动态 SQL

MyBatis 最强大的功能之一始终是其动态 SQL 功能。如果你对 JDBC 或任何类似的框架有任何经验，你就会明白有条件地连接 SQL 字符串是多么痛苦，还要确保不忘记空格或在列列表末尾省略逗号。动态 SQL 可能非常痛苦。

虽然使用动态 SQL 永远不会是一场派对，但 MyBatis 肯定通过一种强大的动态 SQL 语言改善了这种情况，该语言可以在任何映射 SQL 语句中使用。

对于使用过 JSTL 或任何类似基于 XML 的文本处理器的任何人来说，动态 SQL 元素应该很熟悉。在 MyBatis 的早期版本中，有很多元素需要了解和理解。MyBatis 3 在此基础上得到了极大的改进，现在需要处理的元素不到这些元素的一半。MyBatis 采用了强大的基于 OGNL 的表达式来消除大多数其他元素

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

if

在动态 SQL 中最常见的事情是有条件地包含 where 子句的一部分。例如

```
<select id="findActiveBlogWithTitleLike"
     resultType="Blog">
  SELECT * FROM BLOG
  WHERE state = 'ACTIVE'
  <if test="title != null">
    AND title like #{title}
  </if>
</select>
```

此语句将提供可选文本搜索类型功能。如果你没有输入标题，则将返回所有活动的博客。但是，如果你输入标题，它将查找类似的标题（对于敏锐的人来说，是的，在这种情况下，你的参数值需要包括任何掩码或通配符）。

如果我们想要选择性地按标题和作者搜索怎么办？首先，我将更改语句的名称以使其更有意义。然后简单地添加另一个条件。

```
<select id="findActiveBlogLike"
     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

有时我们不希望应用所有条件，而是希望在众多选项中只选择一个情况。与 Java 中的 switch 语句类似，MyBatis 提供了一个 choose 元素。

让我们使用上面的示例，但现在如果提供了标题，则仅按标题搜索，如果提供了作者，则仅按作者搜索。如果两者都没有提供，让我们仅返回精选博客（可能是由管理员策略性地列出的，而不是返回大量无意义的随机博客列表）。

```
<select id="findActiveBlogLike"
     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

前面的示例一直在巧妙地回避一个臭名昭著的动态 SQL 挑战。考虑一下，如果我们回到我们的“if”示例，但这次我们将“ACTIVE = 1”也作为动态条件怎么办。

```
<select id="findActiveBlogLike"
     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>
```

如果未满足任何条件会怎样？你最终会得到类似这样的 SQL

```
SELECT * FROM BLOG
WHERE
```

这将失败。如果只有第二个条件满足，会怎样？您最终会得到类似这样的 SQL

```
SELECT * FROM BLOG
WHERE
AND title like 'someTitle'
```

这也会失败。此问题无法通过条件轻松解决，如果您曾经不得不编写它，那么您可能再也不想这么做了。

MyBatis 有一个简单的答案，在 90% 的情况下可能会起作用。在不起作用的情况下，您可以对其进行自定义，使其起作用。通过一个简单的更改，一切都正常工作

```
<select id="findActiveBlogLike"
     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>
```

如果包含标签返回任何内容，*where* 元素知道仅插入“WHERE”。此外，如果该内容以“AND”或“OR”开头，它知道将其剔离。

如果*where* 元素的行为与您希望的完全不同，您可以通过定义自己的 trim 元素来对其进行自定义。例如，*where* 元素的 trim 等效项为

```
<trim prefix="WHERE" prefixOverrides="AND |OR ">
  ...
</trim>
```

prefixOverrides 属性采用要覆盖的文本的管道分隔列表，其中空格是相关的。结果是删除*prefixOverrides* 属性中指定的所有内容，并插入*prefix* 属性中的所有内容。

对于称为*set* 的动态更新语句，有一个类似的解决方案。*set* 元素可用于动态包含要更新的列，并省略其他列。例如

```
<update id="updateAuthorIfNecessary">
  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>
```

在此，*set* 元素将动态前置 SET 关键字，并消除应用条件后可能拖尾值分配的任何无关逗号。

或者，您可以使用*trim* 元素来实现相同的效果

```
<trim prefix="SET" suffixOverrides=",">
  ...
</trim>
```

请注意，在这种情况下，我们正在覆盖后缀，而我们仍然附加前缀。

foreach

动态 SQL 的另一个常见需求是需要遍历集合，通常是为了构建 IN 条件。例如

```
<select id="selectPostIn" resultType="domain.blog.Post">
  SELECT *
  FROM POST P
  <where>
    <foreach item="item" index="index" collection="list"
        open="ID in (" separator="," close=")" nullable="true">
          #{item}
    </foreach>
  </where>
</select>
```

foreach 元素的用法非常简单。它允许您指定集合，声明可以在元素主体中使用的项和索引变量。它还允许您指定开始和结束字符串，并在迭代之间添加分隔符。该元素很聪明，不会意外附加额外的分隔符。

foreach 元素非常强大，它允许您指定集合，声明可以在元素主体内部使用的项和索引变量。它还允许您指定开始和结束字符串，并在迭代之间添加分隔符。该元素很智能，不会意外附加额外的分隔符。

注意 您可以将任何 Iterable 对象（例如 List、Set 等）以及任何 Map 或 Array 对象作为集合参数传递给 foreach。当使用 Iterable 或 Array 时，index 将是当前迭代的数字，value 项将是此迭代中检索的元素。当使用 Map（或 Map.Entry 对象的集合）时，index 将是键对象，item 将是值对象。

这结束了有关 XML 配置文件和 XML 映射文件的讨论。下一部分将详细讨论 Java API，以便您可以充分利用您创建的映射。

脚本

对于在带注释的映射器类中使用动态 SQL，可以使用 *script* 元素。例如

```
    @Update({"<script>",
      "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}",
      "</script>"})
    void updateAuthorValues(Author author);
```

绑定

`bind` 元素允许你创建一个 OGNL 表达式的变量并将其绑定到上下文。例如

```
<select id="selectBlogsLike" resultType="Blog">
  <bind name="pattern" value="'%' + _parameter.getTitle() + '%'" />
  SELECT * FROM BLOG
  WHERE title LIKE #{pattern}
</select>
```

多数据库供应商支持

如果配置了 databaseIdProvider，则“_databaseId”变量可用于动态代码，因此你可以根据数据库供应商构建不同的语句。请看以下示例

```
<insert id="insert">
  <selectKey keyProperty="id" resultType="int" order="BEFORE">
    <if test="_databaseId == 'oracle'">
      select seq_users.nextval from dual
    </if>
    <if test="_databaseId == 'db2'">
      select nextval for seq_users from sysibm.sysdummy1"
    </if>
  </selectKey>
  insert into users values (#{id}, #{name})
</insert>
```

动态 SQL 的可插入脚本语言

从版本 3.2 开始，MyBatis 支持可插入脚本语言，因此你可以插入一个语言驱动程序并使用该语言编写动态 SQL 查询。

你可以通过实现以下接口来插入语言

```
public interface LanguageDriver {
  ParameterHandler createParameterHandler(MappedStatement mappedStatement, Object parameterObject, BoundSql boundSql);
  SqlSource createSqlSource(Configuration configuration, XNode script, Class<?> parameterType);
  SqlSource createSqlSource(Configuration configuration, String script, Class<?> parameterType);
}
```

一旦你拥有自定义语言驱动程序，你可以通过在 mybatis-config.xml 文件中配置它来将其设置为默认值

```
<typeAliases>
  <typeAlias type="org.sample.MyLanguageDriver" alias="myLanguage"/>
</typeAliases>
<settings>
  <setting name="defaultScriptingLanguage" value="myLanguage"/>
</settings>
```

你可以通过添加 `lang` 属性来指定特定语句的语言，如下所示，而不是更改默认值

```
<select id="selectBlog" lang="myLanguage">
  SELECT * FROM BLOG
</select>
```

或者，如果你正在使用映射器，则使用 `@Lang` 注释

```
public interface Mapper {
  @Lang(MyLanguageDriver.class)
  @Select("SELECT * FROM BLOG")
  List<Blog> selectBlog();
}
```

注意 你可以使用 Apache Velocity 作为你的动态语言。有关详细信息，请查看 MyBatis-Velocity 项目。

你在前几节中看到的所有 xml 标记都是由 MyBatis 默认语言提供的，该语言由驱动程序 `org.apache.ibatis.scripting.xmltags.XmlLanguageDriver` 提供，该驱动程序别名为 `xml`。