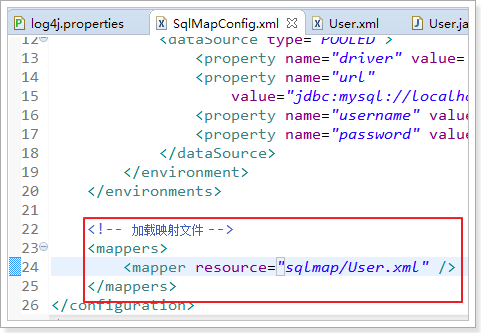
## Mybatis的核心配置文件名称

SqlMapConfig.xml

## 核心配置文件加载映射配置文件



## 映射配置文件的属性

<?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">

<!-- namespace：命名空间，用于隔离sql，还有一个很重要的作用，后面会讲 -->

<mapper namespace="test">

<!-- id:statement的id 或者叫做sql的id-->

<!-- parameterType:声明输入参数的类型 -->

<!-- resultType:声明输出结果的类型，应该填写pojo的全路径 -->

<!-- #{}：输入参数的占位符，相当于jdbc的？ -->

<select id="queryUserById" parameterType="int"

resultType="cn.itcast.mybatis.pojo.User">

SELECT \* FROM `user` WHERE id = #{id}

</select>

</mapper>

## #{}和${}的区别

#{}表示一个占位符号，通过#{}可以实现preparedStatement向占位符中设置值，自动进行java类型和jdbc类型转换。#{}可以有效防止sql注入。 #{}可以接收简单类型值或pojo属性值。 如果parameterType传输单个简单类型值，#{}括号中可以是value或其它名称。

${}表示拼接sql串，通过${}可以将parameterType 传入的内容拼接在sql中且不进行jdbc类型转换， ${}可以接收简单类型值或pojo属性值，如果parameterType传输单个简单类型值，${}括号中只能是value。

## mysql自增主键返回

<insert id="saveUser" parameterType="cn.itcast.mybatis.pojo.User">

<!-- selectKey 标签实现主键返回 -->

<!-- keyColumn:主键对应的表中的哪一列 -->

<!-- keyProperty：主键对应的pojo中的哪一个属性 -->

<!-- order：设置在执行insert语句前执行查询id的sql，孩纸在执行insert语句之后执行查询id的sql -->

<!-- resultType：设置返回的id的类型 -->

<selectKey keyColumn="id" keyProperty="id" order="AFTER"

resultType="int">

SELECT LAST\_INSERT\_ID()

</selectKey>

INSERT INTO `user`

(username,birthday,sex,address) VALUES

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

</insert>

## mybatis与hibernate不同

1:mybatis不完全是一个orm框架,需要手动写sql.通过xml的方式灵活的配置要执行的sql语句.

2:mybatis学习门槛低,可以控制sql,灵活度高,无法做到与数据库的无关性.hibernate的关系映射能力较强,可以与数据库做到很好的无关性,但是学习门槛高.

## SqlSession

SqlSession是一个面向用户的接口，sqlSession中定义了数据库操作方法。

每个线程都应该有它自己的SqlSession实例。SqlSession的实例不能共享使用，它也是线程不安全的。因此最佳的范围是请求或方法范围。绝对不能将SqlSession实例的引用放在一个类的静态字段或实例字段中。

## 动态代理方式

Mapper接口开发方法只需要程序员编写Mapper接口（相当于Dao接口），由Mybatis框架根据接口定义创建接口的动态代理对象，代理对象的方法体同上边Dao接口实现类方法。

Mapper接口开发需要遵循以下规范：

1、 Mapper.xml文件中的namespace与mapper接口的类路径相同。

2、 Mapper接口方法名和Mapper.xml中定义的每个statement的id相同

3、 Mapper接口方法的输入参数类型和mapper.xml中定义的每个sql 的parameterType的类型相同

4、 Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同

## 自定义别名

在核心配置文件中自定义别名

<typeAliases>

<!-- 单个别名定义 -->

<typeAlias alias="user" type="cn.itcast.mybatis.pojo.User" />

<!-- 批量别名定义，扫描整个包下的类，别名为类名（大小写不敏感） -->

<package name="cn.itcast.mybatis.pojo" />

<package name="其它包" />

</typeAliases>

## Mapper配置的几种方法：

7.4.1. <mapper resource=" " />

使用相对于类路径的资源（现在的使用方式）

如：<mapper resource="sqlmap/User.xml" />

7.4.2. <mapper class=" " />

使用mapper接口类路径

如：<mapper class="cn.itcast.mybatis.mapper.UserMapper"/>

注意：此种方法要求mapper接口名称和mapper映射文件名称相同，且放在同一个目录中。

7.4.3. <package name=""/>

注册指定包下的所有mapper接口

如：<package name="cn.itcast.mybatis.mapper"/>

注意：此种方法要求mapper接口名称和mapper映射文件名称相同，且放在同一个目录中。

## Resultmap

数据库的name与实体类的属性不一致的时候用resultmap封装

<mapper namespace="cn.itcast.mybatis.mapper.OrderMapper">

<!-- resultMap最终还是要将结果映射到pojo上，type就是指定映射到哪一个pojo -->

<!-- id：设置ResultMap的id -->

<resultMap type="order" id="orderResultMap">

<!-- 定义主键 ,非常重要。如果是多个字段,则定义多个id -->

<!-- property：主键在pojo中的属性名 -->

<!-- column：主键在数据库中的列名 -->

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

<!-- 定义普通属性 -->

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

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

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

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

</resultMap>

<!-- 查询所有的订单数据 -->

<select id="queryOrderAll" resultMap="orderResultMap">

SELECT id, user\_id,

number,

createtime, note FROM `order`

</select>

</mapper>

## Xml中的if标签

<select id="queryUserByWhere" parameterType="user" resultType="user">

SELECT id, username, birthday, sex, address FROM `user`

WHERE 1=1

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

AND sex = #{sex}

</if>

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

AND username LIKE

'%${username}%'

</if>

</select>

Where标签

去掉前边的and

<select id="queryUserByWhere" parameterType="user" resultType="user">

SELECT id, username, birthday, sex, address FROM `user`

<!-- where标签可以自动添加where，同时处理sql语句中第一个and关键字 -->

注意 where标签只能去掉前边的and

<where>

<if test="sex != null">

AND sex = #{sex}

</if>

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

AND username LIKE

'%${username}%'

</if>

</where>

</select>

## Sql片段

<sql id="userFields">

id, username, birthday, sex, address

</sql>

## foreach标签

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

SELECT \* FROM `user`

<where>

<!-- foreach标签，进行遍历 -->

<!-- collection：遍历的集合，这里是QueryVo的ids属性 -->

<!-- item：遍历的项目，可以随便写，，但是和后面的#{}里面要一致 -->

<!-- open：在前面添加的sql片段 -->

<!-- close：在结尾处添加的sql片段 -->

<!-- separator：指定遍历的元素之间使用的分隔符 -->

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

separator=",">

#{item}

</foreach>

</where>

</select>

## Mapper的一对一封装对象

<resultMap type="order" id="orderUserResultMap">

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

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

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

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

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

<!-- association ：配置一对一属性 -->

<!-- property:order里面的User属性名 -->

<!-- javaType:属性类型 -->

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

<!-- id:声明主键，表示user\_id是关联查询对象的唯一标识-->

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

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

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

</association>

</resultMap>

## Mapper的一对多

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

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

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

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

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

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

<!-- 配置一对多的关系 -->

<collection property="orders" javaType="list" ofType="order">

<!-- 配置主键，是关联Order的唯一标识 -->

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

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

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

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

</collection>

</resultMap>

## Spring整合mybatis代理开发

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<!-- 配置Mapper接口 -->

<property name="basePackage" value="cn.itcast.mybatis.mapper" />

</bean>