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 resultType和resultMap

2.1 resultType--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

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

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

```xml
<select id="getTeacherById" resultType="cn.tedu.pojo.Teacher">
    SELECT id, name, age, title, manager, salary, comm, gender, subject_id
    FROM teacher
    WHERE id = #{id}
</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

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

- select标签的 `resultType` 属性的值是POJO的全限定名

```xml
<select id="getTeacherAll" resultType="cn.tedu.pojo.Teacher">
    SELECT id, name, age, title, manager, salary, comm, gender, subject_id
    FROM teacher
</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标签
    - 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

```xml
<!--mysql数据库驱动依赖-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
<!--引入相关mybatis依赖-->
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>2.2.0</version>
</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

```yaml
#数据库链接
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/tedu?serverTimezone=Asia/Shanghai&characterEncoding=utf8
    username: root
    password: root
#MyBatis开启驼峰映射,并且扫描xml文件
mybatis:
  configuration:
    map-underscore-to-camel-case: true
  mapper-teacher: classpath:/mapper/*.xml

#开启日志设置
logging:
  level:
    cn:
      tedu: debug
```

3.2 查询teacher表中的记录

① `TeacherMapper`

```java
package cn.tedu.mapper;

import cn.tedu.pojo.Teacher;
import org.apache.ibatis.annotations.Mapper;

//指定这是一个操作数据库的mapper
@Mapper
public interface TeacherMapper {
    public Teacher getTeacherById(Integer id);
}
```

② `TeacherMapper.xml`

```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="cn.tedu.mapper.TeacherMapper">
    <select id="getTeacherById" resultType="cn.tedu.pojo.Teacher">
        SELECT * FROM Teacher WHERE id = #{id}
    </select>
</mapper>
```

③ `TestResultMap`

```java
@SpringBootTest
class TestResultMap {
    @Autowired
    private TeacherMapper teacherMapper;

    @Test
    public void testGetTeacherById(){
        Teacher teacher = teacherMapper.getTeacherById(1);
        System.out.println(teacher);
    }
}
```

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

```yaml
mybatis:
  configuration:
    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`

```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="cn.tedu.mapper.TeacherMapper">
    <select id="getTeacherById" resultMap="teacher">
        SELECT *
        FROM Teacher
        WHERE id = #{id}
    </select>
    <!--resultMap自定义javaBean映射规则
        type: 自定义规则的Java类型
        id:唯一id,方便引用
        建议只要定义resultMap,就将整表的全部列的映射全部写上
    -->
    <resultMap id="teacher" type="cn.tedu.pojo.Teacher">
        <!--指定主键列的封装规则
            id定义主键,底层会有优化
            column:指定数据库表的哪一列
            property:指定javaBean的哪一属性
        -->
        <id column="id" property="id"/>
        <!--指定普通列的封装规则-->
        <result column="name" property="name"/>
        <result column="age" property="age"/>
        <result column="title" property="title"/>
        <result column="manager" property="manager"/>
        <result column="salary" property="salary"/>
        <result column="comm" property="comm"/>
        <result column="gender" property="gender"/>
        <result column="subject_id" property="subjectId"/>
    </resultMap>
</mapper>
```

② `application.yml`

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/tedu?serverTimezone=Asia/Shanghai&characterEncoding=utf8
    username: root
    password: root
mybatis:
#  configuration:
#    map-underscore-to-camel-case: true
  mapper-locations: classpath:/mapper/*.xml

logging:
  level:
    cn:
      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

```
mybatis:
  #指定entity扫描包类让mybatis自定扫描到自定义的包路径,这样在mapper.xml中就直接写类名即可
  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

```xml
<select id="getTeacherById" resultType="Teacher">
    SELECT *
    FROM Teacher
    WHERE id = #{id}
</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`

```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="cn.tedu.mapper.TeacherMapper">
    <select id="getTeacherById" resultMap="teacher">
        SELECT *
        FROM Teacher
        WHERE id = #{id}
    </select>
    <!--resultMap自定义javaBean映射规则
        type: 自定义规则的Java类型
        id:唯一id,方便引用
        建议只要定义resultMap,就将整表的全部列的映射全部写上
    -->
    <resultMap id="teacher" type="Teacher" autoMapping="true">
        <!--指定主键列的封装规则
            id定义主键,底层会有优化
            column:指定数据库表的哪一列
            property:指定javaBean的哪一属性
        -->
        <id column="id" property="id"/>
        <!--指定普通列的封装规则-->
        <result column="name" property="name"/>
        <result column="age" property="age"/>
        <result column="title" property="title"/>
        <result column="manager" property="manager"/>
        <result column="salary" property="salary"/>
        <result column="comm" property="comm"/>
        <result column="gender" property="gender"/>
        <result column="subject_id" property="subjectId"/>
    </resultMap>
</mapper>
```

② `application.yml`

```yaml
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/tedu?
serverTimezone=Asia/Shanghai&characterEncoding=utf8
    username: root
    password: root
mybatis:
  configuration:
    map-underscore-to-camel-case: true
  mapper-locations: classpath:/mapper/*.xml
  type-aliases-package: cn.tedu.pojo
```

```yaml
logging:
  level:
    cn:
      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:

```sql
SELECT t.id,
       t.name,
       t.age,
       t.title,
       t.manager,
       t.salary,
       t.comm,
       t.gender,
       t.subject_id,
       s.id,
       s.name
FROM teacher t,
     subject s
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`

```java
public class Teacher {
    private Long id;
    private String name;
    private Long age;
    private String title;
    private Long manager;
    private Long salary;
    private Long comm;
    private String gender;
    private Long subjectId;
    private Subject subject;

    public Subject getSubject() {
        return subject;
    }

    public void setSubject(Subject subject) {
        this.subject = subject;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }


    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }


    public Long getAge() {
        return age;
    }

    public void setAge(Long age) {
        this.age = age;
    }


    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }


    public Long getManager() {
        return manager;
    }

    public void setManager(Long manager) {
        this.manager = manager;
    }


    public Long getSalary() {
        return salary;
    }

    public void setSalary(Long salary) {
        this.salary = salary;
    }


    public Long getComm() {
        return comm;
    }

    public void setComm(Long comm) {
        this.comm = comm;
    }


    public String getGender() {
            return gender;
        }

        public void setGender(String gender) {
            this.gender = gender;
        }


        public Long getSubjectId() {
            return subjectId;
        }

        public void setSubjectId(Long subjectId) {
            this.subjectId = subjectId;
        }

        @Override
        public String toString() {
            return "Teacher{" +
                    "id=" + id +
                    ", name='" + name + '\'' +
                    ", age=" + age +
                    ", title='" + title + '\'' +
                    ", manager=" + manager +
                    ", salary=" + salary +
                    ", comm=" + comm +
                    ", gender='" + gender + '\'' +
                    ", subjectId=" + subjectId +
                    ", subject=" + subject +
                    '}';
        }
    }
```

② `TeacherMapper接口`

```java
@Mapper
public interface TeacherMapper {
    public Teacher getTeacherById(Integer id);
    public Teacher getTeacherSubjectById(Integer id);
}
```

③ `TeacherMapper.xml`

```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="cn.tedu.mapper.TeacherMapper">
    <select id="getTeacherById" resultMap="teacher">
        SELECT *
        FROM Teacher
        WHERE id = #{id}
    </select>
    <select id="getTeacherSubjectById" resultMap="teacher2">
        SELECT t.id,
               t.name,
               t.age,
               t.title,
               t.manager,
               t.salary,
               t.comm,
               t.gender,
               t.subject_id,
                s.id sid,
                s.name sname
        FROM teacher t,
             subject s
        WHERE t.subject_id = s.id
          AND t.id = #{id};
    </select>
    <!--resultMap自定义javaBean映射规则
        type: 自定义规则的Java类型
        id:唯一id,方便引用
        建议只要定义resultMap,就将整表的全部列的映射全部写上
    -->
    <resultMap id="teacher" type="Teacher" autoMapping="true">
        <!--指定主键列的封装规则
            id定义主键,底层会有优化
            column:指定数据库表的哪一列
            property:指定javaBean的哪一属性
        -->
        <id column="id" property="id"/>
        <!--指定普通列的封装规则-->
        <result column="name" property="name"/>
        <result column="age" property="age"/>
        <result column="title" property="title"/>
        <result column="manager" property="manager"/>
        <result column="salary" property="salary"/>
        <result column="comm" property="comm"/>
        <result column="gender" property="gender"/>
        <result column="subject_id" property="subjectId"/>
    </resultMap>
    <!--
        联合查询: 使用级联属性封装结果集
    -->
    <resultMap id="teacher2" type="Teacher" autoMapping="true">
        <!--指定主键列的封装规则
            id定义主键,底层会有优化
            column:指定数据库表的哪一列
            property:指定javaBean的哪一属性
        -->
        <id column="id" property="id"/>
        <!--指定普通列的封装规则-->
        <result column="name" property="name"/>
        <result column="age" property="age"/>
        <result column="title" property="title"/>
        <result column="manager" property="manager"/>
        <result column="salary" property="salary"/>
        <result column="comm" property="comm"/>
        <result column="gender" property="gender"/>
        <result column="subject_id" property="subjectId"/>
        <result column="sid" property="subject.id"/>
        <result column="sname" property="subject.name"/>
    </resultMap>
</mapper>
```

④ `TestResultMap`

```
package cn.tedu;

import cn.tedu.mapper.TeacherMapper;
import cn.tedu.pojo.Locations;
import cn.tedu.pojo.Teacher;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
```

```java

@SpringBootTest
class TestResultMap {
    @Autowired
    private TeacherMapper teacherMapper;

    @Test
    public void testGetTeacherById(){
        Teacher teacher = teacherMapper.getTeacherById(1);
        System.out.println(teacher);
    }
    @Test
    public void getTeacherSubjectById() {
        Teacher teacher = teacherMapper.getTeacherSubjectById(1);
        System.out.println(teacher);
        System.out.println(teacher.getSubject());
    }
}
```

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`

```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="cn.tedu.mapper.LocationsMapper">
    <select id="getLocationById" resultMap="location">
        SELECT * FROM locations WHERE location_id = #{locationId}
    </select>
    <select id="getLocationCountryById" resultMap="location3">
        SELECT l.location_id,l.street_address,l.postal_code,
               l.city,l.state_province,l.country_id lcid,
               c.country_id cid,c.country_name,c.region_id
        FROM locations l,countries c
        WHERE l.country_id = c.country_id AND l.location_id = #{locationId}
    </select>
    <resultMap id="location" type="Locations">...</resultMap>
    <!--
    联合查询: 使用级联属性封装结果集
    -->
    <resultMap id="location2" type="Locations">...</resultMap>
    <!--association属性定义单个对象封装规则-->
    <resultMap id="location3" type="Locations">
        <id column="location_id" property="locationId"/>
        <result column="street_address" property="streetAddress"/>
        <result column="postal_code" property="postalCode"/>
        <result column="city" property="city"/>
        <result column="state_province" property="stateProvince"/>
        <result column="lcid" property="countryId"/>
        <!--association可以指定联合的javaBean对象
        property="countries" 指定哪个属性是联合的对象
        javaType="Countries" 指定这个属性对象的类型[不能省略]-->
        <association property="countries" javaType="Countries"
autoMapping="true">
            <id column="cid" property="countryId"/>
            <result column="country_name" property="countryName"/>
            <result column="region_id" property="regionId"/>
        </association>
    </resultMap>
</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:

```
SELECT s.id,
       s.name,
       t.id   tid,
       t.name tname,
       t.age,
       t.title,
       t.manager,
       t.salary,
       t.comm,
       t.gender,
       t.subject_id
FROM subject s
         LEFT JOIN teacher t ON s.id = t.subject_id
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`

```
package cn.tedu.pojo;


import java.util.List;

public class Subject {
    private Long id;
    private String name;
    private List<Teacher> teachers;

    public List<Teacher> getTeachers() {
        return teachers;
    }

    public void setTeachers(List<Teacher> teachers) {
        this.teachers = teachers;
    }

    @Override
    public String toString() {
        return "Subject{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", teachers=" + teachers +
                '}';
    }
```

```java

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }


    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

② `SubjectMapper接口`

```java
package cn.tedu.mapper;

import cn.tedu.pojo.Subject;
import org.apache.ibatis.annotations.Mapper;

//指定这是一个操作数据库的mapper
@Mapper
public interface SubjectMapper {
    public Subject getSubjectTeacherByName(String name);
}
```

③ `SubjectMapper.xml`

```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="cn.tedu.mapper.SubjectMapper">
    <select id="getSubjectTeacherByName" resultMap="subject">
        SELECT s.id,
               s.name,
               t.id   tid,
               t.name tname,
               t.age,
               t.title,
               t.manager,
               t.salary,
               t.comm,
               t.gender,
               t.subject_id
        FROM subject s
                 LEFT JOIN teacher t ON s.id = t.subject_id
        WHERE s.name = #{name};
    </select>
    <resultMap id="subject" type="Subject">
        <id column="id" property="id"/>
        <result column="name" property="name"/>
        <!--定义关联集合类型的属性的封装规则
            ofType 指定集合的元素类型
        -->
        <collection property="teachers" ofType="Teacher">
            <!--定义集合中的元素的封装规则-->
```

```xml
                <id column="tid" property="id"/>
                <result column="tname" property="name"/>
                <result column="age" property="age"/>
                <result column="title" property="title"/>
                <result column="manager" property="manager"/>
                <result column="salary" property="salary"/>
                <result column="comm" property="comm"/>
                <result column="gender" property="gender"/>
                <result column="subject_id" property="subjectId"/>
            </collection>
        </resultMap>
    </mapper>
```

④ `TestResultMap`

```java
package cn.tedu;

import cn.tedu.mapper.SubjectMapper;
import cn.tedu.mapper.TeacherMapper;
import cn.tedu.pojo.Locations;
import cn.tedu.pojo.Subject;
import cn.tedu.pojo.Teacher;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import java.util.List;

@SpringBootTest
class TestResultMap {
    @Autowired
    private TeacherMapper teacherMapper;
    @Autowired
    private SubjectMapper subjectMapper;

    @Test
    public void testGetTeacherById() {
        Teacher teacher = teacherMapper.getTeacherById(1);
        System.out.println(teacher);
    }

    @Test
    public void getTeacherSubjectById() {
        Teacher teacher = teacherMapper.getTeacherSubjectById(2);
        System.out.println(teacher);
        System.out.println(teacher.getSubject());
    }

    @Test
    public void testGetSubjectTeacherByName() {
        Subject subject = subjectMapper.getSubjectTeacherByName("语文");
        System.out.println(subject);
        List<Teacher> teachers = subject.getTeachers();
        for (Teacher teacher : teachers) {
            System.out.println(teacher);
        }
    }
}
```