**动态** **SQL**

MyBatis 的一个强大的特性之一通常是它的动态 SQL 能力。 如果你有使用 JDBC 或其他 相似框架的经验,你就明白条件地串联 SQL 字符串在一起是多么的痛苦,确保不能忘了空 格或在列表的最后省略逗号。动态 SQL 可以彻底处理这种痛苦。

通常使用动态 SQL 不可能是独立的一部分,MyBatis 当然使用一种强大的动态 SQL 语 言来改进这种情形,这种语言可以被用在任意映射的 SQL 语句中。

* if
* choose (when, otherwise)
* trim (where, set)
* foreach

**if**

在动态 SQL 中所做的最通用的事情是包含部分 where 字句的条件。比如:

<select id="findActiveBlogWithTitleLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG

WHERE state = ‘ACTIVE’

<if test="title != null">

AND title like #{title}

</if>

</select>

这条语句会提供一个可选的文本查找功能。如果你没有传递 title,那么所有激活的博客 都会被返回。但是如果你传递了 title,那么就会查找相近的 title(对于敏锐的检索,这中情 况下你的参数值需要包含任意的遮掩或通配符)的博客。

假若我们想可选地搜索 title 和 author 呢?首先,要改变语句的名称让它有意义。然后 简单加入另外的一个条件。

<select id="findActiveBlogLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG WHERE state = ‘ACTIVE’

<if test="title != null">

AND title like #{title}

</if>

<if test="author != null and author.name != null">

AND author\_name like #{author.name}

</if>

</select>

**choose, when, otherwise**

有时我们不想应用所有的条件,相反我们想选择很多情况下的一种。 Java 中的 switch和 语句相似,MyBatis 提供 choose 元素。

我们使用上面的示例,但是现在我们来搜索当 title 提供时仅有 title 条件,当 author 提 供时仅有 author 条件。如果二者都没提供,只返回 featured blogs(也许是由管理员策略地选 择的结果列表,而不是返回大量没有意义的随机博客结果列表)。

<select id="findActiveBlogLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG WHERE state = ‘ACTIVE’

<choose>

<when test="title != null">

AND title like #{title}

</when>

<when test="author != null and author.name != null">

AND author\_name like #{author.name}

</when>

<otherwise>

AND featured = 1

</otherwise>

</choose>

</select>

**trim, where, set**

前面的例子已经方便地处理了一个臭名昭著的动态 SQL 问题。要考虑我们回到“if”示 例后会发生什么,但是这次我们将“ACTIVE = 1”也设置成动态的条件。

<select id="findActiveBlogLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG

WHERE

<if test="state != null">

state = #{state}

</if>

<if test="title != null">

AND title like #{title}

</if>

<if test="author != null and author.name != null">

AND author\_name like #{author.name}

</if>

</select>

如果这些条件都没有匹配上将会发生什么?这条 SQL 结束时就会成这样:

SELECT \* FROM BLOG

WHERE

这会导致查询失败。如果仅仅第二个条件匹配是什么样的?这条 SQL 结束时就会是这 样:

SELECT \* FROM BLOG

WHERE

AND title like ‘someTitle’

这个查询也会失败。这个问题不能简单的用条件来解决,如果你从来没有这样写过,那 么你以后也不会这样来写。

MyBatis 有一个简单的处理,这在 90%的情况下都会有用。而在不能使用的地方,你可 以自定义处理方式。加上一个简单的改变,所有事情都会顺利进行:

<select id="findActiveBlogLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG

<where>

<if test="state != null">

state = #{state}

</if>

<if test="title != null">

AND title like #{title}

</if>

<if test="author != null and author.name != null">

AND author\_name like #{author.name}

</if>

</where>

</select>

where 元素知道如果由被包含的标记返回任意内容,就仅仅插入“WHERE”。而且,如 果以“AND”或“OR”开头的内容,那么就会跳过 WHERE 不插入。

如果 where 元素没有做出你想要的,你可以使用 trim 元素来自定义。比如,和 where元素相等的 trim 元素是:

<trim prefix="WHERE" prefixOverrides="AND |OR ">

...

</trim>

prefixOverrides 属性采用管道文本分隔符来覆盖,这里的空白也是重要的。 它的结果就是移除 在 prefixOverrides 属性中指定的内容,插入在 with 属性中的内容。

和动态更新语句相似的解决方案是 set。set 元素可以被用于动态包含更新的列,而不包 含不需更新的。比如:

<update id="updateAuthorIfNecessary"

parameterType="domain.blog.Author">

update Author

<set>

<if test="username != null">username=#{username},</if>

<if test="password != null">password=#{password},</if>

<if test="email != null">email=#{email},</if>

<if test="bio != null">bio=#{bio}</if>

</set>

where id=#{id}

</update>

这里,set 元素会动态前置 SET 关键字,而且也会消除任意无关的逗号,那也许在应用 条件之后来跟踪定义的值。

如果你对和这相等的 trim 元素好奇,它看起来就是这样的:

<trim prefix="SET" suffixOverrides=",">

...

</trim>

注意这种情况下我们覆盖一个后缀,而同时也附加前缀。

**foreach**

另外一个动态 SQL 通用的必要操作是迭代一个集合,通常是构建在 IN 条件中的。 比如:

<select id="selectPostIn" resultType="domain.blog.Post">

SELECT \*

FROM POST P

WHERE ID in

<foreach item="item" index="index" collection="list"

open="(" separator="," close=")">

#{item}

</foreach>

</select>

foreach 元素是非常强大的,它允许你指定一个集合,声明集合项和索引变量,它们可 以用在元素体内。它也允许你指定开放和关闭的字符串,在迭代之间放置分隔符。这个元素 是很智能的,它不会偶然地附加多余的分隔符。

**注意** 你可以传递一个 List 实例或者数组作为参数对象传给 MyBatis。当你这么做的时 候,MyBatis 会自动将它包装在一个 Map 中,用名称在作为键。List 实例将会以“list”作为键,而数组实例将会以“array”作为键。