动态SQL

实例：

```
<select id="findUserByCondition" resultMap="userMap" parameterType="user">
      select * from user where 1=1
   </select>
```

一、if 语句

```
<select id="findUserByCondition" resultMap="userMap" parameterType="user">
      select * from user where 1=1
      <if test="userName != null">
        and username = #{userName}
      </if>

      <if test="userSex != null">
         and sex = #{userSex}
      </if>
</select>
```

注意：条件之间的连接用 and

二、where+if

```
 <select id="findUserByCondition" resultMap="userMap" parameterType="user">
      select * from user
      <where>
         <if test="userName != null">
            and username = #{userName}
         </if>
         <if test="userSex != null">
            and sex = #{userSex}
         </if>
      </where>
</select>
```

三、foreach

需求：我们需要查询 user 表中 id 分别为1,2,3的用户

sql语句：select * from user where id=1 or id=2 or id=3

select * from user where id in (1,2,3)

1、建立一个 UserVo 类，里面封装一个 List<Integer> ids 的属性

```
<select
id="selectUserByListId" parameterType="com.ys.vo.UserVo" resultType="com.ys.po.User">

   select * from user
   <where>
      <!--
         collection:指定输入对象中的集合属性
         item:每次遍历生成的对象
         open:开始遍历时的拼接字符串
         close:结束时拼接的字符串
         separator:遍历对象之间需要拼接的字符串
         select * from user where 1=1 and (id=1 or id=2 or id=3)
       -->
      <foreach collection="ids" item="id" open="and (" close=")" separator="or">
         id=#{id}
      </foreach>
   </where>
</select>
```

2、我们用 foreach 来改写 select * from user where id=1 or id=2 or id=3

```
<select
id="selectUserByListId" parameterType="com.ys.vo.UserVo" resultType="com.ys.po.User">

   select * from user
   <where>
      <!--
         collection:指定输入对象中的集合属性
         item:每次遍历生成的对象
         open:开始遍历时的拼接字符串
         close:结束时拼接的字符串
         separator:遍历对象之间需要拼接的字符串
         select * from user where 1=1 and (id=1 or id=2 or id=3)
       -->
      <foreach collection="ids" item="id" open="and (" close=")" separator="or">
         id=#{id}
      </foreach>
   </where>
</select>
```

3、用 foreach 来改写 select * from user where id in (1,2,3)

```
<select
id="selectUserByListId" parameterType="com.ys.vo.UserVo" resultType="com.ys.po.User">

      select * from user
      <where>
```

```
        <!--
            collection:指定输入对象中的集合属性
            item:每次遍历生成的对象
            open:开始遍历时的拼接字符串
            close:结束时拼接的字符串
            separator:遍历对象之间需要拼接的字符串
            select * from user where 1=1 and id in (1,2,3)
         -->
        <foreach collection="ids" item="id" open="and id in (" close=") " separator=",">
            #{id}
        </foreach>
    </where>
</select>
```

4、测试

```
//根据id集合查询user表数据
@Test
public void testSelectUserByListId(){
    String statement = "com.ys.po.userMapper.selectUserByListId";
    UserVo uv = new UserVo();
    List<Integer> ids = new ArrayList<>();
    ids.add(1);
    ids.add(2);
    ids.add(3);
    uv.setIds(ids);
    List<User> listUser = session.selectList(statement, uv);
    for(User u : listUser){
        System.out.println(u);
    }
    session.close();
}
```

四、include

```
<!-- 抽取重复的sql语句-->
    <sql id="defaultUser">
        select * from user
    </sql>


  <!-- 查询所有 -->
    <select id="findAll" resultMap="userMap">
        <include refid="defaultUser"></include>
    </select>
```

五、choose

有时候，我们不想用到所有的查询条件，只想选择其中的一个，查询条件有一个满足即可，使用 choose 标签可以解决此类问题，类似于 Java 的 switch 语句

```
<select
id="selectUserByChoose" resultType="com.ys.po.User" parameterType="com.ys.po.User">

    select * from user
    <where>
      <choose>
        <when test="id !='' and id != null">
          id=#{id}
        </when>
        <when test="username !='' and username != null">
          and username=#{username}
        </when>
        <otherwise>
          and sex=#{sex}
        </otherwise>
      </choose>
    </where>
  </select>
```

也就是说，这里我们有三个条件，id,username,sex，只能选择一个作为查询条件

如果 id 不为空，那么查询语句为：select * from user where id=?

如果 id 为空，那么看username 是否为空，如果不为空，那么语句为 select * from user where username=?;

如果 username 为空，那么查询语句为 select * from user where sex=?

六、set

用于update操作

```
<!-- 根据 id 更新 user 表的数据 -->
<update id="updateUserById" parameterType="com.ys.po.User">
   update user u
      <set>
        <if test="username != null and username != ''">
          u.username = #{username},
        </if>
        <if test="sex != null and sex != ''">
          u.sex = #{sex}
        </if>
      </set>
```

```
    where id=#{id}
</update>
```

七、trim

trim标记是一个格式化的标记，可以完成set或者是where标记的功能

1、用 trim 改写上面第二点的 if+where 语句

prefix：前缀

prefixoverride：去掉第一个and或者是or

```
<select
id="selectUserByUsernameAndSex" resultType="user" parameterType="com.ys.po.User">
      select * from user
      <!-- <where>
         <if test="username != null">
           username=#{username}
         </if>

         <if test="username != null">
           and sex=#{sex}
         </if>
      </where>  -->
      <trim prefix="where" prefixOverrides="and | or">
         <if test="username != null">
           and username=#{username}
         </if>
         <if test="sex != null">
           and sex=#{sex}
         </if>
      </trim>
   </select>
```

2、用 trim 改写上面第三点的 if+set 语句

suffix：后缀

suffixoverride：去掉最后一个逗号（也可以是其他的标记，就像是上面前缀中的and一样）

```
<!-- 根据 id 更新 user 表的数据 -->
   <update id="updateUserById" parameterType="com.ys.po.User">
      update user u
         <!-- <set>
            <if test="username != null and username != ''">
               u.username = #{username},
            </if>
            <if test="sex != null and sex != ''">
               u.sex = #{sex}
            </if>
```

```
        </set> -->
        <trim prefix="set" suffixOverrides=",">
            <if test="username != null and username != ''">
                u.username = #{username},
            </if>
            <if test="sex != null and sex != ''">
                u.sex = #{sex},
            </if>
        </trim>

    where id=#{id}
</update>
```