springboot(四)：thymeleaf使用详解

转自：http://www.ityouknow.com/springboot/2016/05/01/spring-boot-thymeleaf.html

在上篇文章[springboot(二)：web综合开发](http://www.ityouknow.com/springboot/2016/02/03/spring-boot-web.html)中简单介绍了一下thymeleaf，这篇文章将更加全面详细的介绍thymeleaf的使用。thymeleaf 是新一代的模板引擎，在spring4.0中推荐使用thymeleaf来做前端模版引擎。

**thymeleaf介绍**

简单说， Thymeleaf 是一个跟 Velocity、FreeMarker 类似的模板引擎，它可以完全替代 JSP 。相较与其他的模板引擎，它有如下三个极吸引人的特点：

* 1.Thymeleaf 在有网络和无网络的环境下皆可运行，即它可以让美工在浏览器查看页面的静态效果，也可以让程序员在服务器查看带数据的动态页面效果。这是由于它支持 html 原型，然后在 html 标签里增加额外的属性来达到模板+数据的展示方式。浏览器解释 html 时会忽略未定义的标签属性，所以 thymeleaf 的模板可以静态地运行；当有数据返回到页面时，Thymeleaf 标签会动态地替换掉静态内容，使页面动态显示。
* 2.Thymeleaf 开箱即用的特性。它提供标准和spring标准两种方言，可以直接套用模板实现JSTL、 OGNL表达式效果，避免每天套模板、该jstl、改标签的困扰。同时开发人员也可以扩展和创建自定义的方言。
* 3.Thymeleaf 提供spring标准方言和一个与 SpringMVC 完美集成的可选模块，可以快速的实现表单绑定、属性编辑器、国际化等功能。

**标准表达式语法**

它们分为四类：

* 1.变量表达式
* 2.选择或星号表达式
* 3.文字国际化表达式
* 4.URL表达式

**变量表达式**

变量表达式即OGNL表达式或Spring EL表达式(在Spring术语中也叫model attributes)。如下所示：  
${session.user.name}

它们将以HTML标签的一个属性来表示：

<span th:text="${book.author.name}">

<li th:each="book : ${books}">

**选择(星号)表达式**

选择表达式很像变量表达式，不过它们用一个预先选择的对象来代替上下文变量容器(map)来执行，如下：  
 \*{customer.name}

被指定的object由th:object属性定义：

<div th:object="${book}">

...

<span th:text="\*{title}">...</span>

...

</div>

**文字国际化表达式**

文字国际化表达式允许我们从一个外部文件获取区域文字信息(.properties)，用Key索引Value，还可以提供一组参数(可选).

#{main.title}

#{message.entrycreated(${entryId})}

可以在模板文件中找到这样的表达式代码：

<table>

...

<th th:text="#{header.address.city}">...</th>

<th th:text="#{header.address.country}">...</th>

...

</table>

**URL表达式**

URL表达式指的是把一个有用的上下文或回话信息添加到URL，这个过程经常被叫做URL重写。  
 @{/order/list}   
URL还可以设置参数：  
 @{/order/details(id=${orderId})}    
相对路径：  
 @{../documents/report}

让我们看这些表达式：

<form th:action="@{/createOrder}">

<a href="main.html" th:href="@{/main}">

**变量表达式和星号表达有什么区别吗？**

如果不考虑上下文的情况下，两者没有区别；星号语法评估在选定对象上表达，而不是整个上下文   
什么是选定对象？就是父标签的值，如下：

<div th:object="${session.user}">

<p>Name: <span th:text="\*{firstName}">Sebastian</span>.</p>

<p>Surname: <span th:text="\*{lastName}">Pepper</span>.</p>

<p>Nationality: <span th:text="\*{nationality}">Saturn</span>.</p>

</div>

这是完全等价于：

<div th:object="${session.user}">

<p>Name: <span th:text="${session.user.firstName}">Sebastian</span>.</p>

<p>Surname: <span th:text="${session.user.lastName}">Pepper</span>.</p>

<p>Nationality: <span th:text="${session.user.nationality}">Saturn</span>.</p>

</div>

当然，美元符号和星号语法可以混合使用：

<div th:object="${session.user}">

<p>Name: <span th:text="\*{firstName}">Sebastian</span>.</p>

<p>Surname: <span th:text="${session.user.lastName}">Pepper</span>.</p>

<p>Nationality: <span th:text="\*{nationality}">Saturn</span>.</p>

</div>

**表达式支持的语法**

**字面（Literals）**

* 文本文字（Text literals）: 'one text', 'Another one!',…
* 数字文本（Number literals）: 0, 34, 3.0, 12.3,…
* 布尔文本（Boolean literals）: true, false
* 空（Null literal）: null
* 文字标记（Literal tokens）: one, sometext, main,…

**文本操作（Text operations）**

* 字符串连接(String concatenation): +
* 文本替换（Literal substitutions）: |The name is ${name}|

**算术运算（Arithmetic operations）**

* 二元运算符（Binary operators）: +, -, \*, /, %
* 减号（单目运算符）Minus sign (unary operator): -

**布尔操作（Boolean operations）**

* 二元运算符（Binary operators）:and, or
* 布尔否定（一元运算符）Boolean negation (unary operator):!, not

**比较和等价(Comparisons and equality)**

* 比较（Comparators）: >, <, >=, <= (gt, lt, ge, le)
* 等值运算符（Equality operators）:==, != (eq, ne)

**条件运算符（Conditional operators）**

* If-then: (if) ? (then)
* If-then-else: (if) ? (then) : (else)
* Default: (value) ?: (defaultvalue)

所有这些特征可以被组合并嵌套：

'User is of type ' + (${user.isAdmin()} ? 'Administrator' : (${user.type} ?: 'Unknown'))

**常用th标签都有那些？**

| **关键字** | **功能介绍** | **案例** |
| --- | --- | --- |
| th:id | 替换id | <input th:id="'xxx' + ${collect.id}"/> |
| th:text | 文本替换 | <p th:text="${collect.description}">description</p> |
| th:utext | 支持html的文本替换 | <p th:utext="${htmlcontent}">conten</p> |
| th:object | 替换对象 | <div th:object="${session.user}"> |
| th:value | 属性赋值 | <input th:value="${user.name}" /> |
| th:with | 变量赋值运算 | <div th:with="isEven=${prodStat.count}%2==0"></div> |
| th:style | 设置样式 | th:style="'display:' + @{(${sitrue} ? 'none' : 'inline-block')} + ''" |
| th:onclick | 点击事件 | th:onclick="'getCollect()'" |
| th:each | 属性赋值 | tr th:each="user,userStat:${users}"> |
| th:if | 判断条件 | <a th:if="${userId == collect.userId}" > |
| th:unless | 和th:if判断相反 | <a th:href="@{/login}" th:unless=${session.user != null}>Login</a> |
| th:href | 链接地址 | <a th:href="@{/login}" th:unless=${session.user != null}>Login</a> /> |
| th:switch | 多路选择 配合th:case 使用 | <div th:switch="${user.role}"> |
| th:case | th:switch的一个分支 | <p th:case="'admin'">User is an administrator</p> |
| th:fragment | 布局标签，定义一个代码片段，方便其它地方引用 | <div th:fragment="alert"> |
| th:include | 布局标签，替换内容到引入的文件 | <head th:include="layout :: htmlhead" th:with="title='xx'"></head> /> |
| th:replace | 布局标签，替换整个标签到引入的文件 | <div th:replace="fragments/header :: title"></div> |
| th:selected | selected选择框 选中 | th:selected="(${xxx.id} == ${configObj.dd})" |
| th:src | 图片类地址引入 | <img class="img-responsive" alt="App Logo" th:src="@{/img/logo.png}" /> |
| th:inline | 定义js脚本可以使用变量 | <script type="text/javascript" th:inline="javascript"> |
| th:action | 表单提交的地址 | <form action="subscribe.html" th:action="@{/subscribe}"> |
| th:remove | 删除某个属性 | <tr th:remove="all"> 1.all:删除包含标签和所有的孩子。2.body:不包含标记删除,但删除其所有的孩子。3.tag:包含标记的删除,但不删除它的孩子。4.all-but-first:删除所有包含标签的孩子,除了第一个。5.none:什么也不做。这个值是有用的动态评估。 |
| th:attr | 设置标签属性，多个属性可以用逗号分隔 | 比如 th:attr="src=@{/image/aa.jpg},title=#{logo}"，此标签不太优雅，一般用的比较少。 |

还有非常多的标签，这里只列出最常用的几个,由于一个标签内可以包含多个th:x属性，其生效的优先级顺序为: include,each,if/unless/switch/case,with,attr/attrprepend/attrappend,value/href,src ,etc,text/utext,fragment,remove。

**几种常用的使用方法**

**1、赋值、字符串拼接**

<p th:text="${collect.description}">description</p>

<span th:text="'Welcome to our application, ' + ${user.name} + '!'">

字符串拼接还有另外一种简洁的写法

<span th:text="|Welcome to our application, ${user.name}!|">

**2、条件判断 If/Unless**

Thymeleaf中使用th:if和th:unless属性进行条件判断，下面的例子中，<a>标签只有在th:if中条件成立时才显示：

<a th:if="${myself=='yes'}" > </i> </a>

<a th:unless=${session.user != null} th:href="@{/login}" >Login</a>

th:unless于th:if恰好相反，只有表达式中的条件不成立，才会显示其内容。

也可以使用 (if) ? (then) : (else) 这种语法来判断显示的内容

**3、for 循环**

<tr th:each="collect,iterStat : ${collects}">

<th scope="row" th:text="${collect.id}">1</th>

<td >

<img th:src="${collect.webLogo}"/>

</td>

<td th:text="${collect.url}">Mark</td>

<td th:text="${collect.title}">Otto</td>

<td th:text="${collect.description}">@mdo</td>

<td th:text="${terStat.index}">index</td>

</tr>

iterStat称作状态变量，属性有：

* index:当前迭代对象的index（从0开始计算）
* count: 当前迭代对象的index(从1开始计算)
* size:被迭代对象的大小
* current:当前迭代变量
* even/odd:布尔值，当前循环是否是偶数/奇数（从0开始计算）
* first:布尔值，当前循环是否是第一个
* last:布尔值，当前循环是否是最后一个

**4、URL**

URL在Web应用模板中占据着十分重要的地位，需要特别注意的是Thymeleaf对于URL的处理是通过语法@{…}来处理的。 如果需要Thymeleaf对URL进行渲染，那么务必使用th:href，th:src等属性，下面是一个例子

*<!-- Will produce 'http://localhost:8080/standard/unread' (plus rewriting) -->*

<a th:href="@{/standard/{type}(type=${type})}">view</a>

*<!-- Will produce '/gtvg/order/3/details' (plus rewriting) -->*

<a href="details.html" th:href="@{/order/{orderId}/details(orderId=${o.id})}">view</a>

设置背景

<div th:style="'background:url(' + @{/<path-to-image>} + ');'"></div>

根据属性值改变背景

<div class="media-object resource-card-image" th:style="'background:url(' + @{(${collect.webLogo}=='' ? 'img/favicon.png' : ${collect.webLogo})} + ')'" ></div>

几点说明：

* 上例中URL最后的(orderId=${o.id}) 表示将括号内的内容作为URL参数处理，该语法避免使用字符串拼接，大大提高了可读性
* @{...}表达式中可以通过{orderId}访问Context中的orderId变量
* @{/order}是Context相关的相对路径，在渲染时会自动添加上当前Web应用的Context名字，假设context名字为app，那么结果应该是/app/order

**5、内联js**

内联文本：[[…]]内联文本的表示方式，使用时，必须先用th:inline=”text/javascript/none”激活，th:inline可以在父级标签内使用，甚至作为body的标签。内联文本尽管比th:text的代码少，不利于原型显示。

**<**script th:inline**=**"javascript"**>**

*/\*<![CDATA[\*/*

...

**var** username **=** */\*[[${sesion.user.name}]]\*/* 'Sebastian';

**var** size **=** */\*[[${size}]]\*/* 0;

...

*/\*]]>\*/*

**<**/script>

js附加代码：

*/\*[+*

*var msg = 'This is a working application';*

*+]\*/*

js移除代码：

*/\*[- \*/*

**var** msg **=** 'This is a non-working template';

*/\* -]\*/*

**6、内嵌变量**

为了模板更加易用，Thymeleaf还提供了一系列Utility对象（内置于Context中），可以通过#直接访问：

* dates ： *java.util.Date的功能方法类。*
* calendars : *类似#dates，面向java.util.Calendar*
* numbers : *格式化数字的功能方法类*
* strings : *字符串对象的功能类，contains,startWiths,prepending/appending等等。*
* objects: *对objects的功能类操作。*
* bools: *对布尔值求值的功能方法。*
* arrays：*对数组的功能类方法。*
* lists: *对lists功能类方法*
* sets
* maps  
  …

下面用一段代码来举例一些常用的方法：

**dates**

/\*

\* Format date with the specified pattern

\* Also works with arrays, lists or sets

\*/

${#dates.format(date, 'dd/MMM/yyyy HH:mm')}

${#dates.arrayFormat(datesArray, 'dd/MMM/yyyy HH:mm')}

${#dates.listFormat(datesList, 'dd/MMM/yyyy HH:mm')}

${#dates.setFormat(datesSet, 'dd/MMM/yyyy HH:mm')}

/\*

\* Create a date (java.util.Date) object for the current date and time

\*/

${#dates.createNow()}

/\*

\* Create a date (java.util.Date) object for the current date (time set to 00:00)

\*/

${#dates.createToday()}

**strings**

/\*

\* Check whether a String is empty (or null). Performs a trim() operation before check

\* Also works with arrays, lists or sets

\*/

${#strings.isEmpty(name)}

${#strings.arrayIsEmpty(nameArr)}

${#strings.listIsEmpty(nameList)}

${#strings.setIsEmpty(nameSet)}

/\*

\* Check whether a String starts or ends with a fragment

\* Also works with arrays, lists or sets

\*/

${#strings.startsWith(name,'Don')} // also array\*, list\* and set\*

${#strings.endsWith(name,endingFragment)} // also array\*, list\* and set\*

/\*

\* Compute length

\* Also works with arrays, lists or sets

\*/

${#strings.length(str)}

/\*

\* Null-safe comparison and concatenation

\*/

${#strings.equals(str)}

${#strings.equalsIgnoreCase(str)}

${#strings.concat(str)}

${#strings.concatReplaceNulls(str)}

/\*

\* Random

\*/

${#strings.randomAlphanumeric(count)}

**使用thymeleaf布局**

使用thymeleaf布局非常的方便

定义代码片段

<footer th:fragment="copy">

&copy; 2016

</footer>

在页面任何地方引入：

<body>

<div th:include="footer :: copy"></div>

<div th:replace="footer :: copy"></div>

</body>

th:include 和 th:replace区别，include只是加载，replace是替换

返回的HTML如下：

<body>

<div> &copy; 2016 </div>

<footer>&copy; 2016 </footer>

</body>

下面是一个常用的后台页面布局，将整个页面分为头部，尾部、菜单栏、隐藏栏，点击菜单只改变content区域的页面

<body class="layout-fixed">

<div th:fragment="navbar" class="wrapper" role="navigation">

<div th:replace="fragments/header :: header">Header</div>

<div th:replace="fragments/left :: left">left</div>

<div th:replace="fragments/sidebar :: sidebar">sidebar</div>

<div layout:fragment="content" id="content" ></div>

<div th:replace="fragments/footer :: footer">footer</div>

</div>

</body>

任何页面想使用这样的布局值只需要替换中见的 content模块即可

<html xmlns:th="http://www.thymeleaf.org" layout:decorator="layout">

<body>

<section layout:fragment="content">

...

也可以在引用模版的时候传参

<head th:include="layout :: htmlhead" th:with="title='Hello'"></head>

layout 是文件地址，如果有文件夹可以这样写 fileName/layout:htmlhead  
htmlhead 是指定义的代码片段 如 th:fragment="copy"

**源码案例**

这里有一个开源项目几乎使用了这里介绍的所有标签和布局，大家可以参考：

[**示例代码-github**](https://github.com/cloudfavorites/favorites-web)

[**示例代码-码云**](https://gitee.com/ityouknow/favorites-web)