一对一查询

主要标签：association（应用于一对一）

1、作用：完成子对象的映射

2、属性

property:映射数据库列的字段或属性。

colum:数据库的列名或者列标签别名。

javaTyp:完整java类名或别名。

jdbcType支持的JDBC类型列表列出的JDBC类型。这个属性只在insert,update或delete的时候针对允许空的列有用。

resultMap: 一个可以映射联合嵌套结果集到一个适合的对象视图上的ResultMap。这是一个替代的方式去调用另一个select语句。

```
<!-- 一对一的关系映射：配置封装user的内容-->
<association property="user" column="uid" javaType="user">
   <id property="id" column="id"></id>
   <result column="username" property="username"></result>
</association>
```

一、示例

1、用户和账户

一个用户可以有多个账户

一个账户只能属于一个用户（多个账户也可以属于同一个用户）

2、步骤：

1、建立两张表：用户表User　　id; username; address; sex;　birthday;

账户表Account　id, uid, money

让用户表和账户表之间具备一对多的关系：需要使用外键在账户表中添加

2、建立两个实体类：用户实体类User.java和账户实体类Account.java

让用户和账户的实体类能体现出来一对多的关系

3、建立两个配置文件

用户的配置文件

账户的配置文件

4、实现配置：

当我们查询用户时，可以同时得到用户下所包含的账户信息

当我们查询账户时，可以同时得到账户的所属用户信息

3、通过两张表查找以下信息

ID	UID	MONEY	username	address
1	41	1000	老王	北京
2	45	1000	传智播客	北京金燕龙

二、代码实现

方法一：核心思想扩展account对象，来完成映射

1、通过新建AccountUser继承account

```
public class AccountUser extends Account {

    private String username;
    private String address;

    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;
    }

    @Override
    public String toString() {
        return super.toString()+"        AccountUser{" +
                "username='" + username + '\'' +
                ", address='" + address + '\'' +
                '}';
    }
}
```

2、查找

```
    <!--查询所有账户同时包含用户名和地址信息-->
<select id="findAllAccount" resultType="accountuser">
    select a.*,u.username,u.address from account a , user u where u.id = a.uid;
</select>
```

方法二：面向对象的思想，在Account对象中添加User对象(在User对象中添加Account对象)

1、在Account对象中添加User对象

```
public class Account implements Serializable {

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

    //从表实体应该包含一个主表实体的对象引用
    //在Account对象中添加User
    private User user;
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }
}
```

2、在Account.xml映射文件里配置resultMap完成映射

```
  <!-- 定义封装account和user的resultMap -->
    <resultMap id="accountUserMap" type="account">
```

```
        <id property="id" column="aid"></id>
        <result property="uid" column="uid"></result>
        <result property="money" column="money"></result>

        <!-- 一对一的关系映射：配置封装user的内容-->
        <association property="user" column="uid" javaType="user">
            <id property="id" column="id"></id>
            <result column="username" property="username"></result>
            <result column="address" property="address"></result>
            <result column="sex" property="sex"></result>
            <result column="birthday" property="birthday"></result>
        </association>
    </resultMap>

    <!-- 查询所有 -->
    <select id="findAll" resultMap="accountUserMap">
        select u.*,a.id as aid,a.uid,a.money from account a , user u where u.id = a.uid;
    </select>
</mapper>
```