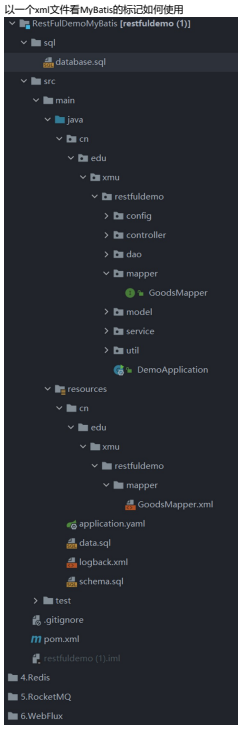
Mapper接口和XML配置

2021年10月4日 20:34

示例项目：RestfulDemoMyBatis

以一个xml文件看MyBatis的标记如何使用



MyBatis的xml文件位于项目的resources目录下，在resources目录下我们建了和mapper接口文件相同的目录，因为这个xml在最后打包的时候是要和mapper接口文件打包到一起的

GoodsMapper.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.edu.xmu.restfuldemo.mapper.GoodsMapper">
    <sql id="Goods_Column_List">
        id,
        goods_sn,
        name,
        category_id,
        brand_id,
        brief,
        spec_list,
        pic_url,
        unit,
        state,
        add_time,
        update_time,
        modi_user
    </sql>

    <sql id="Goods_Insert_List">
        goods_sn,
        name,
        category_id,
        brand_id,
        brief,
        spec_list,
        pic_url,
        unit,
        state,
        add_time
    </sql>

    <sql id="Product_Insert_List">
        goods_id,
        product_sn,
        name,
        original_price,
        stock,
        weight,
        add_time,
        state
    </sql>

    <resultMap type="GoodsPo" id="GoodsProductResultMap" autoMapping="true">
        <id property="id" column="id"/>
        <collection property="productList" ofType="cn.edu.xmu.restfuldemo.model.ProductPo" autoMapping="true">
            <id property="id" column="product_id"/>
            <result property="name" column="product_name"/>
            <result property="state" column="product_state"/>
            <result property="addTime" column="product_add_time"/>
            <result property="updateTime" column="product_update_time"/>
        </collection>
    </resultMap>

    <select id="findGoods" parameterType="GoodsPo" resultType="GoodsPo">
        SELECT
        <include refid="Goods_Column_List"/>
        FROM oomall_goods
        WHERE
        state != 2
        <if test="id!=null">and id = #{id} </if>
        <if test="goodsSn!=null and goodsSn!=''">and goods_sn = #{goodsSn} </if>
        <if test="name!=null and name!=''">and name = #{name} </if>
        <if test="categoryId!=null">and category_id = #{categoryId} </if>
        <if test="brandId!=null">and brand_id = #{brandId} </if>
        <if test="state!=null">and state = #{state} </if>
    </select>

    <select id="findGoodsWithProduct" parameterType="GoodsPo" resultMap="GoodsProductResultMap">
        SELECT
        g.id as id,
        goods_sn,
        g.name as name,
        category_id,
        brand_id,
```

resultMap标记 id定义了这个resultMap的名字 type定义最后要返回什么类型

定义这个resultMap的原因：

GoodsPo是一个有关联的对象，其中productList是关联规格对象的

而goods和product分别对应两张表

结合下面的findGoodsWithProduct，我们可以看出定义这个resultMap是为了查询两张表，然后形成一个商品对象中带了多个关联对象的一个对象，再返回回来

因为type定义了要返回的是GoodsPo，所以后面的property都是GoodsPo中的属性以及Product中的property

autoMapping是自动对应，能够自动映射的属性和字段就会自动映射，不用autoMapping就要一个一个写对应关系。不能自动映射的就需要我们手动定义

第一个，id，规定column的id对应property的id，这是主键

collection标签，因为GoodsPo中有一个属性是List类型，其中每一个都是ProductPo

所以后面的ofType定义了每一个元素的类型，要把整个路径写全，后面同样使用了autoMapping

下面的id和result是不能autoMapping的，所以手动写

这样就定义了使用resultMap作为返回值的不是简单的返回一个GoodsPo对象，而是返回一个带关联的、里面有ProductList的GoodsPo

```java
/**
 */
@Data
@Alias("GoodsPo")
public class GoodsPo {

    private Integer id;

    private String goodsSn;

    private String name;

    private Integer categoryId;

    private Integer brandId;

    private String brief;

    private String picUrl;

    private String unit;

    private Integer state;

    private String specList;

    private LocalDateTime addTime;

    private LocalDateTime updateTime;

    private Integer modiUser;

    private List<ProductPo> productList;
}
```

select标记，用于从数据库中查询数据，返回给对象模型

id属性，和上面的<mapper namespace>一起

定位到Mapper接口中的findGoods方法

parameterType定义传过来的参数是什么样的

因为我们传参基本都是传对象

如果想传多个参数，又不想用对象传，就定义parameterMap

resultType定义返回类型，我们返回是直接返回一个对象，所以直接用resultType

中间写的是SELECT语句

使用了include标签，引用的是Goods_Column_List，可以在上面找到对应的sql标记

下面的<if>是动态sql语句中的标记，用来对参数进行判断

这里是如果id、goodsSn、name、category、brandId、state等不会空的话，如果不为空，将其加入SQL语句作为查询条件

其中的#{id}这样的标记表示这个参数应该来自于parameter，这里表示这个id应该来自于GoodsPo的一个属性

（使用 # 比使用 $ 更为安全，$会产生注入问题）

```yaml
mybatis:
  #domain对象的位
  type-aliases-package: cn.edu.xmu.restfuldemo.model
  #mapper.xml所在的位置
  mapper-locations: classpath:cn.edu.xmu.restfuldemo.mapper/*.xml
```

```xml
    SELECT
    g.id as id,
    goods_sn,
    g.name as name,
    category_id,
    brand_id,
    brief,
    spec_list,
    pic_url,
    unit,
    g.state as state,
    g.add_time as add_time,
    g.update_time as update_time,
    modi_user,
    p.id as product_id,
    product_sn,
    p.name as product_name,
    original_price,
    stock,
    weight,
    p.add_time as product_add_time,
    p.update_time as product_update_time,
    p.state as product_state
    FROM oomall_goods g LEFT JOIN oomall_product p
    ON g.id = p.goods_id
    WHERE
    g.state != 2
    <if test="id!=null">and g.id = #{id} </if>
    <if test="goodsSn!=null and goodsSn!=''">and goods_sn = #{goodsSn} </if>
    <if test="name!=null and name!=''">and g.name = #{name} </if>
    <if test="categoryId!=null">and category_id = #{categoryId} </if>
    <if test="brandId!=null">and brand_id = #{brandId} </if>
    <if test="state!=null">and g.state = #{state} </if>
</select>

<select id="findProduct" parameterType="ProductPo" resultType="ProductPo">
    SELECT
    id,
    goods_id,
    product_sn,
    name,
    original_price,
    stock,
    weight,
    add_time,
    update_time,
    state
    FROM oomall_product
    WHERE
    state != 2
    <if test="id!=null">AND p.id = #{id} </if>
    <if test="productSn!=null and productSn!=''">AND product_sn = #{productSn} </if>
    <if test="goodsId!=null and goodsId!=''">AND goods_id = #{goodsId} </if>
    <if test="state!=null and state!=''">AND state = #{state} </if>
</select>

<select id="findEffectPrice" parameterType="Integer" resultType="PriceStockPo">
SELECT
id,
price,
quantity,
begin_time,
end_time
FROM oomall_price_stock
WHERE product_id =#{id}
AND end_time >= now()
AND begin_time <= now()
</select>

<insert id="createProduct" parameterType="ProductPo" keyProperty="id" useGeneratedKeys="true">
    INSERT INTO oomall_product(<include refid="Product_Insert_List"/>)
    VALUES(#{goodsId},#{productSn},#{name},#{originalPrice}, #{stock}, #{weight}, CURRENT_TIMESTAMP, 0)
</insert>

<insert id="createGoods" parameterType="GoodsPo" keyProperty="id" useGeneratedKeys="true">
    INSERT INTO oomall_goods(<include refid="Goods_Insert_List"/>)
    VALUES(#{goodsSn},#{name},#{categoryId},#{brandId}, #{brief},#{specList},#{picUrl},#{unit},0,CURRENT_TIMESTAMP)
</insert>

<update id="updateGoods" parameterType="GoodsPo">
    UPDATE oomall_goods
    <set>
        <if test="name!=null and name!=''">name = #{name}, </if>
        <if test="categoryId!=null">category_id = #{categoryId}, </if>
        <if test="brandId!=null">brand_id = #{brandId}, </if>
        <if test="brief!=null and brief!=''">brief = #{brief}, </if>
        <if test="specList!=null and specList!=''">spec_list = #{specList}, </if>
        <if test="picUrl!=null and picUrl!=''">pic_url = #{picUrl}, </if>
        <if test="unit!=null and unit!=''">unit = #{unit}, </if>
        <if test="state!=null">state = #{state}, </if>
        update_time = CURRENT_TIME,
        modi_user = #{modiUser}
    </set>
    WHERE id = #{id}
</update>

<update id="deleteGoods" parameterType="Integer">
    UPDATE oomall_goods
    SET state = 2
    WHERE id = #{id}
</update>

<update id="deleteProductByGoodsId" parameterType="Integer">
    UPDATE oomall_product
    SET state = 2
    WHERE goods_id = #{id}
</update>

<update id="deleteProduct" parameterType="Integer">
    UPDATE oomall_product
    SET state = 2
    WHERE id = #{id}
</update>
</mapper>
```

的话，如果不为空，将其加入SQL语句作为查询条件

其中的#{id}这样的标记表示这个参数应该来自于parameter，这里表示这个id应该来自于GoodsPo的一个属性
（使用 # 比使用 $ 更为安全，$会产生注入问题）

最后要注意，对象中的属性是命名驼峰，数据库中的字段名是下划线，我们在application.yaml中定义了这两种命名风格的映射

```yaml
mybatis:
  #domain对象的包
  type-aliases-package: cn.edu.xmu.restfuldemo.model
  #mapper.xml所在的位置
  mapper-locations: classpath:cn.edu.xmu.restfuldemo.mapper/*.xml
  #自动将SQL中查出来的带下划线的字段，转换为驼峰标志，再去匹配类中的属性
  configuration:
    #输出产生的SQL
    #log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
    map-underscore-to-camel-case: true
```

这个select是使用了resultMap的

参数类型是GoodsPo，返回值不是简单的某个对象，而是resultMap中定义的对象
但这里最后还是会返回一个GoodsPo类型的对象，因为resultMap中定义了type

SELECT语句最后会产生多条记录，因为这是一个多表查询，最后做了左连接
我们使用resultMap做了一个一对多的映射关系，把查询到的多条记录最后组装成多个GoodsPo返回回去，这就是一个resultMap的一个经典使用

insert标记，向数据库中插入记录
映射到Mapper接口中的createGoods方法上，参数是GoodsPo

keyProperty和useGeneratedKeys是insert中的特别属性
因为数据库中有主键自增字段，所以这里设置id字段使用自增
下面的是很正常的insert语句
（CURRENT_TIMESTAMP是SQL一个自带的变量，用于记录当前时间）

update标记，在数据库中更新字段
映射到Mapper接口的updateGoods方法上，参数是GoodsPo上

注意这里的<set>标签
这里我们也使用<if>动态SQL，让传过来的对象中属性不为空的才更新