**一对一关系映射**

方式一：resultType

第一步：创建pojo

将sql查询的 订单orders和用户user结果集 全部成功映射到 pojo中，注意应继承字段比较多的pojo类

public class OrdersCustom extends Orders{

//继承Orders是为了补充添加以下用户属性

/\*USER.username,

USER.sex,

USER.address \*/

private String username;

private String sex;

private String address;

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

第二步：配置OrdersMapperCustom.xml

<?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.yb.mapper.OrderCustomMapper">

<select id="findOrderUser" resultType="com.yb.entity.OrdersCustom">

SELECT

orders.\* ,

USER.username,

USER.sex,

USER.address

FROM

orders,

USER

where

orders.user\_id=user.id

</select>

</mapper>

第三步：添加OrdersMapperCustom.java接口实现以下方法

public List<OrdersCustom> findOrderUser();

第四步：在OrdersMapperCustomTest.java中测试.

public class OrdersMapperCustomTest {

@Test

public void testFindOrdersUser() throws Exception {

InputStream inputStream = Resources.getResourceAsStream("mybatis-config.xml");

SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);

//得到会话sqlSession

SqlSession sqlSession=sqlSessionFactory.openSession();

//创建代理对象

OrderCustomMapper mapper = sqlSession.getMapper(OrderCustomMapper.class);

//调用mapper的方法

List<OrdersCustom> list=mapper.findOrderUser();

System.out.println(list);

sqlSession.close();

}

}

方式二：resultMap

第一步：同上

第二步：配置OrdersMapperCustom.xml

<?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">

<!--方式二resultMap

定义一个resultMap,订单查询 关联 用户的 ,将整个查询的结果映射到cn.itcast.mybatis.pojo.Orders中-->

<resultMap type="com.yb.entity.Orders" id="OrdersUserResultMap">

<!-- (1) 配置订单信息 -->

<!-- 数据库表orders对应 pojo中的Orders.java -->

<!-- id:唯一标识

column：订单信息的唯 一标识 列

property：订单信息的唯 一标识 列所映射到Orders中哪个属性

result:普通标识 -->

<id column="id" property="id"/>

<result column="user\_id" property="userId"/>

<result column="number" property="number"/>

<result column="createtime" property="createtime"/>

<result column="note" property="note"/>

<!-- (2)配置关联用户信息 -->

<!-- 数据库表user对应 pojo中的User.java -->

<!-- association：用于映射关联查询单个对象的信息

property：要将关联查询的用户信息映射到Orders中哪个属性

javaType：映射到user的哪个属性-->

<association property="user" javaType="com.yb.entity.User">

<!-- id：关联查询用户的唯 一标识

column：指定唯 一标识用户信息的列

result:普通标识

-->

<id column="user\_id" property="id"/>

<result column="username" property="username"/>

<result column="sex" property="sex"/>

<result column="address" property="address"/>

</association>

</resultMap>

<!--第二种：举例一对一： ResultMap:查询订单关联查询用户信息 -->

<select id="findOrdersUserResultMap" resultMap="OrdersUserResultMap">

SELECT

orders.\* ,

USER.username,

USER.sex,

USER.address

FROM

orders,

USER

WHERE

orders.user\_id=user.id

</select>

</mapper>

第三步：在接口OrdersMapperCustom.java中添加方法

public List<Orders> findOrdersUserResultMap();

第四步：测试

@Test

public void testFindOrdersUserResultMap() throws Exception {

InputStream inputStream = Resources.getResourceAsStream("mybatis-config.xml");

SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);

//得到会话sqlSession

SqlSession sqlSession=sqlSessionFactory.openSession();

//创建代理对象

OrderCustomMapper mapper = sqlSession.getMapper(OrderCustomMapper.class);

//调用mapper的方法,返回映射在pojo 中Orders(里面补充了User user)

List<Orders>list=mapper.findOrdersUserResultMap();

System.out.println(list);

sqlSession.close();

}