输出映射

1. resultType

使用resultType进行输出映射时，只有查询出来的列名和pojo中的属性名一致，该列才可以映射成功。
如果查询出来的列名和pojo中的属性名全部不一致，没有创建pojo对象。
只要查询出来的列名和pojo的属性名有一个一致，就会创建pojo对象。
可以使用 `select id id_ from user` 测试。

1. 输出简单类型

- 需求
  - 用户信息的综合查询列表总数，通过查询总数和上边用户综合查询列表才可以实现分页。
- mapper.xml

```
<!-- 用户信息的综合查询总数 -->
<select id="findUserCount" parameterType="com.swxc.mybatis.po.UserQueryVo" resultType="int">
    select count(*) from user where user.sex = #{userCustom.sex} and user.username like '%${userCustom.username}%'
</select>
```

- mapper.java

```
// 用户信息综合查询总数
public int findUserCount(UserQueryVo userQueryVo) throws Exception;
```

- 测试

```
// 用户信息的综合查询总数
@Test
public void findUserCountTest() throws Exception {
    SqlSession sqlSession = sqlSessionFactory.openSession();
    UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
    UserQueryVo userQueryVo = new UserQueryVo();
    UserCustom userCustom = new UserCustom();
    userCustom.setSex("g");
    userCustom.setUsername("五");
    userQueryVo.setUserCustom(userCustom);
    int count = userMapper.findUserCount(userQueryVo);
    sqlSession.close();
    System.out.println(count);
}
```

- 小结
  - 查询出来的结果集只要一行且一列，可以使用简单类型进行输出映射。

2. 输出pojo对象和pojo列表

- 不管是输出的pojo单个对象还是一个列表（list中包括pojo），在mapper.xml中resultType指定的类型是一样的，在mapper.java中指定的方法的返回值类型不一样。
  - 如果输出的是单个对象，方法返回值是单个对象类型；
  - 如果输出的是pojo对象列表，方法返回值是List
- 生成的动态代理对象中是根据mapper方法的返回值类型确认是调用selectOne（返回单个对象调用）还是selectList（返回集合对象调用）。

3. 输出hashmap

- 输出pojo对象可以改用hashmap输出类型，将输出的字段名称作为map的key，value为字段值。

3. resultMap

mybatis中使用resultMap完成高级输出结果映射。

1. resultMap使用方法

- 如果查询出来的列名和pojo的属性名不一致，通过定义一个resultMap对列名和属性名之间做一个映射关系。
- 定义resultMap
- 使用resultMap作为statement的输出映射类型

2. 将下边的sql使用User完成映射

- select id id,*username username* from user where id = #{id}
- User类中属性的列名和上边查询的列名不一致。
  - 定义resultMap

```
<!-- 定义resultMap
 将select id id_,username username_ from user where id = #{id} 和User类中的属性作一个映射关系
 id：对resultMap的唯一标识
 type：resultMap最终映射的java对象类型，可以使用别名。
 -->
<resultMap id="userResultMap" type="user">
    <!-- id表示的是查询结果集中唯一标识
     column：查询出来的列名
     property：type所指定的pojo中的属性名
     最终resultMap对column和property作一个映射关系（对应关系）
     -->
    <id column="id_" property="id" />
    <!--
    对普通列的映射定义
    column：查询出来的列名
    property：type所指定的pojo中的属性名
    最终resultMap对column和property作一个映射关系（对应关系）
    -->
    <result column="username_" property="username" />
</resultMap>
```

  - 使用resultMap作为statement的输出映射类型

```
<!-- 使用resultMap进行输出映射
 resultMap：指定定义的resultMap的id，如果这个resultMap在其他的mapper文件中，前边需要加namespace
 -->
<select id="findUserByIdResultMap" parameterType="int" resultMap="userResultMap">
    select id id_,username username_ from user where id = #{id}
</select>
```

  - mapper接口 mapper.java

```
// 根据id查询用户信息，使用resultMap输出
public User findUserByIdResultMap(int id) throws Exception;
```

  - 测试

```
// 根据id查询用户，使用resultMap进行输出
@Test
public void testFindUserByIdResultMap() throws Exception {
    SqlSession sqlSession = sqlSessionFactory.openSession();
    UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
    User user = userMapper.findUserByIdResultMap(1);
    sqlSession.close();
    System.out.println(user);
}
```