

# resultMap

## 1 学习目标

1. 重点掌握ResultMap的作用
2. 重点掌握ResultMap的使用方式
3. 重点掌握ResultMap的级联属性

## 2 resultMap和resultMap

### 2.1 resultMap--自动映射

情况一：如果查询结果是一行数据: 例如根据id查询某个职位信息

使用 POJO 封装一行数据

- 接口中的方法的返回值就是用于封装结果集的 POJO 类型

```
1 public Teacher getTeacherById(String jobId);
```

- select标签中的 resultMap 属性的值是封装结果集的POJO的全限定名

```
1 <select id="getTeacherById" resultMap="cn.tedu.pojo.Teacher">
2     SELECT id, name, age, title, manager, salary, comm, gender, subject_id
3     FROM teacher
4     WHERE id = #{id}
5 </select>
```

情况二：如果查询结果为多行数据: 例如查询所有职位信息

使用 List<POJO> 封装多行数据

- 接口中的方法的返回值是 List<POJO> 类型

```
1 public List<Teacher> getTeacherAll();
```

- select标签的 resultMap 属性的值是POJO的全限定名

```
1 <select id="getTeacherAll" resultMap="cn.tedu.pojo.Teacher">
2     SELECT id, name, age, title, manager, salary, comm, gender, subject_id
3     FROM teacher
4 </select>
```

### 2.2 resultMap--手动映射

- 使用mybatis，有两个属性标签 <resultType>、<resultMap> 可以提供结果映射
- resultMap标签: 是用于定义javaBean和数据库表的映射规则,建议只要定义resultMap,就将整表的全部列的映射全部写上
  - id属性: 给resultMap的映射关系自定义一个名称,是唯一值
  - type属性: 用于指定需要自定义规则的Java类型,其中如果设置了包扫描,就可以直接写类名即可
- id标签: 指定主键列的封装规则,也可以使用result标签定义,但是对主键的特殊照顾,底层会有优化
  - column属性: 映射主键列
  - property属性: 映射主键属性

——result标签: 指定普通列的封装规则

- result对象: 提供是 映射非主键列
  - column属性: 映射非主键列
  - property属性: 映射非主键属性

## 3 入门案例

### 3.1 前期准备

①在JSDSecondStage项目下,创建 ResultMapDemo ,将版本设置为2.5.4

②在 pom.xml 中添加相关的依赖

```
1  <!--mysql数据库驱动依赖-->
2  <dependency>
3      <groupId>mysql</groupId>
4      <artifactId>mysql-connector-java</artifactId>
5      <scope>runtime</scope>
6  </dependency>
7  <!--引入相关mybatis依赖-->
8  <dependency>
9      <groupId>org.mybatis.spring.boot</groupId>
10     <artifactId>mybatis-spring-boot-starter</artifactId>
11     <version>2.2.0</version>
12 </dependency>
```

③在模块下的 src/main/java/cn/tedu 包中,将 pojo 包导入进去,该包下包含两个类

- Teacher tedu库中teacher表的javaBean类
- Subject tedu库中subject表的javaBean类

④在模块下的 src/main/java/cn/tedu 包中,将 mapper 包导入进去,该包下包含两个接口

- TeacherMapper 用于定义映射teacher表的接口
- SubjectMapper 用于定义映射subject表的接口

⑤在模块下的 src/main/resources 目录中,将 mapper 文件夹导入进去,该包下包含两个SQL文件

- TeacherMapper.xml 用于写操作teacher表的SQL语句
- SubjectMapper.xml 用于写操作subject表的SQL语句

⑥配置文件 application.yml 内容

```
1  #数据库链接
2  spring:
3      datasource:
4          url: jdbc:mysql://localhost:3306/tedu?
serverTimezone=Asia/Shanghai&characterEncoding=utf8
5          username: root
6          password: root
7  #MyBatis开启驼峰映射,并且扫描xml文件
8  mybatis:
9      configuration:
10         map-underscore-to-camel-case: true
11         mapper-teacher: classpath:/mapper/*.xml
12
13  #开启日志设置
14  logging:
15      level:
16          cn:
17              tedu: debug
```

## 3.2 查询teacher表中的记录

### ① TeacherMapper

```
1 package cn.tedu.mapper;
2
3 import cn.tedu.pojo.Teacher;
4 import org.apache.ibatis.annotations.Mapper;
5
6 //指定这是一个操作数据库的mapper
7 @Mapper
8 public interface TeacherMapper {
9     public Teacher getTeacherById(Integer id);
10 }
```

### ② TeacherMapper.xml

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5 <mapper namespace="cn.tedu.mapper.TeacherMapper">
6     <select id="getTeacherById" resultType="cn.tedu.pojo.Teacher">
7         SELECT * FROM Teacher WHERE id = #{id}
8     </select>
9 </mapper>
```

### ③ TestResultMap

```
1 @SpringBootTest
2 class TestResultMap {
3     @Autowired
4     private TeacherMapper teacherMapper;
5
6     @Test
7     public void testGetTeacherById(){
8         Teacher teacher = teacherMapper.getTeacherById(1);
9         System.out.println(teacher);
10     }
11 }
```

## 3.3 使用resultMap

- resultType配置之后,开启自动映射,我们需要在 application.yml 中开启驼峰映射,实现自动匹配

```
1 mybatis:
2   configuration:
3     map-underscore-to-camel-case: true
```

- 而resultMap可以手动的将属性和表字段匹配

### ① TeacherMapper.xml

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5 <mapper namespace="cn.tedu.mapper.TeacherMapper">
6     <select id="getTeacherById" resultMap="teacher">
7         SELECT *
8         FROM Teacher
9         WHERE id = #{id}
```

```

10     </select>
11     <!--resultMap自定义javaBean映射规则
12         type: 自定义规则的Java类型
13         id:唯一id,方便引用
14         建议只要定义resultMap,就将整表的全部列的映射全部写上
15     -->
16     <resultMap id="teacher" type="cn.tedu.pojo.Teacher">
17         <!--指定主键列的封装规则
18             id定义主键,底层会有优化
19             column:指定数据库表的哪一列
20             property:指定javaBean的哪一属性
21         -->
22         <id column="id" property="id"/>
23         <!--指定普通列的封装规则-->
24         <result column="name" property="name"/>
25         <result column="age" property="age"/>
26         <result column="title" property="title"/>
27         <result column="manager" property="manager"/>
28         <result column="salary" property="salary"/>
29         <result column="comm" property="comm"/>
30         <result column="gender" property="gender"/>
31         <result column="subject_id" property="subjectId"/>
32     </resultMap>
33 </mapper>

```

## ② application.yml

```

1  spring:
2    datasource:
3      url: jdbc:mysql://localhost:3306/tedu?
serverTimezone=Asia/Shanghai&characterEncoding=utf8
4      username: root
5      password: root
6  mybatis:
7    # configuration:
8    #   map-underscore-to-camel-case: true
9    mapper-locations: classpath:/mapper/*.xml
10
11  logging:
12    level:
13      cn:
14        tedu: debug

```

## 3.4 优化

### 3.4.1 开启包扫描

- resultType在定义时,每次总要写全对应的javaBean的全路径,很麻烦,所以可以在配置文件中添加如下的配置,开启包扫描

```

1  mybatis:
2    #指定entity扫描包类让mybatis自动扫描到自定义的包路径,这样在mapper.xml中就直接写类名即可
3    type-aliases-package: cn.tedu.pojo

```

- 那么MyBatis会自动扫描该包下的javaBean,这样我们就直接写类名即可

```

1  <select id="getTeacherById" resultType="Teacher">
2      SELECT *
3      FROM Teacher
4      WHERE id = #{id}
5  </select>

```

### 3.4.2 resultMap开启自动映射

- resultMap是可以自动映射和手动映射兼容的
- 在resultMap标签中使用autoMapping属性,如果为true就表示开启自动映射
- 但是要是开启自动映射,就需要添加驼峰规则配置

#### ① TeacherMapper.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE mapper
3      PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5  <mapper namespace="cn.tedu.mapper.TeacherMapper">
6      <select id="getTeacherById" resultMap="teacher">
7          SELECT *
8          FROM Teacher
9          WHERE id = #{id}
10     </select>
11     <!--resultMap自定义javaBean映射规则
12         type: 自定义规则的Java类型
13         id:唯一id,方便引用
14         建议只要定义resultMap,就将整表的全部列的映射全部写上
15     -->
16     <resultMap id="teacher" type="Teacher" autoMapping="true">
17         <!--指定主键列的封装规则
18             id定义主键,底层会有优化
19             column:指定数据库表的哪一列
20             property:指定javaBean的哪一属性
21         -->
22         <id column="id" property="id"/>
23         <!--指定普通列的封装规则-->
24         <result column="name" property="name"/>
25         <result column="age" property="age"/>
26         <result column="title" property="title"/>
27         <result column="manager" property="manager"/>
28         <result column="salary" property="salary"/>
29         <result column="comm" property="comm"/>
30         <result column="gender" property="gender"/>
31         <result column="subject_id" property="subjectId"/>
32     </resultMap>
33 </mapper>

```

#### ② application.yml

```

1  spring:
2      datasource:
3          url: jdbc:mysql://localhost:3306/tedu?
serverTimezone=Asia/Shanghai&characterEncoding=utf8
4          username: root
5          password: root
6  mybatis:
7      configuration:
8          map-underscore-to-camel-case: true
9      mapper-locations: classpath:/mapper/*.xml
10     type-aliases-package: cn.tedu.pojo

```

```
11 logging:
12   level:
13     cn:
14       tedu: debug
```

## 4 resultMap的级联用法

### 4.1 老师表和科目表的关系

- teacher表就是老师信息表,用于收集老师的信息
- subject表就是科目表,用于收集科目相关信息
- 这两张表具有以下关系:
  - 在科目表的角度: 一个科目对应多个老师,也就是一对多的关系
  - 在老师表的角度: 一个老师对应一个科目,也就是一对一的关系

### 4.1 一对一查询

- 查询teacher表记录的同时,将对应的subject表中的内容查询出来,SQL如下:

```
1  SELECT t.id,
2         t.name,
3         t.age,
4         t.title,
5         t.manager,
6         t.salary,
7         t.comm,
8         t.gender,
9         t.subject_id,
10        s.id,
11        s.name
12  FROM teacher t,
13        subject s
14  WHERE t.subject_id = s.id AND t.id = 1;
```

- 在这条SQL中,使用了多表关联查询,并且表字段也起了别名

#### ① Teacher

```
1  public class Teacher {
2      private Long id;
3      private String name;
4      private Long age;
5      private String title;
6      private Long manager;
7      private Long salary;
8      private Long comm;
9      private String gender;
10     private Long subjectId;
11     private Subject subject;
12
13     public Subject getSubject() {
14         return subject;
15     }
16
17     public void setSubject(Subject subject) {
18         this.subject = subject;
19     }
20
21     public Long getId() {
```

```
22         return id;
23     }
24
25     public void setId(Long id) {
26         this.id = id;
27     }
28
29
30     public String getName() {
31         return name;
32     }
33
34     public void setName(String name) {
35         this.name = name;
36     }
37
38
39     public Long getAge() {
40         return age;
41     }
42
43     public void setAge(Long age) {
44         this.age = age;
45     }
46
47
48     public String getTitle() {
49         return title;
50     }
51
52     public void setTitle(String title) {
53         this.title = title;
54     }
55
56
57     public Long getManager() {
58         return manager;
59     }
60
61     public void setManager(Long manager) {
62         this.manager = manager;
63     }
64
65
66     public Long getSalary() {
67         return salary;
68     }
69
70     public void setSalary(Long salary) {
71         this.salary = salary;
72     }
73
74
75     public Long getComm() {
76         return comm;
77     }
78
79     public void setComm(Long comm) {
80         this.comm = comm;
81     }
82
83
84     public String getGender() {
```

```

85         return gender;
86     }
87
88     public void setGender(String gender) {
89         this.gender = gender;
90     }
91
92
93     public Long getSubjectId() {
94         return subjectId;
95     }
96
97     public void setSubjectId(Long subjectId) {
98         this.subjectId = subjectId;
99     }
100
101     @Override
102     public String toString() {
103         return "Teacher{" +
104             "id=" + id +
105             ", name='" + name + '\'' +
106             ", age=" + age +
107             ", title='" + title + '\'' +
108             ", manager=" + manager +
109             ", salary=" + salary +
110             ", comm=" + comm +
111             ", gender='" + gender + '\'' +
112             ", subjectId=" + subjectId +
113             ", subject=" + subject +
114             '}';
115     }
116 }

```

## ② TeacherMapper接口

```

1  @Mapper
2  public interface TeacherMapper {
3      public Teacher getTeacherById(Integer id);
4      public Teacher getTeacherSubjectById(Integer id);
5  }

```

## ③ TeacherMapper.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE mapper
3      PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5  <mapper namespace="cn.tedu.mapper.TeacherMapper">
6      <select id="getTeacherById" resultMap="teacher">
7          SELECT *
8          FROM Teacher
9          WHERE id = #{id}
10     </select>
11     <select id="getTeacherSubjectById" resultMap="teacher2">
12         SELECT t.id,
13             t.name,
14             t.age,
15             t.title,
16             t.manager,
17             t.salary,
18             t.comm,
19             t.gender,
20             t.subject_id,

```



```

21         s.id sid,
22         s.name sname
23     FROM teacher t,
24         subject s
25     WHERE t.subject_id = s.id
26           AND t.id = #{id};
27 </select>
28 <!--resultMap自定义javaBean映射规则
29     type: 自定义规则的Java类型
30     id:唯一id,方便引用
31     建议只要定义resultMap,就将整表的全部列的映射全部写上
32 -->
33 <resultMap id="teacher" type="Teacher" autoMapping="true">
34     <!--指定主键列的封装规则
35         id定义主键,底层会有优化
36         column:指定数据库表的哪一列
37         property:指定javaBean的哪一属性
38 -->
39     <id column="id" property="id"/>
40     <!--指定普通列的封装规则-->
41     <result column="name" property="name"/>
42     <result column="age" property="age"/>
43     <result column="title" property="title"/>
44     <result column="manager" property="manager"/>
45     <result column="salary" property="salary"/>
46     <result column="comm" property="comm"/>
47     <result column="gender" property="gender"/>
48     <result column="subject_id" property="subjectId"/>
49 </resultMap>
50 <!--
51     联合查询: 使用级联属性封装结果集
52 -->
53 <resultMap id="teacher2" type="Teacher" autoMapping="true">
54     <!--指定主键列的封装规则
55         id定义主键,底层会有优化
56         column:指定数据库表的哪一列
57         property:指定javaBean的哪一属性
58 -->
59     <id column="id" property="id"/>
60     <!--指定普通列的封装规则-->
61     <result column="name" property="name"/>
62     <result column="age" property="age"/>
63     <result column="title" property="title"/>
64     <result column="manager" property="manager"/>
65     <result column="salary" property="salary"/>
66     <result column="comm" property="comm"/>
67     <result column="gender" property="gender"/>
68     <result column="subject_id" property="subjectId"/>
69     <result column="sid" property="subject.id"/>
70     <result column="sname" property="subject.name"/>
71 </resultMap>
72 </mapper>

```

#### ④ TestResultMap

```

1 package cn.tedu;
2
3 import cn.tedu.mapper.TeacherMapper;
4 import cn.tedu.pojo.Locations;
5 import cn.tedu.pojo.Teacher;
6 import org.junit.jupiter.api.Test;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.boot.test.context.SpringBootTest;

```

```

9
10 @SpringBootTest
11 class TestResultMap {
12     @Autowired
13     private TeacherMapper teacherMapper;
14
15     @Test
16     public void testGetTeacherById(){
17         Teacher teacher = teacherMapper.getTeacherById(1);
18         System.out.println(teacher);
19     }
20     @Test
21     public void getTeacherSubjectById() {
22         Teacher teacher = teacherMapper.getTeacherSubjectById(1);
23         System.out.println(teacher);
24         System.out.println(teacher.getSubject());
25     }
26 }

```

## 4.2 association定义一对一关系映射

- 上述的案例中,如果像类似于 `<result column="sid" property="subject.id"/>`,这样的字段比较多的情况,每次关联属性都要写subject的级联属性的方式封装结果集,会比较繁琐,重复字段较多,其次,看不出关联关系,所以可以使用 `association` 标签进行封装
- association标签: 可以指定联合的javaBean对象
  - property属性: 指定哪个属性是联合的对象
  - javaType属性: 指定这个属性对象的类型[不能省略]

### ① TeacherMapper.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE mapper
3      PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5  <mapper namespace="cn.tedu.mapper.LocationsMapper">
6      <select id="getLocationById" resultMap="location">
7          SELECT * FROM locations WHERE location_id = #{locationId}
8      </select>
9      <select id="getLocationCountryById" resultMap="location3">
10         SELECT l.location_id,l.street_address,l.postal_code,
11             l.city,l.state_province,l.country_id lcid,
12             c.country_id cid,c.country_name,c.region_id
13         FROM locations l,countries c
14         WHERE l.country_id = c.country_id AND l.location_id = #{locationId}
15     </select>
16     <resultMap id="location" type="Locations">...</resultMap>
17     <!--
18     联合查询: 使用级联属性封装结果集
19     -->
20     <resultMap id="location2" type="Locations">...</resultMap>
21     <!--association属性定义单个对象封装规则-->
22     <resultMap id="location3" type="Locations">
23         <id column="location_id" property="locationId"/>
24         <result column="street_address" property="streetAddress"/>
25         <result column="postal_code" property="postalCode"/>
26         <result column="city" property="city"/>
27         <result column="state_province" property="stateProvince"/>
28         <result column="lcid" property="countryId"/>
29         <!--association可以指定联合的javaBean对象
30         property="countries" 指定哪个属性是联合的对象

```

```

31         javaType="Countries" 指定这个属性对象的类型[不能省略]-->
32         <association property="countries" javaType="Countries"
    autoMapping="true">
33             <id column="cid" property="countryId"/>
34             <result column="country_name" property="countryName"/>
35             <result column="region_id" property="regionId"/>
36         </association>
37     </resultMap>
38 </mapper>

```

## 4.3 collection定义一对多关系映射

- 查询教指定科目的老师信息,SQL如下:

```

1  SELECT s.id,
2         s.name,
3         t.id   tid,
4         t.name tname,
5         t.age,
6         t.title,
7         t.manager,
8         t.salary,
9         t.comm,
10        t.gender,
11        t.subject_id
12 FROM subject s
13      LEFT JOIN teacher t ON s.id = t.subject_id
14 WHERE s.name = '语文';

```

- collection标签: 可以指定联合的集合类型的属性
  - property属性: 指定哪个属性是联合的对象
  - ofType属性: 指定集合的元素类型[不能省略]

### ① Subject

```

1  package cn.tedu.pojo;
2
3
4  import java.util.List;
5
6  public class Subject {
7      private Long id;
8      private String name;
9      private List<Teacher> teachers;
10
11      public List<Teacher> getTeachers() {
12          return teachers;
13      }
14
15      public void setTeachers(List<Teacher> teachers) {
16          this.teachers = teachers;
17      }
18
19      @Override
20      public String toString() {
21          return "Subject{" +
22              "id=" + id +
23              ", name='" + name + '\'' +
24              ", teachers=" + teachers +
25              '}';
26      }
27  }

```

```

27
28     public Long getId() {
29         return id;
30     }
31
32     public void setId(Long id) {
33         this.id = id;
34     }
35
36
37     public String getName() {
38         return name;
39     }
40
41     public void setName(String name) {
42         this.name = name;
43     }
44 }

```

## ② SubjectMapper接口

```

1  package cn.tedu.mapper;
2
3  import cn.tedu.pojo.Subject;
4  import org.apache.ibatis.annotations.Mapper;
5
6  //指定这是一个操作数据库的mapper
7  @Mapper
8  public interface SubjectMapper {
9      public Subject getSubjectTeacherByName(String name);
10 }

```

## ③ SubjectMapper.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE mapper
3      PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5  <mapper namespace="cn.tedu.mapper.SubjectMapper">
6      <select id="getSubjectTeacherByName" resultMap="subject">
7          SELECT s.id,
8                 s.name,
9                 t.id   tid,
10                t.name tname,
11                t.age,
12                t.title,
13                t.manager,
14                t.salary,
15                t.comm,
16                t.gender,
17                t.subject_id
18          FROM subject s
19                LEFT JOIN teacher t ON s.id = t.subject_id
20          WHERE s.name = #{name};
21      </select>
22      <resultMap id="subject" type="Subject">
23          <id column="id" property="id"/>
24          <result column="name" property="name"/>
25          <!--定义关联集合类型的属性的封装规则
26              ofType 指定集合的元素类型
27          -->
28          <collection property="teachers" ofType="Teacher">
29              <!--定义集合中的元素的封装规则-->

```

```

30         <id column="tid" property="id"/>
31         <result column="tname" property="name"/>
32         <result column="age" property="age"/>
33         <result column="title" property="title"/>
34         <result column="manager" property="manager"/>
35         <result column="salary" property="salary"/>
36         <result column="comm" property="comm"/>
37         <result column="gender" property="gender"/>
38         <result column="subject_id" property="subjectId"/>
39     </collection>
40 </resultMap>
41 </mapper>

```

#### ④ TestResultMap

```

1  package cn.tedu;
2
3  import cn.tedu.mapper.SubjectMapper;
4  import cn.tedu.mapper.TeacherMapper;
5  import cn.tedu.pojo.Locations;
6  import cn.tedu.pojo.Subject;
7  import cn.tedu.pojo.Teacher;
8  import org.junit.jupiter.api.Test;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.boot.test.context.SpringBootTest;
11
12 import java.util.List;
13
14 @SpringBootTest
15 class TestResultMap {
16     @Autowired
17     private TeacherMapper teacherMapper;
18     @Autowired
19     private SubjectMapper subjectMapper;
20
21     @Test
22     public void testGetTeacherById() {
23         Teacher teacher = teacherMapper.getTeacherById(1);
24         System.out.println(teacher);
25     }
26
27     @Test
28     public void getTeacherSubjectById() {
29         Teacher teacher = teacherMapper.getTeacherSubjectById(2);
30         System.out.println(teacher);
31         System.out.println(teacher.getSubject());
32     }
33
34     @Test
35     public void testGetSubjectTeacherByName() {
36         Subject subject = subjectMapper.getSubjectTeacherByName("语文");
37         System.out.println(subject);
38         List<Teacher> teachers = subject.getTeachers();
39         for (Teacher teacher : teachers) {
40             System.out.println(teacher);
41         }
42     }
43 }

```





































































