

[首页](#) [资讯](#) [精华](#) [论坛](#) [问答](#) [博客](#) [专栏](#) [群组](#) [更多 ▼](#)  
[您还未登录！](#) [登录](#) [注册](#)

# 好好学习，天天向上

- [博客](#)
- [微博](#)
- [相册](#)
- [收藏](#)
- [留言](#)
- [关于我](#)

## MyBatis的动态SQL详解

博客分类：

- [mybatis](#)

### [MyBatis动态SQLDynamicforeach](#)

基础部分可以查看我的另一篇博客：<http://haohaoxuexi.iteye.com/blog/1333271>

MyBatis的动态SQL是基于OGNL表达式的，它可以帮助我们方便的在SQL语句中实现某些逻辑。

MyBatis中用于实现动态SQL的元素主要有：

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

if就是简单的条件判断，利用if语句我们可以实现某些简单的条件选择。先来看如下一个例子：

Xml代码 ☆

1. `<select id="dynamicIfTest" parameterType="Blog" resultType="Blog">`
2. `select * from t_blog where 11 = 1`
3. `<if test="title != null">`
4. `and title = #{title}`
5. `</if>`
6. `<if test="content != null">`

```
7.         and content = #{content}
8.     </if>
9.     <if test="owner != null">
10.         and owner = #{owner}
11.     </if>
12. </select>
```

这条语句的意思非常简单，如果你提供了title参数，那么就要满足title=#{title}，同样如果你提供了Content和Owner的时候，它们也需要满足相应的条件，之后就是返回满足这些条件的所有Blog，这是非常实用的一个功能，以往我们使用其他类型框架或者直接使用JDBC的时候，如果我们要达到同样的选择效果的时候，我们就需要拼SQL语句，这是极其麻烦的，比起来，上述的动态SQL就要简单多了。

**choose**元素的作用就相当于JAVA中的switch语句，基本上跟JSTL中的choose的作用和用法是一样的，通常都是与when和otherwise搭配的。看如下一个例子：

Xml代码 ☆

```
1. <select id="dynamicChooseTest" parameterType="Blog" resultType="Blog">
2.     select * from t_blog where 11 = 1
3.     <choose>
4.         <when test="title != null">
5.             and title = #{title}
6.         </when>
7.         <when test="content != null">
8.             and content = #{content}
9.         </when>
10.        <otherwise>
11.            and owner = "owner1"
12.        </otherwise>
13.    </choose>
14. </select>
```

when元素表示当when中的条件满足的时候就输出其中的内容，跟JAVA中的switch效果差不多的是按照条件的顺序，当when中有条件满足的时候，就会跳出choose，即所有的when和otherwise条件中，只有一个会输出，当所有的条件都不满足的时候就输出otherwise中的内容。所以上述语句的意思非常简单，当title!=null的时候就输出and title = #{title}，不再往下判断条件，当title为空且content!=null的时候就输出and content = #{content}，当所有条件都不满足的时候就输出otherwise中的内容。

**where**语句的作用主要是简化SQL语句中where中的条件判断的，先看一个例子，再解释一下where的好处。

Xml代码 ☆

```
1. <select id="dynamicWhereTest" parameterType="Blog" resultType="Blog">
2.     select * from t_blog
3.     <where>
4.         <if test="title != null">
```

```
5.         title = #{title}
6.     </if>
7.     <if test="content != null">
8.         and content = #{content}
9.     </if>
10.    <if test="owner != null">
11.        and owner = #{owner}
12.    </if>
13. </where>
14. </select>
```

where元素的作用是在写入where元素的地方输出一个where，另外一个好处是你不需要考虑where元素里面的条件输出是什么样子的，MyBatis会智能的帮你处理，如果所有的条件都不满足那么MyBatis就会查出所有的记录，如果输出后是and 开头的，MyBatis会把第一个and忽略，当然如果是or开头的，MyBatis也会把它忽略；此外，在where元素中你不需要考虑空格的问题，MyBatis会智能的帮你加上。像上述例子中，如果title=null，而content != null，那么输出的整个语句会是select \* from t\_blog where content = #{content}，而不是select \* from t\_blog where and content = #{content}，因为MyBatis会智能的把首个and 或 or 给忽略。

trim元素的主要功能是在自己包含的内容前加上某些前缀，也可以在其后加上某些后缀，与之对应的属性是prefix和suffix；可以把包含内容的首部某些内容覆盖，即忽略，也可以把尾部的某些内容覆盖，对应的属性是prefixOverrides和suffixOverrides；正因为trim有这样的功能，所以我们可以非常简单的利用trim来代替where元素的功能，示例代码如下：

Xml代码 ☆

```
1. <select id="dynamicTrimTest" parameterType="Blog" resultType="Blog">
2.     select * from t_blog
3.     <trim prefix="where" prefixOverrides="and |or">
4.         <if test="title != null">
5.             title = #{title}
6.         </if>
7.         <if test="content != null">
8.             and content = #{content}
9.         </if>
10.        <if test="owner != null">
11.            or owner = #{owner}
12.        </if>
13.    </trim>
14. </select>
```

set元素主要是用在更新操作的时候，它的主要功能和where元素其实是差不多的，主要是在包含的语句前输出一个set，然后如果包含的语句是以逗号结束的话将会把该逗号忽略，如果set包含的内容为空的话则会出错。有了set元素我们就可以动态的更新那些修改了的字段。下面是一段示例代码：

Xml代码 ☆

```

1. <update id="dynamicSetTest" parameterType="Blog">
2.     update t_blog
3.     <set>
4.         <if test="title != null">
5.             title = #{title},
6.         </if>
7.         <if test="content != null">
8.             content = #{content},
9.         </if>
10.        <if test="owner != null">
11.            owner = #{owner}
12.        </if>
13.    </set>
14.    where id = #{id}
15. </update>

```

上述示例代码中，如果set中一个条件都不满足，即set中包含的内容为空的时候就会报错。

**foreach**的主要用在构建in条件中，它可以在SQL语句中进行迭代一个集合。foreach元素的属性主要有item，index，collection，open，separator，close。item表示集合中每一个元素进行迭代时的别名，index指定一个名字，用于表示在迭代过程中，每次迭代到的位置，open表示该语句以什么开始，separator表示在每次进行迭代之间以什么符号作为分隔符，close表示以什么结束，在使用foreach的时候最关键的也是最容易出错的就是collection属性，该属性是必须指定的，但是在不同情况下，该属性的值是不一样的，主要有一下3种情况：

1. 如果传入的是单参数且参数类型是一个List的时候，collection属性值为list
2. 如果传入的是单参数且参数类型是一个array数组的时候，collection的属性值为array
3. 如果传入的参数是多个的时候，我们就需要把它们封装成一个Map了，当然单参数也可以封装成map，实际上如果你在传入参数的时候，在MyBatis里面也是会把它封装成一个Map的，map的key就是参数名，所以这个时候collection属性值就是传入的List或array对象在自己封装的map里面的key

下面分别来看看上述三种情况的示例代码：

1.单参数List的类型：

Xml代码 ☆

```

1. <select id="dynamicForeachTest" resultType="Blog">
2.     select * from t_blog where id in
3.     <foreach collection="list" index="index" item="item" open="(" separator="," close=")">
4.         #{item}
5.     </foreach>
6. </select>

```

上述collection的值为list，对应的Mapper是这样的

Java代码 ☆

```

1. public List<Blog> dynamicForeachTest(List<Integer> ids);

```

测试代码：

Java代码 ☆

```
1. @Test
2. public void dynamicForeachTest() {
3.     SqlSession session = Util.getSqlSessionFactory().openSession();
4.     BlogMapper blogMapper = session.getMapper(BlogMapper.class);
5.     List<Integer> ids = new ArrayList<Integer>();
6.     ids.add(1);
7.     ids.add(3);
8.     ids.add(6);
9.     List<Blog> blogs = blogMapper.dynamicForeachTest(ids);
10.    for (Blog blog : blogs)
11.        System.out.println(blog);
12.    session.close();
13. }
```

2.单参数array数组的类型：

Xml代码 ☆

```
1. <select id="dynamicForeach2Test" resultType="Blog">
2.     select * from t_blog where id in
3.     <foreach collection="array" index="index" item="item" open="(" separator="," close=")">
4.         #{item}
5.     </foreach>
6. </select>
```

上述collection为array，对应的Mapper代码：

Java代码 ☆

```
1. public List<Blog> dynamicForeach2Test(int[] ids);
```

对应的测试代码：

Java代码 ☆

```
1. @Test
2. public void dynamicForeach2Test() {
3.     SqlSession session = Util.getSqlSessionFactory().openSession();
4.     BlogMapper blogMapper = session.getMapper(BlogMapper.class);
5.     int[] ids = new int[] {1,3,6,9};
6.     List<Blog> blogs = blogMapper.dynamicForeach2Test(ids);
7.     for (Blog blog : blogs)
8.         System.out.println(blog);
9.     session.close();
10. }
```

### 3.自己把参数封装成Map的类型

Xml代码 ☆

```
1. <select id="dynamicForeach3Test" resultType="Blog">
2.     select * from t_blog where title like "%#{title}%" and id in
3.     <foreach collection="ids" index="index" item="item" open="(" separator="," close=")">
4.         #{item}
5.     </foreach>
6. </select>
```

上述collection的值为ids，是传入的参数Map的key，对应的Mapper代码：

Java代码 ☆

```
1. public List<Blog> dynamicForeach3Test(Map<String, Object> params);
```

对应测试代码：

Java代码 ☆



```
1. @Test
2. public void dynamicForeach3Test() {
3.     SqlSession session = Util.getSqlSessionFactory().openSession();
4.     BlogMapper blogMapper = session.getMapper(BlogMapper.class);
5.     final List<Integer> ids = new ArrayList<Integer>();
6.     ids.add(1);
7.     ids.add(2);
8.     ids.add(3);
9.     ids.add(6);
10.    ids.add(7);
11.    ids.add(9);
12.    Map<String, Object> params = new HashMap<String, Object>();
13.    params.put("ids", ids);
14.    params.put("title", "中国");
15.    List<Blog> blogs = blogMapper.dynamicForeach3Test(params);
16.    for (Blog blog : blogs)
17.        System.out.println(blog);
18.    session.close();
19. }
```

基础部分可以查看我的另一篇博客：<http://haohaoxuexi.iteye.com/blog/1333271>

顶

5

踩

分享到:  

[基于注解的SpringMVC简单介绍](#) | [MyBatis之ResultMap简介, 关联对象](#)

- 2012-01-09 12:08
- 浏览 122912
- [评论\(12\)](#)
- 分类: [编程语言](#)
- [相关推荐](#)

评论

12 楼 [yuang](#) 2015-12-29


□" target="\_blank">□" />□" target="\_blank">□" wmode="" quality="high" menu="false" pluginspage="http://www.macromedia.com/go/getflashplayer" type="application/x-shockwave-flash" width="200" height="200">

[b][b]



11 楼 [happymzw](#) 2014-12-04

Syhenian 写道

Java代码 

```
1. <update id="dynamicSetTest" parameterType="Blog">
2.     update t_blog
3.     <set>
4.         <if test="title != null">
5.             title = #{title},
6.         </if>
7.         <if test="content != null">
8.             content = #{content},
9.         </if>
10.        <if test="owner != null">
11.            owner = #{owner}
12.        </if>
13.    </set>
14.    where id = #{id}
15. </update>
```

这段可以这样写吗 如果ownner等于null 那语句就是

update t\_blog title= 'title', content='content', where id='id'

多了个红色的逗号不会报错?

楼主说了，set会自动处理掉后面的那个逗号的

10 楼 [简单就是美了](#) 2014-11-20



9 楼 [shiping8000](#) 2014-11-06

写的不错，给赞

8 楼 [Syhenian](#) 2014-08-29

Java代码

```
1. <update id="dynamicSetTest" parameterType="Blog">
2.     update t_blog
3.     <set>
4.         <if test="title != null">
5.             title = #{title},
6.         </if>
7.         <if test="content != null">
8.             content = #{content},
9.         </if>
10.        <if test="owner != null">
11.            owner = #{owner}
12.        </if>
13.    </set>
14.    where id = #{id}
15. </update>
```

这段可以这样写吗 如果owner等于null 那语句就是

update t\_blog title= 'title', content='content', where id='id'

多了个红色的逗号不会报错？

7 楼 [slc2006119](#) 2014-06-27

简单，易懂，确实写的不错，赞一个！

6 楼 [dmewy](#) 2014-06-12

不错 适合开发初期看

5 楼 [宋建勇](#) 2014-05-28

今天整合MyBatis，帮了大忙了啊！不错啊！

4 楼 [18289753290](#) 2014-05-23

不错

3 楼 [lcg19880128](#) 2014-03-14



2 楼 [coffeeBoy2012](#) 2014-02-07

很详细

1 楼 [huxiulei](#) 2013-11-22

嗯 写的不错... 简明易懂





## 发表评论



[您还没有登录,请您登录后再发表评论](#)



234390216

- 浏览: 2175213 次
- 性别:
- 来自: 长沙
- 我现在离线

## 最近访客

[更多访客>>](#)



[canzyp](#)



[陈轻尘](#)



[天天来注册](#)



[endless](#)

## 博客专栏

ORACLE

[Oracle基础](#)

浏览量：55408



[springMVC介绍](#)

浏览量：562681



### [Mybatis简介](#)

浏览量 : 546455



### [Spring整合JMS](#)

浏览量 : 117405



### [Ehcache简介](#)

浏览量 : 73385



### [Cas简介](#)

浏览量 : 27651



### [Spring Security...](#)

浏览量 : 94759

## 文章分类

- [全部博客 \(218\)](#)
- [配置文件 \(2\)](#)
- [dwr \(2\)](#)
- [安全 \(1\)](#)
- [电子邮件 \(3\)](#)
- [cxf \(7\)](#)
- [Cas \(10\)](#)
- [CKEDITOR \(3\)](#)
- [ehcache \(10\)](#)
- [extjs4 \(6\)](#)
- [freemarker \(2\)](#)
- [hibernate \(5\)](#)
- [itext \(1\)](#)
- [JasperReport \(4\)](#)
- [java \(19\)](#)
- [jbpm4 \(2\)](#)
- [jfreeChart \(3\)](#)
- [Jms \(5\)](#)



- [jquery \(7\)](#)
- [jsp \(4\)](#)
- [Linux \(1\)](#)
- [Lucene \(1\)](#)
- [maven \(7\)](#)
- [mybatis \(9\)](#)
- [MySQL \(5\)](#)
- [oracle \(29\)](#)
- [poi \(7\)](#)
- [日志 \(1\)](#)
- [Servlet \(4\)](#)
- [实用技巧 \(5\)](#)
- [Spring \(12\)](#)
- [SpringMVC \(13\)](#)
- [spring Security \(20\)](#)
- [struts2 \(5\)](#)
- [swing \(1\)](#)
- [svn \(1\)](#)
- [Tomcat \(1\)](#)
- [web前端 \(5\)](#)
- [Weblogic \(4\)](#)
- [问题 \(2\)](#)
- [xml \(2\)](#)
- [面试 \(1\)](#)

#### 社区版块

- [我的资讯 \(0\)](#)
- [我的论坛 \(17\)](#)
- [我的问答 \(4\)](#)

#### 存档分类

- [2016-01 \(2\)](#)
- [2015-12 \(6\)](#)
- [2015-10 \(6\)](#)
- [更多存档...](#)

#### 评论排行榜

- [Spring Security \(16\) ——基于表达式的权 ...](#)
- [Spring Security \(18\) ——Jsp标签](#)
- [Spring Security \(17\) ——基于方法的权限 ...](#)
- [Spring Security \(14\) ——权限鉴定基础](#)
- [Spring Security \(19\) ——对Acl的支持](#)



## 最新评论

- [z401111207](#) : 关于在 while 循环 read() 过程中按博主的意思整理 ...  
[Java Socket编程](#)
- [zhongwen7710](#) :  
[Maven简介 \(五\) ——pom.xml](#)
- [a87604476](#) : 楼主，这句话什么意思？timeToldle、timeToLiv ...  
[Spring使用Cache、整合Ehcache](#)
- [a87604476](#) : testAdd()方法中直接addCache("te ...  
[Ehcache \(06\) ——监听器](#)
- [huochuner](#) : 讲的真是太好了，有如醍醐灌顶。  
[基于注解的SpringMVC简单介绍](#)

---

声明：ITeye文章版权属于作者，受法律保护。没有作者书面许可不得转载。若作者同意转载，必须以超链接形式标明文章原始出处和作者。

© 2003-2016 ITeye.com. All rights reserved. [ 京ICP证110151号 京公网安备110105010620 ]