**“”  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
从v  亲爱·Ytyk<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  
mybatis的mapper.xml配置**

**ps--》用于配置sql执行环境**

**命名空间，类型别名，sql的代码，参数映射，返回值映射，动态拼接sql。。。**

**-------------------------------基本结构----------------------------------------**

<?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="**命名空间（一般填模块名），防止sql的id冲突，一定要配置**">

</mapper>

**-------------------------------重要配置---------------------------------------**

**insert、delete、update语句**

<insert  id="insertAuthor"  parameterType="domain.blog.Author"

  keyProperty=""  keyColumn=""  useGeneratedKeys="" >

    insert的语句

</insert >

<update >

<delete >

**示例：**

<insert id="insertAuthor" useGeneratedKeys="true"

    keyProperty="id">

  insert into Author (username,password,email,bio)

  values (#{username},#{password},#{email},#{bio})

</insert>

ps-》如果数据库不支持自动生成主键-》也采用uuid+时间戳。

**语句中的参数**

${}：直接把参数拼接到sql

#{}：先拼接？ 再设置参数 -防止sql注入

ps-》尽量用#

**-------------session的sql参数（一个Object参数）------------------**

**a）传基本类型包装类型字符串等**

-》#{任何名}一般用#{param}

**b）bean对象或Map对象**

${map的key或bean的属性名}

#{map的key或bean的属性名}

bean对象：优点可读性好，缺点加参数该参数需要修改类

map对象：有点代码少，不需要增加类，可读性差

**c）数组或list**

    #｛list[下标]｝#｛array[下标]｝

**-------------接口开发自定义的参数（可以多个参数）----------------**

**ps-》多个参数自动从0开始编号**

**a）通过下标取出参数**

#{下标}

b）注意不能直接通过形参名取出参数-如果想使用名字（需要配合 @param(“参数名”)）来设置使用

#{ @param的参数名}

c）如果【一个参数】的时候

**1）bean对象或Map对象**

${map的key或bean的属性名}

#{map的key或bean的属性名}

bean对象：优点可读性好，缺点加参数该参数需要修改类

map对象：有点代码少，不需要增加类，可读性差

**2）数组或list**

    #｛list[下标]｝#｛array[下标]｝

**----------------------------------------------------------------------------------**

select \* from teacher where tid=#{tid}

select \* from teacher where tid=${tid}

**#{参数名}：防止sql注入，采用?，再set参数，数字参数会被转换为字符串**

如果参数值是：111

select \* from teacher where tid='111'

**${参数名}：不能防止sql注入，直接拼接参数**

如果参数值是：111

select \* from teacher where tid= 111

**#的其他配置**

#{参数名,jdbcType=NUMERIC,typeHandler=MyTypeHandler}

ps-》

jdbcType=NUMERIC：对应的数据库类型

typeHandler=MyTypeHandler：转换使用的处理器（一般自定义类型转换使用）

**select语句**

<select

  id="selectPerson"

  parameterType="int"

  parameterMap="**已经废弃了**"

  resultType="hashmap"

  resultMap="personResultMap"

  ">

**parameterType：参数类型，一般使用：基本类型，javaBean，map**

**resultType：（结果类型自动映射），一般使用：基本类型，javaBean，map**

**---》自动把查询的结果映射到对应的类型中**

**ps-》注意如果是集合情形，那应该是集合可以包含的类型，而不能是集合本身**

**resultMap（结果类型手动映射）**：**手动把查询的结果需要的列和java属性进行映射。（并且可以配置映射到bean）**

**列名不匹配，类型不匹配，个数不匹配**

**示例1：**

<select id="selectUsers" resultType="User">

  select user\_id, user\_name, hashed\_password

  from some\_table

  where id = #{id}

</select>

**ps-》这里会把结果**user\_id, user\_name, hashed\_password自动放入User【同名的属性】中

**select语句-》resultMap手动结果集映射**

**基本的映射（单表）**

**示例2：**

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

  <id property="id" column="user\_id" />

  <result property="username" column="user\_name"/>

</resultMap>

<select id="selectUsers" resultMap="userResultMap">

  select user\_id, user\_name, hashed\_password

  from some\_table

  where id = #{id}

</select>

**ps-》这里会把结果**user\_id, user\_name,自动放入User【id和username】中

**高级结果映射（多表）（处理-当前表关联的一方（一个对象）-关联的多方（集合对象））**

**PS->自动在程序中把表的结果数据进行分组聚合处理，并封装到bean**

**表说明：博客表，作者表，评论表，回复表**

**一个作者可以写多个博客，一个博客可以有多个评论，一个评论可以多个回复**

**示例2：**

**--》以下查询是一个博客的列表-也关联查询了一个作者数据（装入了**Author 对象**），多个评论数据**

<select id="selectBlogDetails" resultMap="detailedBlogResultMap">

  select

       B.id as blog\_id,

       B.title as blog\_title,

       B.author\_id as blog\_author\_id,

       A.id as author\_id,

       A.username as author\_username,

       A.email as author\_email,

       P.id as post\_id,

       P.subject as post\_subject,

       P.draft as draft,

       P.body as post\_body,

    an.answer\_id,

    an.content

 from **Blog**B

       left join **Author**A on B.author\_id = A.id

       left join **Post**P on B.id = P.blog\_id

       left join **Answer**an on  an.post\_id=p.id

</select>

**--------------------------------------------------------------------**

<resultMap id="detailedBlogResultMap" type="**Blog**">

  <constructor>

    <idArg column="blog\_id" javaType="int"/>

  </constructor>

  <result property="title" column="blog\_title"/>

**<association property="author" javaType="Author">**

    <id property="id" column="author\_id"/>

    <result property="username" column="author\_username"/>

    <result property="email" column="author\_email"/>

**</association>**

**<collection property="posts" ofType="Post">**

    <id property="id" column="post\_id"/>

    <result property="subject" column="post\_subject"/>

  **<collection property="answers" ofType="Answer" >**

      <id property="id" column="answer\_id"/>

      <result property="content" column="content"/>

**</collection>**

**</collection>**

</resultMap>

resultMap的 子标记**association ：处理当前bean关联的一个bean（针对一方）**

**java类型就是一个bean**

resultMap的 子标记**collection ：处理当前bean关联的多个个bean（针对多方）**

**java类型就是一个conllection<bean>**

**--------------------------------------------------------------------------------------------------------**

**ResultMap里面association 和collection**

**--可以参考已经定义的resultMap，方便多次复用**

**上述【ResultMap】可以改为**

<resultMap id="detailedBlogResultMap" type="**Blog**">

  <constructor>

    <idArg column="blog\_id" javaType="int"/>

  </constructor>

  <result property="title" column="blog\_title"/>

**<association property="author"**resultMap="**authorMap**" **>**

**</association>**

**<collection property="posts"**resultMap="**postMap**" **>**

**</collection>**

</resultMap>

<!-- 作者map -->

<resultMap id="**authorMap**" type="**Author**">

       <id property="id" column="author\_id"/>

    <result property="username" column="author\_username"/>

    <result property="email" column="author\_email"/>

</resultMap>

<!-- 评论map -->

<resultMap id="**postMap**" type="**Post**">

       <id property="id" column="post\_id"/>

    <result property="subject" column="post\_subject"/>

  **<collection property="answers" ofType="Answer" >**

      <id property="id" column="answer\_id"/>

      <result property="content" column="content"/>

**</collection>**

</resultMap>

**---------------------------------------------------------------------------------------------**

**关联查询**

**N+1-问题**

a）你执行了一个单独的 SQL 语句来获取结果列表(就是“+1”)。

b）对返回的每条记录,你执行了一个查询语句来为每个加载细节(就是“N”)。

**ps-》association 和collection 标记关联了select进行查询，可以进行懒加载，需要在config.xml里启用懒加载功能，底层需要使用动态代理，需要使用cglib.jar支持**

<settings>

<setting name="lazyLoadingEnabled" value="true"/>

<setting name="aggressiveLazyLoading" value="false"/>

</settings>

**关联查询-对一方**

<resultMap id="blogResult" type="Blog">

  <association

**property="author"**

**column="author\_id"**

**javaType="Author"**

**select="selectAuthor"**/>

</resultMap>

**----------------------------**

**property="author"  对应java 成员属性名**

**javaType="Author" 对应java 成员属性类型**

**column="author\_id" 对应关联对方的列（一般是：对应一方（外键参考列），对应多方（被参考列）），该列的数据会被作为参数设置，另一个参数用#｛｝取出，如果是联合主键column= ” {prop1=col1,prop2=col2} ”。**

**select="selectAuthor" 对应关联的查询语句的id**

**---------------------------**

<select id="selectBlog" resultMap="blogResult">

  SELECT \* FROM BLOG WHERE ID = #{id}

</select>

<select id="selectAuthor"**resultType="Author"**>

  SELECT \* FROM AUTHOR WHERE ID =**#{id}**

</select>

**关联查询-对多方**

<resultMap id="blogResult" type="Blog">

  <collection property="posts"

javaType="ArrayList"

column="id"

ofType="Post" select="selectPostsForBlog"/>

</resultMap>

<select id="selectBlog" resultMap="blogResult">

  SELECT \* FROM BLOG WHERE ID = #{id}

</select>

<select id="selectPostsForBlog" resultType="Post">

  SELECT \* FROM POST WHERE BLOG\_ID = #{id}

</select>