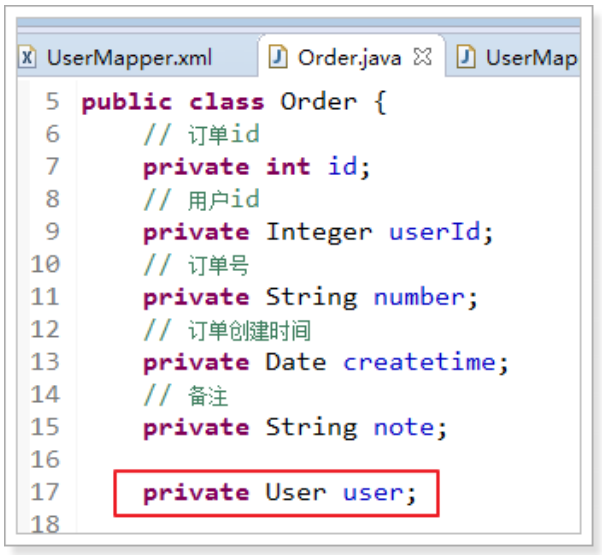


小结　　定义专门的 pojo 类作为输出类型，其中定义了 sql 查询结果集所有的字段。此方法较为简单，企业中使用普遍。

(2)方法二：使用 resultMap，使用 resultMap，定义专门的 resultMap 用于映射一对一查询结果。

①改造 pojo 类

在 Order 类中加入 User 属性，user 属性中用于存储关联查询的用户信息，因为订单关联查询用户是一对一关系，所以这里使用单个 User 对象存储关联查询的用户信息。改造 Order 如下图：



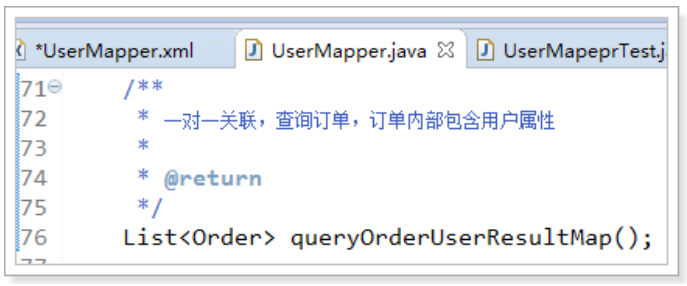
②Mapper.xml 这里 resultMap 指定 orderUserResultMap，如下：

```
<resultMap type="order" id="orderUserResultMap">
    <id property="id" column="id" />
    <result property="userId" column="user_id" />
    <result property="number" column="number" />
    <result property="createtime" column="createtime" />
    <result property="note" column="note" />
    <!-- association ： 配置一对一属性 -->
    <!-- property:order 里面的 User 属性名 -->
    <!-- javaType:属性类型 -->
    <association property="user" javaType="user">
        <!-- id:声明主键，表示 user_id 是关联查询对象的唯一标识-->
```

```
            <id property="id" column="user_id" />
            <result property="username" column="username" />
            <result property="address" column="address" />
        </association>
</resultMap>
<!-- 一对一关联，查询订单，订单内部包含用户属性 -->
<select id="queryOrderUserResultMap" resultMap="orderUserResultMap">
        SELECT
        o.id,
        o.user_id,
        o.number,
        o.createtime,
        o.note,
        u.username,
        u.address
        FROM
        `order` o
        LEFT JOIN `user` u ON o.user_id = u.id
</select>
```

③Mapper 接口

编写 UserMapper 如下图：



④测试方法

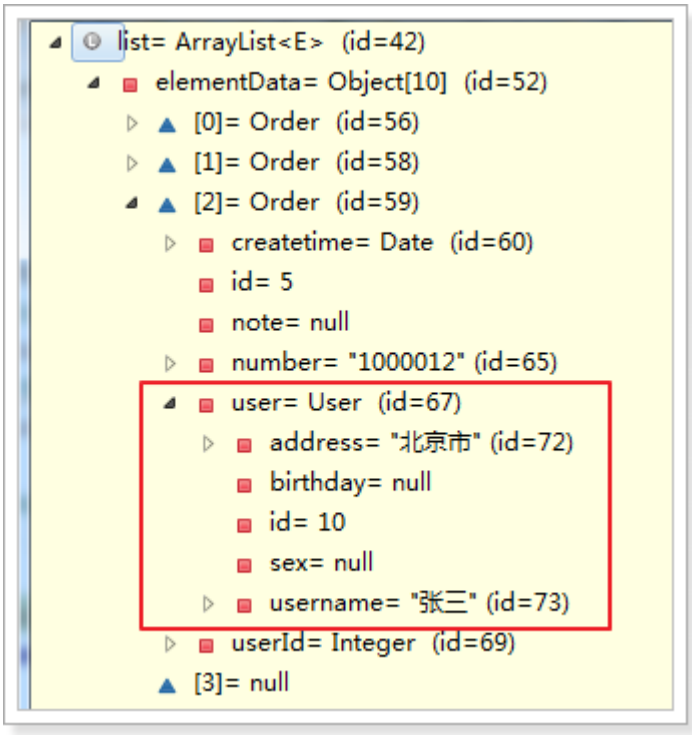
在 UserMapperTest 增加测试方法，如下：

```
@Test
public void testQueryOrderUserResultMap() {
```

```
        // mybatis 和 spring 整合，整合之后，交给 spring 管理
        SqlSession sqlSession = this.sqlSessionFactory.openSession();
        // 创建 Mapper 接口的动态代理对象，整合之后，交给 spring 管理
        UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
        // 使用 userMapper 执行根据条件查询用户
        List<Order> list = userMapper.queryOrderUserResultMap();
        for (Order o : list) {
            System.out.println(o);
        }
        // mybatis 和 spring 整合，整合之后，交给 spring 管理
        sqlSession.close();
}
```

⑤实验效果

测试效果如下图：



2. 查询所有用户信息及用户关联的订单信息(用户信息和订单信息为一对多关系)。

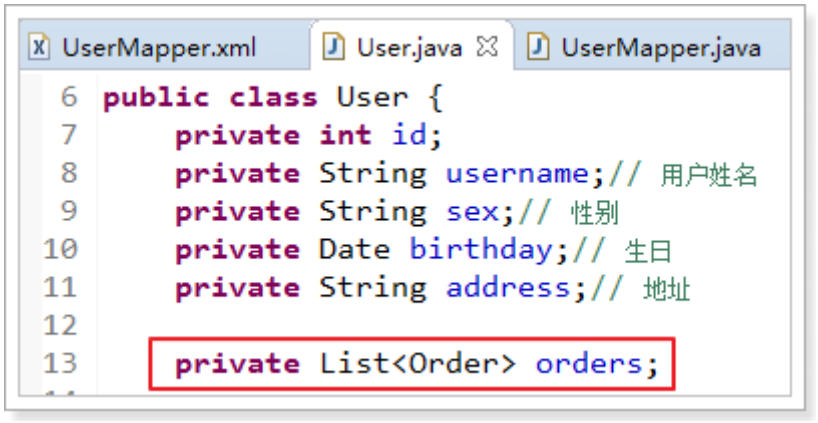
sql 语句：

```
SELECT
```

```
    u.id,
    u.username,
    u.birthday,
    u.sex,
    u.address,
    o.id oid,
    o.number,
    o.createtime,
    o.note
FROM
    `user` u
LEFT JOIN `order` o ON u.id = o.user_id
```

①修改 pojo 类

在 User 类中加入 List<Order> orders 属性,如下图：



②Mapper.xml

在 UserMapper.xml 添加 sql，如下：

```
<resultMap type="user" id="userOrderResultMap">
    <id property="id" column="id" />
    <result property="username" column="username" />
    <result property="birthday" column="birthday" />
    <result property="sex" column="sex" />
```

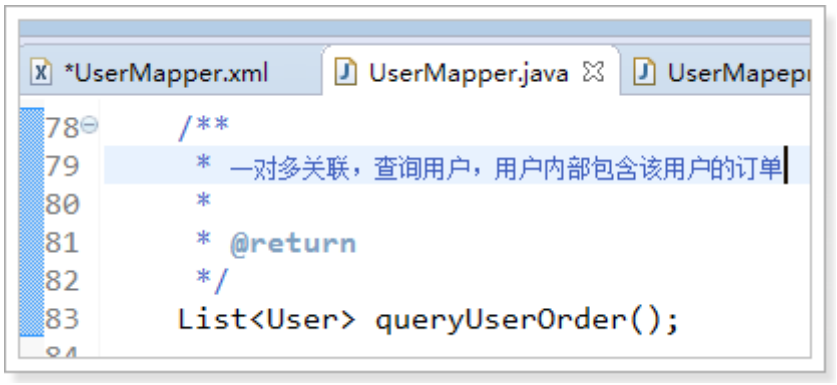
```
        <result property="address" column="address" />
        <!-- 配置一对多的关系 -->
        <collection property="orders" javaType="list" ofType="order">
            <!-- 配置主键，是关联 Order 的唯一标识 -->
            <id property="id" column="oid" />
            <result property="number" column="number" />
            <result property="createtime" column="createtime" />
            <result property="note" column="note" />
        </collection>
</resultMap>
```

<!-- 一对多关联，查询订单同时查询该用户下的订单 -->

```
<select id="queryUserOrder" resultMap="userOrderResultMap">
        SELECT
        u.id,
        u.username,
        u.birthday,
        u.sex,
        u.address,
        o.id oid,
        o.number,
        o.createtime,
        o.note
        FROM
        `user` u
        LEFT JOIN `order` o ON u.id = o.user_id
</select>
```

③Mapper 接口

编写 UserMapper 接口，如下图：



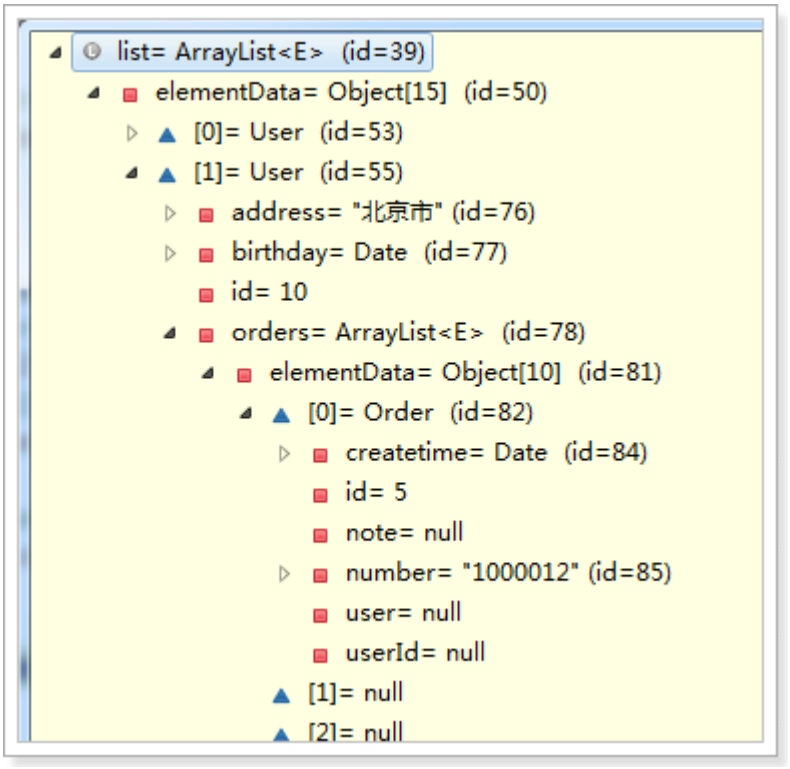
④测试方法

在 UserMapperTest 增加测试方法，如下

```
@Test
public void testQueryUserOrder() {
    // mybatis 和 spring 整合，整合之后，交给 spring 管理
    SqlSession sqlSession = this.sqlSessionFactory.openSession();
    // 创建 Mapper 接口的动态代理对象，整合之后，交给 spring 管理
    UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
    // 使用 userMapper 执行根据条件查询用户
    List<User> list = userMapper.queryUserOrder();
    for (User u : list) {
        System.out.println(u);
    }
    // mybatis 和 spring 整合，整合之后，交给 spring 管理
    sqlSession.close();
}
```

⑤效果

测试效果如下图：



五、 思考题：

关联映射在 MyBatis 框架中的重要作用？

六、 实验作业要求：

（一） 实验目的：

（二） 实验内容：

（三） 实验结果：可以是运行结果截图或其他形式的结果展示

（四） 问题及解决：实验中遇到的问题及解决方法。

（五） 回答思考题提出的问题。