2017年12月13日

11:34

**Mybatis总结**

**总结1**

1.Mybatis是一个持久层框架，作用是跟数据库交互完成增删改查

2.原生Dao实现

3.动态代理方式（只需要接口）

mapper接口代理实现填写规则：

1.映射文件中namespace要等于接口的全路径名

2.映射文件中sql语句id要等于接口的方法名称

3.映射文件中传入参数类型要等于接口方法的传入参数类型

4.映射文件中返回结果集类型要等于接口方法的返回值类型

4.#{}占位符：占位

如果传入的是基本类型，那么#{}中的变量名可以随意写

如果传入的参数是pojo类型，那么#{}中的变量名称必须是pojo中的属性.属性…..

5.${}拼接符：字符串原样拼接

如果传入的是基本类型，那么${}中的变量名必须是value

如果传入的参数是pojo类型，那么${}中的变量名称必须是pojo中的属性.属性…..

注意：使用拼接符有可能造成sql注入，在页面输入的时候可以加入校验，不可输入sql关键字，不可输入空格

6.映射文件

1.传入参数类型通过parameterType属性指定

2.返回结果集类型通过resultType属性指定

7.hibernate和mybatis区别：

Hibernate：它是一个标准的orm框架，比较重量级，学习成本高

优点：高度封装，使用起来不用写sql，开发时会降低开发周期

缺点：sql语句无法优化

应用场景：oa（办公自动化系统），erp（企业的流程系统）等，还有一些政府项目，总的来说，在用于量不大，并发量小的时候使用。

Mybatis：它不是orm框架，它是jdbc的轻量级封装，学习成本低，比较简单

优点：学习成本低，sql语句可以优化，执行效率高，速度快

缺点：编码量较大，会拖慢开发周期

应用场景：互联网项目，比如电商，p2p等，总之，用户量较大，并发高的项目。



**总结2**

* 1. 输入映射（就是映射文件中可以传入哪些参数类型）
  2. 基本类型
  3. Pojo类型
  4. Vo类型
  5. 输出映射（返回的结果集可以有哪些类型）
  6. 基本类型
  7. Pojo类型
  8. List类型
  9. 动态sql：动态的拼接sql语句，因为sql中where条件有可能多也有可能少
  10. where：可以自动添加where关键字，还可以去掉第一个条件的and关键字
  11. if：判断传入的参数是否为空
  12. foreach：循环遍历传入的集合参数
  13. sql：封装查询条件，以达到重用的目的
  14. 对单个对象的映射关系：
  15. 自动关联（偷懒的办法）：可以自定义一个大而全的pojo类，然后自动映射其实是根据数据库中的字段名称和pojo中的属性名称对应。
  16. 手动关联：需要指定数据库中的表的字段名称和Java的pojo类中的属性名称的对应关系。使用association标签。
  17. 对集合对象的映射关系

只能使用手动映射：指定表中字段名称和pojo中属性名称的对应关系，使用collection标签。

* 1. spring和mybatis整合

整合后会话工厂都归spring管理

* 1. 原生Dao实现：

需要在spring配置文件中指定dao实现类

Dao实现类需要继承SqlSessionDaoSupport超类

在dao实现类中不要手动关闭会话，不要手动提交事务

* 1. Mapper接口代理实现：

在spring配置文件中可以使用包扫描的方式一次性将所有mapper加载

* 1. 逆向工程：自动生成pojo类，还可以自动生成mapper接口和映射文件

注意：生成的方式是追加而不是覆盖，所以不可以重复生成，重复生成的文件有问题。

数据库函数：

|  |  |
| --- | --- |
| SELECT LAST\_INSERT\_ID(); | 自增主键时使用，MySQL中 |
| SELECT UUID() | 自增主键UUID，MySQL的主键生成策略 |
| SELECT seq.nextval FROM dual | 序列sequence，是Oracle的主键生成策略，返回值可以是int型 |

**映射文件：UserMappeer.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="com.itcast.mapper.UserMapper">

<select id="findUserById" parameterType="int" resultType="user">

select

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

</select>

<select id="findUserByName" parameterType="java.lang.String"

resultType="User">

select \* from user where username like '%${value}%';

</select>

<insert id="addUser" parameterType="com.itcast.pojo.User">

<selectKey keyProperty="id" order="AFTER" resultType="int">

SELECT

LAST\_INSERT\_ID();

</selectKey>

insert into user (username,birthday,sex,address)

values

(#{username},#{birthday},#{sex},#{address});

</insert>

<delete id="delUserById" parameterType="int">

delete from user where

id=#{id};

</delete>

<update id="modifyUserById" parameterType="com.itcast.pojo.User">

update user set

username = #{username},

birthday = #{birthday},

sex = #{sex},

address =

#{address}

where id = #{id};

</update>

<select id="findUsersByQueryVo" parameterType="queryVo"

resultType="user">

select \* from user

where username like '%${user.username}%'

and sex = #{user.sex};

</select>

<select id="findUsersTotalNumber" resultType="int">

select count(\*)

from user;

</select>

<sql id="where\_user">

<where>

<if test="username != null and username != ''">

and username like '%${username}%'

</if>

<if test="sex != null and sex !=''">

and sex = #{sex}

</if>

</where>

</sql>

<select id="findUsersByNameAndSex" parameterType="user"

resultType="user">

select \* from user

<include refid="where\_user" />

</select>

<select id="findUsersByIds" parameterType="queryVo" resultType="user">

select \* from user

<where>

<if test="ids != null">

<foreach collection="ids" item="id" open="id in(" close=")"

separator=",">

#{id}

</foreach>

</if>

</where>

</select>

<select id="findOrdersAndUser1" resultType="customOrders">

SELECT a.\*,b.id uid,b.username,b.birthday,b.sex,b.address

FROM orders a,user b WHERE a.user\_id = b.id;

</select>

<resultMap type="orders" id="ordersMap">

<id column="id" property="id"/>

<result column="user\_id" property="userId"/>

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

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

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

<association property="user" javaType="user">

<id column="uid" property="id"/>

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

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

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

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

</association>

</resultMap>

<select id="findOrdersAndUser2" resultMap="ordersMap">

SELECT a.\*,b.id uid,b.username,b.birthday,b.sex,b.address

FROM orders a,user b WHERE a.user\_id = b.id;

</select>

<resultMap type="user" id="userMap">

<id column="id" property="id"/>

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

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

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

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

<collection property="ordersList" ofType="orders">

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

<result column="user\_id" property="userId"/>

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

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

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

</collection>

</resultMap>

<select id="findUserAndOrder" resultMap="userMap">

SELECT a.\*,b.id o\_id,b.user\_id,b.number,b.createtime,b.note

FROM user a,orders b WHERE a.id = b.user\_id;

</select>

</mapper>