Day30

java课程-李彦伯

Mybatis

SqlMapConfig.xml配置文件

- properties属性引入外部的properties文件,可以用于引入外部的数据库连接文件,使用的时候使用el表达式进行引入,`路径是src路径`,其内部有标签`<property>`,如果两个都进行设置,会先加载内部的property再加载外部的properties的resource属性

```
<configuration>

    <properties resource="db.properties">
        <property name="jdbc.user" value="roo
t"/>
        <property name="jdbc.password" valu
e="root"/>
    </properties>
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC"
/>
            <dataSource type="POOLED">
                <property name="driver" valu
e="${jdbc.driverClass}" />
                <property name="url" valu
e="${jdbc.jdbcUrl}" />
                <property name="username" val
ue="${jdbc.user}" />
                <property name="password" val
ue="${jdbc.password}" />
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <mapper resource="sqlmap/User.xml"/>
    </mappers>
</configuration>
```

- typeAliases类型别名属性,在mapper文件中,如果使用参数类型或者返回值类型的时候需要指定类的全名,可以使用别名

来进行配置使用其内部的标签`<typeAlias type="com.zhiyou100.mybatis.model.User" alias="User"/>`将对应的类的别名设置为User或者使用`<package name="com.zhiyou100.mybatis.model"/>`会自动扫描当前包和其子包将别名配置为`类名首字母大写(User)`和`类名小写(user)`,推荐使用这种

```
<typeAliases>
    <!--
        设置com.zhiyou100.mybatis.model.User的别名为User,如果这样设置,每一个model都需要设置
     -->
    <!-- <typeAlias type="com.zhiyou100.mybatis.model.User" alias="User"/> -->
    <!--
        会自动设置当前包和其子包中的所有的类的别名为User或者user
        一般在开发的过程中使用package
     -->
    <package name="com.zhiyou100.mybatis.model"/>
</typeAliases>
```

- mybatis内部配置的别名,在mapper中写哪个都可以

别名	映射的类型	别名	映射的类型
_byte	byte	long	Long
_long	long	short	Short

_short	short	int	Integer
_int	int	integer	Integer
_integer	int	double	Double
_double	double	float	Float
_float	float	boolean	Boolean
_boolean	boolean	date	Date
string	String	decimal	BigDecimal
byte	Byte	bigdecimal	BigDecimal
map	Map		

- mappers映射器属性,内部设置的标签是`<mapper>`用来设置mapper映射文件的路径
  - `<mapper resource="sqlmap/User.xml"/>`resource指定的classpath路径不是相对路径
  - `<mapper class="com.zhiyou100.mybatis.mapper.UserMapper"/>`通过接口文件找到对应的xml文件,**此种状态下必须保证xml文件和接口名称相同,且放在同一个包下**
  - `<package name="com.zhiyou100.mybatis.mapper"/>`会扫描指定包下及其子包中的所有的mapper文件,**此种状态下必须保证xml文件和接口名称相同,且放在同一个包下**

输入映射(parameterType)

> POJO（Plain Ordinary Java Object）简单的Java对象，实际就是普通JavaBeans

- 简单类型
  - 整型
  - String类型

- 复杂类型
  - POJO
  - POJO的包装类(内部包含一个普通的类,可以进行多条件查询)
  - 应用范围:查询条件可能是综合的查询条件，不仅包括用户查询条件还包括其它的查询条件（比如查询用户信息的时候，将用户购买商品信息也作为查询条件），这时可以使用包装对象传递输入参数。

输出映射(resultType)

- 简单类型
  - 整型
  - 字符串

```
<select id="findUserCount" resultType="Intege
r">
select count(*)  from user
</select>
<select id="findUserName" resultType="String"
parameterType="Integer">
select username  from user where id = #{v}
</select>
```

- 复杂类型
  - POJO
  - POJO列表

resultMap

> 如果sql查询字段名和pojo的属性名不一致，可以通过resultMap将字段名和属性名作一个对应关系 ，resultMap实质上还需要将查询结果映射到pojo对象中。

- id映射`<id column="id" property="id"/>`
- 其他字段映射`<result column="user_id" property="userId"/>`
- 一对一映射`<association property="">`
  `</association>`
- 一对多映射`<collection property=""></collection>`

```
<mapper namespace="com.zhiyou100.mybatis.mapper.OrderMapper">
<resultMap type="Orders" id="orderUserMap">
    <result column="user_id" property="userId"/>
</resultMap>
<select id="findAllOrder" resultMap="orderUserMap">
SELECT id, user_id,number,createtime, note FROM  orders
</select>
</mapper>
```

动态sql

if标签

> 可用于字符串的非空判断

```
<!-- 根据性别和姓名查询用户 -->
<select id="findUserBySexAndUserName" paramet
erType="User" resultType="User">
select * 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

> 可以去掉第一个前and标签

```
<!-- 根据性别和姓名查询用户 -->
<select id="findUserBySexAndUserName" paramet
erType="User" resultType="User">
select * from user
<where>
<if test="sex != null and sex != ''">
and sex = #{sex}
</if>
<if test="username != null and username !=
''">
and  username like '%' #{username} '%'
</if>
</where>
</select>
```

sql片段

> 可以把多次出现的sql封装成一个片段,在使用的时候进行引用

```
<sql id="selector">
select * from user
</sql>

<!-- 根据性别和姓名查询用户 -->
<select id="findUserBySexAndUserName" paramet
erType="User" resultType="User">
<include refid="selector"/>
<where>
<if test="sex != null and sex != ''">
and sex = #{sex}
</if>
<if test="username != null and username !=
''">
and  username like '%' #{username} '%'
</if>
</where>
</select>
```

foreach标签

> 向sql传递数组或List，mybatis使用foreach解析，如下：根据多个id查询用户信息查询sql：SELECT * FROM user WHERE id IN (1,10,24)

```
<select id="findUserByIds" parameterType="Que
ryVO" resultType="User">
<include refid="selector"/>
<where>
    <foreach collection="idList" item="id" se
parator="," open="id in  (" close=")">
        #{id}
    </foreach>
</where>
</select>
```

```
public void testUser03(){
    SqlSession session = factory.openSessio
n();
    QueryVO qv = new QueryVO();
    List<Integer> list = new ArrayList<>();
    list.add(1);
    list.add(10);
    list.add(16);
    qv.setIdList(list);
    UserMapper user = session.getMapper(UserM
apper.class);
    System.out.println(user.findUserByIds(lis
t));
}
```

```
<select id="findUserByIds" resultType="User">
<include refid="selector"/>
<where>
<--
如果传入的是数组collection要写array
如果传入的是listcollection要写list
-->
    <foreach collection="arrays" item="id" se
parator="," open="id in  (" close=")">
        #{id}
    </foreach>
</where>
</select>
```

```
public void testUser03(){
    SqlSession session = factory.openSessio
n();
    UserMapper user = session.getMapper(UserM
apper.class);
    Integer [] ids = new Integer[]{1,10,16};
    System.out.println(user.findUserByIds(id
s));
}
```

关联查询

一对一查询

将order类中添加一个username属性

```
<resultMap type="Orders" id="otou">
<id column="user_id" property="userId"/>
</resultMap>
<select id="findAllOrderIncludeUser" resultMa
p="otou">
select o.*,username from orders o left join u
ser u on o.user_id = u.id
</select>
```

将order类中添加一个User类型属性

```
<resultMap type="Orders" id="otou">
<id column="oid" property="id"/>
<result column="user_id" property="userId"/>
<result column="number" property="number"/>
<result column="createtime" property="createt
ime"/>
<result column="note" property="note"/>
<association property="u" javaType="User">
<result column="username" property="usernam
e"/>
<result column="address" property="address"/>
</association>
</resultMap>
<select id="findAllOrderIncludeUser" resultMa
p="otou">
select o.id oid ,o.user_id,o.number,o.createt
ime,o.note,u.username,u.address
 from orders o
 left join
 user u
 on o.user_id = u.id
</select>
```

一对多查询

```
<resultMap type="User" id="userWithOrder">
<id column="id" property="id"/>
<result column="username" property="usernam
e"/>
<collection property="orders" ofType="Order
s">
<result column="createtime" property="createt
ime"/>
</collection>
</resultMap>

<select id="findAllUserWithOrder" resultMa
p="userWithOrder">
select u.id, u.username,o.createtime from use
r u left join orders o on u.id = o.user_id
</select>
```