（namespace）命名空间对sql进行分类化管理，理解sql的隔离

使用mapper代理开发，namespace有特殊的重要作用

id：标示映射文件中的sql。将sql语句封装到mapperStatment对象中，所以将id称为statement的id。

mapper开发规范：

1. 在mapper.xml中，namespace等于mapper接口地址。mapper.xml中需要这样写：

<mapper namespace=”xxx.mapper(对应的mapper文件)”>

1. mapper.java接口中的方法名，和mapper.xml中statement的id一致。
2. mapper接口中的方法输入参数的类型和mapper.xml中statement的parameterType指定的类型一致
3. mapper.java中返回值的类型和mapper.xml中statement

例如：

mapper.java中这样写:

public User findUserById(int id) throws Exception();

mapper.xml中这样写

<select id=” findUserById” parameterType=”int” resultType=”xxxxxx.User”>

select \* from user\_info where id = #{id};

</select>

输出映射：resultMap

<resultMap type=”User” id=”userResultMap”>

<!—id表示查询结果集中唯一表示,如果有多个列组成唯一标识，则配置多个id

column表示查询出来的列名

property表示type指定的pojo类型的属性名

最终resultMap对column和property做一个映射关系（对应关系）

-->

<id column=”id\_” property=”id” />

<!—result:对普通名映射定义

column：查询出来的列名

property表示type指定的pojo类型的属性名

-->

<result column=”user\_name” property=”userName” />

</resyltMap>

使用方法：

<select id=”selectUserNameById” parameterType=”int” resultMap=” userResultMap”>//对应前面的resultMap的id。

sql片段

<sql id=”query\_user\_where”>

<if test=”userCustom!=null”>

<if test=”userCustom.sex!=null and userCustom.sex!=’’”>

and user.sex = #{userCustom.sex}

</if>

<if test=”userCustom.userName!=null and userCustom.userName!=’’”>

and user.user\_name like ‘%${userCustom.userName}%’

</if>

</if>

引用sql片段的方法

<where>

<include id=” query\_user\_where”>//id为引用sql片段的id，如果指定的id不在本mapper文件中，需要前面加namespace

</where>

动态sql遍历（foreach）

<!—

collection:指定输入对象集合

item：每个遍历生成对象

open：开始遍历时拼接的串

close：结束遍历时拼接的串

separator：遍历的两个对象中需要拼接的串

-->

<foreach collection=”ids” item=”user\_id” open=”AND (” close=”)” separator=”OR”>

id=#{user\_id}

</foreach>

高级映射：

<resultMap type=”” id=””>

<!—新增：配置映射的关联的用户信息

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

property：映射到user的哪个属性

-->

<association property=”user” javaType=”user的property”>

<id column=”user\_id” property=”id”/>

</association>

</resultMap>

resultType和resultMap的区别

实现一对一查询：

select order.\* user.id user.name,user,sex from orders,users where user.id=order.user\_id;

resultType：使用resultType实现较为简单，如果pojo中没有包括查询出来的列名，需要增加列名对应的属性，即可完成映射。

如果没有查询结果的特殊要求，建议使用resultType

resultMap：需要单独定义resultMap，实现有点麻烦，如果对查询结果有特殊要求，使用resultMap可以完成将关联的信息映射到pojo的属性中，

resultMap可以实现延迟加载

resultType无法实现延迟加载

一对多查询

在一对一查询的基础上添加

select order.\* user.id user.name,user.sex,orderdetail.item\_id,orderdetail.item\_num,

orderdetail.order\_id

from orders,users,orderdetail

where user.id=order.user\_id and orderdetail.user\_id=user.id;

<!- - 订单信息关联查询了多条明细，要使用collection进行映射

collection：对关联查询到多条记录映射到集合对象中

property：将关联查询到多条记录映射到pojo中

ofType：指定映射到集合属性中pojo类型

>

<collection property=”pojo地址” ofType=”指定到具体属性（具体类下的属性）”>

</colection>

mybatis使用resultMap的collection对关联查询的多条记录映射到一个list集合属性中。

使用resultType实现：

将结果映射到orderdetails中，需要自己处理，使用双重循环遍历，去掉重复记录，将订单明细放在orderdetails中

多对多查询

需求：查询用户买的商品明细

关联的表：用户表，订单表，明细表，商品表

映射思路：

将用户信息映射到user中。

在user类中添加订单列表属性List<Orders> orderslist，将用户创建的订单映射到orderslist

在Orders中添加订单明细列表属性List<OrderDetail>orderdetials，将订单的明细映射到orderdetials

在OrderDetail中添加Items属性，将订单明细所对应的商品映射到Items

将查询用户购买的商品信息明细清单（用户名、用户地址、购买商品名称、购买商品时间、购买商品数量）

针对上边的需求就使用resultType将查询到的记录映射到一个扩展的pojo中很简单实现明细清单的功能。

resultMap小结：

resultMap总结

resultType

作用

将查询结果按照sql列名pojo属性名一致性映射到pojo中。

场合

常见一些明细记录的展示比如用户购买商品明细将关联查询信息全部展示在页面时此时可直接使用

resultType将每一条记录映射到pojo中在前端页面遍历listlist中是pojo即可。

resultMap总结：

使用association和collection完成一对一和一对多高级映射对结果有特殊的映射要求。

association www.docin.comJavaEE课程 作用

将关联查询信息映射到一个pojo对象中。

场合

为了方便查询关联信息可以使用association将关联订单信息映射为用户对象的pojo属性中比如查询订单

及关联用户信息。

使用resultType无法将查询结果映射到pojo对象的pojo属性中根据对结果集查询遍历的需要选择使用

resultType还是resultMap。

collection

作用

将关联查询信息映射到一个list集合中。

场合

为了方便查询遍历关联信息可以使用collection将关联信息映射到list集合中比如查询用户权限范围模块

及模块下的菜单可使用collection将模块映射到模块list中将菜单列表映射到模块对象的菜单list属性中这

样的作的目的也是方便对查询结果集进行遍历查询。

如果使用resultType无法将查询结果映射到list集合中。

6 延迟加载 6.1 什么是延迟加载

resultMap可以实现高级映射使用association、collection实现一对一及一对多映射association、collection具

备延迟加载功能。

需求

如果查询订单并且关联查询用户信息。如果先查询订单信息即可满足要求当我们需要查询用户信息时再查询用

户信息。把对用户信息的按需去查询就是延迟加载。

二级缓存应用场景：

对访问多的，并实时要求不高的时候，可以采用mybatis的二级缓存，降低对数据库的访问量。例如，耗时较高的统计分析sql

实现方法如下：通过设置刷新间隔时间，由mybatis每隔一段时间，自动清空缓存，根据变化频率设置刷新间隔时间。