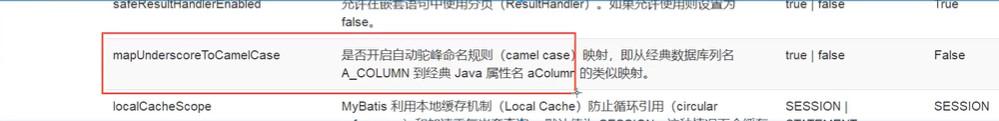
1、动态SQL：

什么是动态SQL：动态SQL就是指根据不同的条件生成不同的SQL语句。（**一个就是把公共的片段抽取出来，提高重用；另一个就是可以按照不同条件进行选取不同片段进行拼接**。）

主要是拼接简单，拼接过sql就知道解析拼接sql的关键字有多麻烦。

2、Java程序中我们常常使用驼峰命名法，但是在数据库中不区分大小写，所以数据库中的字段通常会自动全部转换为大写或者是小写。Mybatis中可以通过setting进行配置（但是注意点就是数据库字段名需要再驼峰位置添加上 \_ 进行划分）



3、动态SQL中的判断或者占位符中也能通过Java的形式进行取值（a.comC）、以及逻辑运算符或者是SQl的逻辑运算：



Trim([Mybatis之trim标签的理解 - WesTward - 博客园 (cnblogs.com)](https://www.cnblogs.com/westward/p/6706570.html))

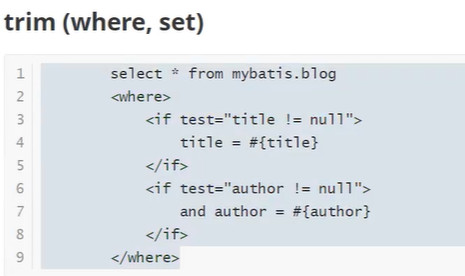
(trim标签的prefixOverrides和prefix分两步骤：

1.如果prefixOverrides有元素，拿元素去匹配 **第一个子标签**sql语句，若匹配上，就删掉sql语句的匹配部分，跳到2

2.如果prefix有值，就在  **第一个子标签**sql语句 的最前面加上 prefix的值。

)

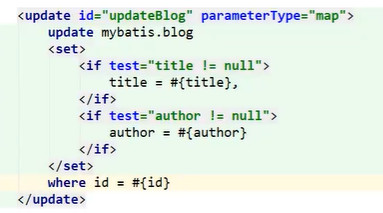


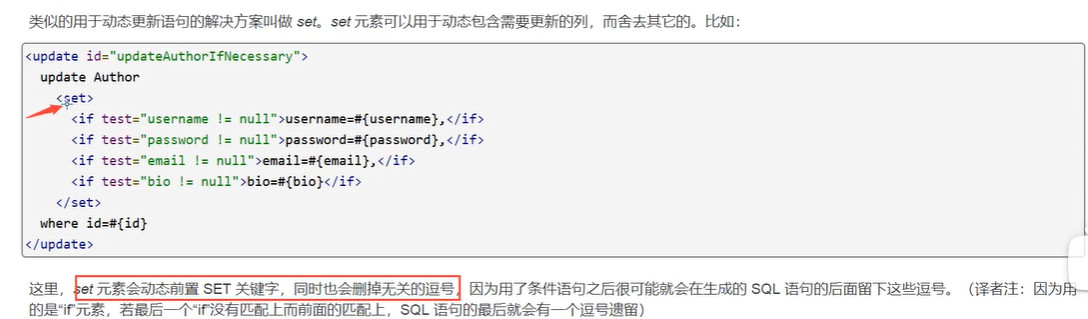


Where标签：



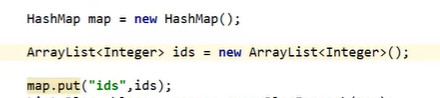
Set标签



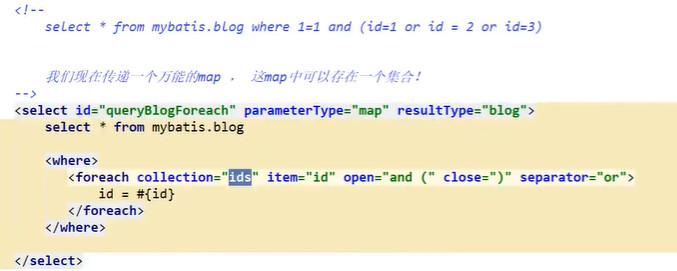


Foreach:





（注意这里可以直接通过名字取出传入对象中的属性对象）





动态SQL :





SQL片段尽量只包含一些判断就行了。



**script**

要在带注解的映射器接口类中使用动态 SQL，可以使用 *script* 元素。比如:

@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);

**bind**

bind 元素允许你在 OGNL 表达式以外创建一个变量，并将其绑定到当前的上下文。比如：

<select id="selectBlogsLike" resultType="Blog">

<bind name="pattern" value="'%' + \_parameter.getTitle() + '%'" />

SELECT \* FROM BLOG

WHERE title LIKE #{pattern}

</select>

**多数据库支持**

如果配置了 databaseIdProvider，你就可以在动态代码中使用名为 “\_databaseId” 的变量来为不同的数据库构建特定的语句。比如下面的例子：

<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 中的插入脚本语言**

MyBatis 从 3.2 版本开始支持插入脚本语言，这允许你插入一种语言驱动，并基于这种语言来编写动态 SQL 查询语句。

可以通过实现以下接口来插入一种语言：

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);

}

实现自定义语言驱动后，你就可以在 mybatis-config.xml 文件中将它设置为默认语言：

<typeAliases>

<typeAlias type="org.sample.MyLanguageDriver" alias="myLanguage"/>

</typeAliases>

<settings>

<setting name="defaultScriptingLanguage" value="myLanguage"/>

</settings>

或者，你也可以使用 lang 属性为特定的语句指定语言：

<select id="selectBlog" lang="myLanguage">

SELECT \* FROM BLOG

</select>

或者，在你的 mapper 接口上添加 @Lang 注解：

public interface Mapper {

@Lang(MyLanguageDriver.class)

@Select("SELECT \* FROM BLOG")

List<Blog> selectBlog();

}