0623技术整理

> title：0623技术整理
>
> author：刘星烨
>
> content：Mybatis

Mybatis基础

持久层框架，主要解决数据库相关操作问题

ORM思想

- 让实体类和数据库表进行一一对应关系

  先让实体类和数据库表对应

  再让实体类属性和表里面字段对应

- 不需要直接操作数据库表，直接操作表对应的实体类对象

Mybatis映射文件

```
<?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="UserMapper">
  <!--查询所有-->
  <select id="findAll" resultType="com.lagou.domain.User">
    select * from user
  </select>
</mapper>
```

Line1: xml相关定义

Line2/3: 映射文件DTD约束头

Line4: 映射根标签，规定namespace，与下面的id共同组成查询标识（通过namespace.id调用某条查询）

Line5: 操作 insert、update、select、delete等。resultType指定结果对应的实体类型。

Line6: 执行的sql语句

Mybatis实现操作

- insert: 注意parameterType，#{param} 这种格式使用

```
<insert id="save" parameterType="com.lagou.domain.User"> insert into
user(username,birthday,sex,address) values(#{username},#{birthday},#
{sex},#{address}) </insert>
```

- delete

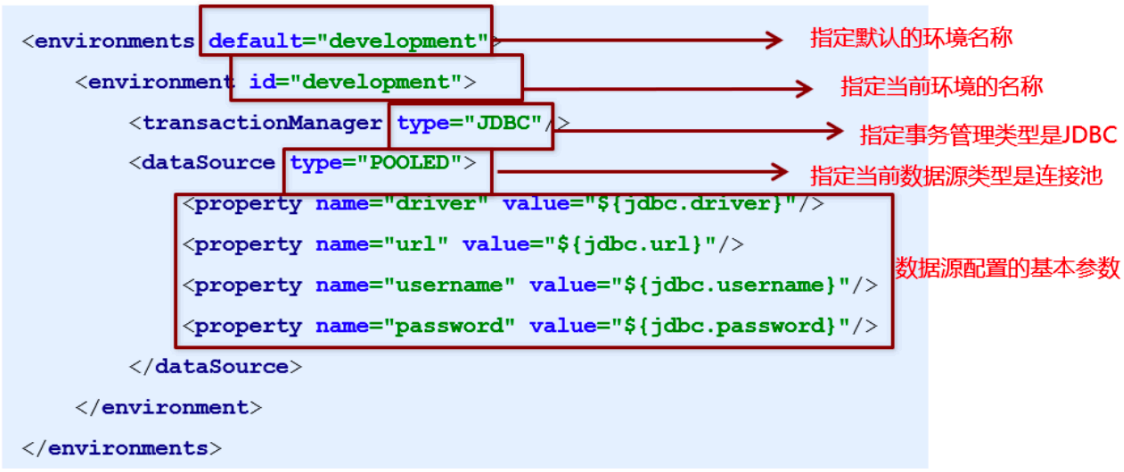
```
<delete id="delete" parameterType="java.lang.Integer"> delete from user
where id = #{id} </delete>
```

- select

```
<select id="findAll" resultType="com.lagou.domain.User">
    select * from user
 </select>
```

Mybatis配置分析

- environments配置



- properties标签

```
<properties resource="jdbc.properties"></properties>
```

在jdbc.properties文件中加入具体的配置信息

- typeAliases标签

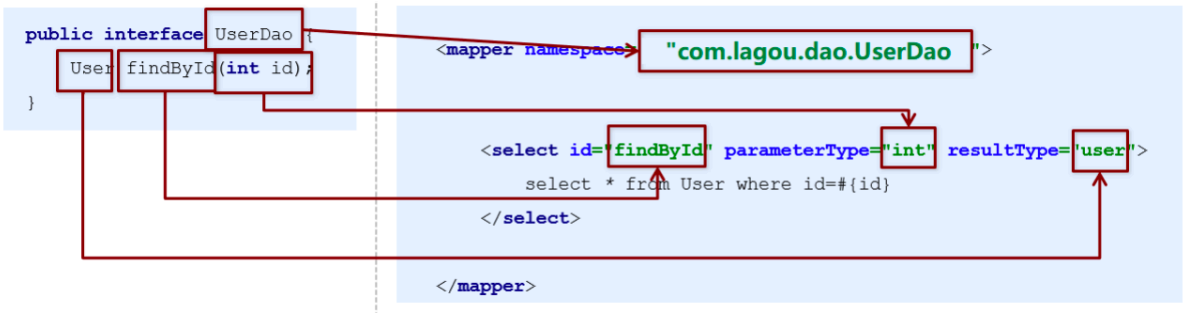
加载别名

Mybatis开发方式

代理开发模式。

需遵循的规范：

- Mapper.xml映射文件中的namespace与mapper接口的全限定名相同
- Mapper接口方法名和Mapper.xml映射文件中定义的每个statement的id相同
- Mapper接口方法的输入参数类型和mapper.xml映射文件中定义的每个sql的parameterType的类型相同
- Mapper接口方法的输出参数类型和mapper.xml映射文件中定义的每个sql的resultType的类型相同



主要的作用是用代理对象实现了SqlSession的多次利用，不用重复创建关闭。

Mybatis应用

ResultMap建立映射

- resultType：如果实体的属性名与表中字段名一致，将查询结果自动封装到实体类中
- ResultMap：如果实体的属性名与表中字段名不一致，可以使用ResutlMap实现手动封装到实体类中

```
<resultMap id="userResultMap" type="user"> <id column="uid"
property="id"></id>
<result column="NAME" property="username"></result> <result
column="PASSWORD" property="username"></result> </resultMap> <select
id="findAllResultMap" resultMap="userResultMap"> SELECT id AS
uid,username AS NAME,password AS PASSWORD FROM USER </select>
```

多条件查询 参数

- 使用 #{arg0}-#{argn} 或者 #{param1}-#{paramn} 获取参数
- 使用注解，引入 @Param() 注解获取参数
- 使用pojo对象传递参数

映射文件

返回主键

声明返回主键，把返回主键的值，封装到实体的id属性中

注意：只适用于主键自增的数据库，mysql和sqlserver支持，oracle不支持

```
<useGeneratedKeys="true" keyProperty="id"  >
```

selectKey

```
<insert id="save" parameterType="user"> <selectKey keyColumn="id"
keyProperty="id" resultType="int" order="AFTER"> SELECT LAST_INSERT_ID();
</selectKey> INSERT INTO `user`(username,birthday,sex,address) values(#
{username},#{birthday},#{sex},#{address}) </insert>
```

SQL的多种表示

将筛选where以及set结果设置为标签，在if标签内判断后续执行语句。

提取重复的sql，在使用时用include引用即可。

标签总结

```
<select>: 查询 <insert>: 插入 <update>: 修改 <delete>: 删除 <selectKey>: 返回主
键 <where>: where条件 <if>: if判断 <foreach>: for循环 <set>: set设置 <sql>: sql
片段抽取
```

加载策略与注解开发

延迟加载

就是在需要用到数据时才进行加载，不需要用到数据时就不加载数据。延迟加载也称懒加载。

```
<collection property="orderList" ofType="order" column="id"
select="com.lagou.dao.OrderMapper.findByUid" fetchType="lazy">
</collection>
```

使用fetchType标签可以启用延迟加载

全局延迟加载

```
<settings> <setting name="lazyLoadTriggerMethods" value="toString()"/>
</settings>
```

在配置了延迟加载策略后，发现即使没有调用关联对象的任何方法，但是在你调用当前对象的equals、clone、hashCode、toString方法时也会触发关联对象的查询。

我们可以在配置文件中使用lazyLoadTriggerMethods配置项覆盖掉上面四个方法。

注解开发

常用注解

```
* @Insert：实现新增，代替了<insert></insert>
* @Delete：实现删除，代替了<delete></delete>
* @Update：实现更新，代替了<update></update>
* @Select：实现查询，代替了<select></select>
* @Result：实现结果集封装，代替了<result></result>
* @Results：可以与@Result 一起使用，封装多个结果集，代替了<resultMap></resultMap>
* @One：实现一对一结果集封装，代替了<association></association>
* @Many：实现一对多结果集封装，代替了<collection></collection>
```