thymeleaf 基本语法

四、标准表达式语法

· 简单表达式 （simple expressions）

${...} 变量表达式

*{...} 选择变量表达式

#{...} 消息表达式

@{...} 链接url表达式

· 字面量

'one text','another one!',... 文本

0,34,3.0,12.3,... 数值

true false 布尔类型

null 空

one,sometext,main 文本字符

· 文本操作

+ 字符串连接

|The name is ${name}| 字符串连接

· 算术运算

+ , - , * , / , % 二元运算符

- 负号（一元运算符）

· 布尔操作

and,or 二元操作符

!,not 非（一元操作符）

· 关系操作符

> , < , >= , <= (gt , lt , ge , le)

== , != (eq, ne)

· 条件判断

(if) ? (then) if-then

(if) ? (then) : (else) if-then-else

```
<tr th:class="${row.even}? 'even' : 'odd'">
  ...
</tr>
```

条件表达式中的三个部分自身也可以是表达式，也可以是变量(${...}, *{...}), 消息(#{...}), URL (@{...}) 或字面量('...')
条件表达式也可以使用括号来嵌套：

```
<tr th:class="${row.even}? (${row.first}? 'first' : 'even') : 'odd'">
...
</tr>
```

else表达式也可以省略，当条件为false时，会返回null：

```
<tr th:class="${row.even}? 'alt'">
...
</tr>
```

(value) ?: (defaultvalue) Default

只有在第一个表达式返回null时，第二个表达式才会运算

·表达式工具对象

- #dates 与java.util.Date对象的方法对应，格式化、日期组件抽取等等
- #calendars 类似#dates，与java.util.Calendar对象对应
- #numbers 格式化数字对象的工具方法
- #strings 与java.lang.String对应的工具方法：contains、startsWith、prepending/appending等等
- #objects 用于对象的工具方法
- #bools 用于布尔运算的工具方法
- #arrays 用于数组的工具方法
- #lists 用于列表的工具方法
- #sets 用于set的工具方法
- #maps 用于map的工具方法
- #aggregates 用于创建数组或集合的聚合的工具方法
- #messages 用于在变量表达式内部获取外化消息的工具方法，与#{...}语法获取的方式相同
- #ids 用于处理可能重复出现（例如，作为遍历的结果）的id属性的工具方法

·链接URL

URL在web模板中是一级重要元素，使用@{...}表示

URL的类型：

绝对URL：

- http://www.thymeleaf.org

相对URL：

- 页面相对： user/login.html
- 上下文相对：/itemdetails?id=3 (服务器上下文名称会被自动添加)
- 服务器相对：~/billing/processInvoice（允许调用同一服务器上的另一个上下文中的URL）
- 协议相对：//code.jquery.com/jquery-2.0.3.min.js

Thymeleaf在任何情况下都可以处理绝对URL，对于相对URL，则需要使用一个实现了IWebContext接口的上下文对象，这个对象包含了来自HTTP请求的信息，这些信息用于创建相对链接。

2016年4月 (2)
2016年3月 (2)
2016年2月 (2)

```
<!-- Will produce 'http://localhost:8080/gtvg/order/details?orderId=3' (plus rewriting) -->
<a href="details.html"
th:href="@{http://localhost:8080/gtvg/order/details(orderId=${o.id})}">view</a>

<!-- Will produce '/gtvg/order/details?orderId=3' (plus rewriting) -->
<a href="details.html" th:href="@{/order/details(orderId=${o.id})}">view</a>

<!-- Will produce '/gtvg/order/3/details' (plus rewriting) -->
<a href="details.html" th:href="@{/order/{orderId}/details(orderId=${o.id})}">view</a>
```

· 预处理

Thymeleaf提供预处理表达式的功能。

它是在表壳式正常执行前执行的操作，允许修改最终将要被执行的表达式。

预处理表达式跟正常的一样，但被两个下划线包围住，例如：__${expression}__

假设有一个i18n消息文件Message_fr.properties，里面有一个条目包含了一个调用具体语言的静态方法的OGNL表达式：

```
article.text=@myapp.translator.Translator@translateToFrench({0})
```

Messages_es.properties中的等价条目：

```
article.text=@myapp.translator.Translator@translateToSpanish({0})
```

可以根据locale先创建用于运算表达式的标记片段，本例中，先通过预处理选择表达式，然后让Thymeleaf处理这个选择出来的表达式：

```
<p th:text="${__#{article.text('textVar')}__}">Some text here...</p>
```

对于locale为French的情况，上面的表达式经过预处理后，得出的等价物如下：

```
<p th:text="${@myapp.translator.Translator@translateToFrench(textVar)}">Some text here...</p>
```

五、 设置属性值

th:attr 任何属性值

```
<form action="subscribe.html" th:attr="action=@{/subscribe}">
   <fieldset>
      <input type="text" name="email" />
      <input type="submit" value="Subscribe me!" th:attr="value=#{subscribe.submit}"/>
   </fieldset>
</form>
```

多个属性一起设置，用逗号隔开

```
<img src="../../images/gtvglogo.png" th:attr="src=@{/images/gtvglogo.png},title=#{logo},alt=#
{logo}" />
```

设置指定属性

```
th:abbr th:accept th:accept-charset
th:accesskey th:action th:align
th:alt th:archive th:audio
th:autocomplete th:axis th:background
th:bgcolor th:border th:cellpadding
th:cellspacing th:challenge th:charset
th:cite th:class th:classid ...
```

```
<input type="submit" value="Subscribe me!" th:value="#{subscribe.submit}"/>
```

```
<form action="subscribe.html" th:action="@{/subscribe}">
```

```
<li><a href="product/list.html" th:href="@{/product/list}">Product List</a></li>
```

设置多个属性在同一时间 有两个特殊的属性可以这样设置： **th:alt-title 和 th:lang-xmllang**

th:alt-title 设置 alt 和 title
th:lang-xmllang 设置 lang 和 xml:lang

```
<img src="../../images/gtvglogo.png" th:attr="src=@{/images/gtvglogo.png},title=#{logo},alt=#{logo}" />

<img src="../../images/gtvglogo.png"th:src="@{/images/gtvglogo.png}" th:title="#{logo}" th:alt="#{logo}" />

<img src="../../images/gtvglogo.png"th:src="@{/images/gtvglogo.png}" th:alt-title="#{logo}" />
```

前置和后置添加属性值 **th:attrappend 和 th:attrprepend**

```
<input type="button" value="Do it!" class="btn" th:attrappend="class=${' ' + cssStyle}" />
```

编译后：

```
<input type="button" value="Do it!" class="btn warning" />
```

还有两个特定的添加属性 **th:classappend 和 th:styleappend**

```
<tr th:each="prod : ${prods}" class="row" th:classappend="${prodStat.odd}? 'odd'">
```

修复的布尔属性

```
<input type="checkbox" name="active" th:checked="${user.active}" />
```

所有修复的布尔属性：

```
|th:async |th:autofocus |th:autoplay |

|th:checked |th:controls |th:declare |

|th:default |th:defer |th:disabled |

|th:formnovalidate|th:hidden |th:ismap |

|th:loop |th:multiple |th:novalidate |

|th:nowrap |th:open |th:pubdate |

|th:readonly |th:required |th:reversed |

|th:scoped |th:seamless |th:selected |
```

HTML5友好的属性及元素名

```
<table>
    <tr data-th-each="user : ${users}">
```

```
        <td data-th-text="${user.login}">...</td>
        <td data-th-text="${user.name}">...</td>
    </tr>
</table>
```

data-{prefix}-{name}是编写HTML5自定义属性的标准语法，不需要开发者使用th:*这样的命名空间，Thymeleaf让这种语法自动对所有dialect都可用。

六、遍历

·基础

```
<tr th:each="prod : ${prods}">
    <td th:text="${prod.name}">Onions</td>
    <td th:text="${prod.price}">2.41</td>
    <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
</tr>
```

可遍历的对象：实现java.util.Iterable、java.util.Map（遍历时取java.util.Map.Entry）、array、任何对象都被当作只有对象自身一个元素的列表

·状态

- 当前遍历索引，从0开始，index属性
- 当前遍历索引，从1开始，count属性
- 总元素数量，size属性
- 每一次遍历的iter变量,current属性
- 当前遍历是even还是odd，even/odd布尔属性
- 当前遍历是第一个，first布尔属性
- 当前遍历是最后一个，last布尔属性

```
<tr th:each="prod,iterStat : ${prods}" th:class="${iterStat.odd}? 'odd'">
    <td th:text="${prod.name}">Onions</td>
    <td th:text="${prod.price}">2.41</td>
    <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
</tr>
```

若不指定状态变量，Thymeleaf会默认生成一个名为“变量名Stat”的状态变量：

```
<tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">
    <td th:text="${prod.name}">Onions</td>
    <td th:text="${prod.price}">2.41</td>
    <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
</tr>
```

七、条件运算

```
<tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">
    <td th:text="${prod.name}">Onions</td>
    <td th:text="${prod.price}">2.41</td>
    <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
    <td>
      <span th:text="${#lists.size(prod.comments)}">2</span> comment/s
      <a href="comments.html" th:href="@{/product/comments(prodId=${prod.id})}" th:if="${not
#lists.isEmpty(prod.comments)}">view</a>
    </td>
</tr>
```

```
<a href="comments.html" th:href="@{/product/comments(prodId=${prod.id})}" th:if="${not
#lists.isEmpty(prod.comments)}">view</a>
```

th:if 不只运算布尔条件，它对以下情况也运算为true:

·**值不为null**

值为boolean且为true

值为数字且非0

值为字符且非0

值是字符串且不是：“false”,“off”，“no”

值不是boolean、数字、字符、字符串

·**如果值为null，则th:if运算结果为false**

th:if的反面是**th:unless**

```
<a href="comments.html" th:href="@{/comments(prodId=${prod.id})}"
th:unless="${#lists.isEmpty(prod.comments)}">view</a>
```

th:switch 和 th:case

```
<div th:switch="${user.role}">
    <p th:case="'admin'">User is an administrator</p>
    <p th:case="#{roles.manager}">User is a manager</p>
</div>

<div th:switch="${user.role}">
    <p th:case="'admin'">User is an administrator</p>
    <p th:case="#{roles.manager}">User is a manager</p>
    <p th:case="*">User is some other thing</p>
</div>
```


nuoyi
关注 - 18
粉丝 - 7
+加关注

6 0

« 上一篇：scroll
» 下一篇：thymeleaf 模板布局

posted @ 2016-06-17 13:10 nuoyi 阅读(114080) 评论(2) 编辑 收藏

评论列表

#1楼 2016-08-02 16:15 **广寒宫第一萌宠的菜地**

整理的很棒！！！赞

支持(0) 反对(0)

#2楼 2018-09-07 13:34 **我最洋气**

学过el的应该看一眼练练就会，只是这个地址比el方便一些

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

相关博文:
· thymeleaf语法
· thymeleaf 内联语法
· Thymeleaf使用&语法
· Thymeleaf 基础语法
· Thymeleaf-语法整理

最新新闻:
· 网约车第一股Lyft给全球市场带来了哪些提示
· GNU Linux-libre 5.0正式发布：不包含任何专有代码
· 罗永浩的电子烟，治不好于大爷的瘾
· 乐视网：公司存在股票被暂停上市的风险
· 两会企业家代表和委员都在读什么书?
» 更多新闻...