代码案例：<https://gitee.com/zx19890628/spring-boot-example/blob/master/lab_006_web_thymeleaf/>

官方：<https://www.thymeleaf.org/>

# 什么是thymeleaf

Thymeleaf是一个Java库。它是一个XML / XHTML / HTML5模板引擎,能够应用于转换模板文件,以显示您的应用程序产生的数据和文本。它尤其适合于基于XHTML / HTML5的web服务应用程序,同时它可以处理任何XML文件,作为web或独立的应用程序。Thymeleaf的主要目的是提供一个优雅和格式良好的方式创建模板。为了实现这一目标,它把预定义的逻辑放在XML的标记和属性上,而不是显式放在XML标记的内容上。依靠智能缓存去解析文件，致使其执行期间的I / O操作达到了最少数量，因此其处理的模板的能力实非常快速的。

spring boot并不推荐使用jsp

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。

## 什么样的模板可以用Thymeleaf制作

开箱即用,Thymeleaf允许您处理六种模板,每个被称为样板。

XML

Valid XML

XHTML

Valid XHTML

HTML5

Legacy HTML5

以上都可以提供格式良好的XML文件，除了Legacy HTML5，Legacy HTML5有些特点的结构不是很好，比如单独的tag（没有关闭的tag）,tag的value没有写，或者没有写在引号之间。为了解决这个问题，Thymeleaf会先把文件转换成格式良好的XML文件。

注意,验证只用于XML和XHTML模板。

然而,这些并不是唯一Thymeleaf可以处理的模板,用户总是能够定义他/她自己的模板，任何可以被建模为一个DOM树(不一定非得是XML)都可以被有效地处理。

## 方言:标准方言

Thymeleaf是一个极其可扩展的模板引擎(事实上它应该更好地称为一个模板引擎的框架),允许您完全定义将在您的模板处理，及将如何被处理的DOM节点,。

被赋予一定逻辑的DOM节点被称为处理者，加上一些额外的功能——被称为方言,Thymeleaf的核心库提供了一个开箱即用的，应该足够大比例的用户的需求的东西称之为标准方言。

本教程涵盖的，在接下来的内容中将学习定义的每个属性和语法功能的方言是标准方言。,即使没有明确提及。

当然,如果用户想利用库的的高级特性定义自己的处理逻辑,用户可以创建自己的方言(甚至扩展标准方言)。模板引擎可以一次同时配置多个方言。

官方thymeleaf-spring3和thymeleaf-spring4都被定义为“Spring Standard Dialect（spring标准方言）”,相当于标准的方言,他们更好地利用Spring框架的一些特性(例如,通过使用Spring Expression language 代替Thymeleaf的标准OGNL)。如果你是一个Spring MVC的用户，你不是在浪费时间,几乎所有你在这里学习的都将会在Spring应用程序中使用。

Thymeleaf标准方言可以在任何模式中制作模板,尤其适用户web模板(XHTML和HTML5的)。除了HTML5,它也特别支持下面的XHTML规格:XHTML 1.0 Transitional, XHTML 1.0 Strict, XHTML 1.0 Frameset , 和 XHTML 1.1.。

大多数标准方言下的处理者是基于XML属性的处理者。这允许浏览器在处理之前正确地显示XHTML / HTML5模板,因为他们会简单地忽略额外的属性。举个栗子,一个JSP使用标签库的片段代码不能直接被浏览器显示：

<form:inputText name="userName" value="${user.name}" />

Thymeleaf标准方言将使我们能够实现相同的功能，但是可以被浏览器显示：

<input type="text" name="userName" value="James Carrot" th:value="${user.name}" />

当模板在静态的浏览器中打开时，模板不仅会正确显示,而且我们(可选)可以指定一个值的属性”James Carrot”,显示。如果模板Thymeleaf处理后，$ { user.name } 会被替换成正确的值显示在value上。

如果需要,这将让你的设计师和开发人员工作在同一模板文件上,减少了需要将静态原型转换成模板文件的工作。这通常被称之为“Natural Templating”。

## 总体结构

Thymeleaf的核心是一个DOM处理引擎。具体地说,它使用自己的高性能——DOM实现，而非的标准DOM API 来构建你的模板。

使用DOM的模板的方式很适合web应用程序，因为web文档通常表示为对象树(事实上DOM树是浏览器把web页面存在内存中方式)。同时, 在大多数web应用程序中只使用一个几十个模板,这些都不是大文件,在应用程序运行时,他们通常都不会改变，Thymeleaf对大多数模板处理操作使用了缓存，这样只需要很少的I / O(如果有的话)。

如果你想要更多的细节,稍后在本教程中有一个完整的章节来介绍缓存。

然而,有一个限制，相比其他模板解析/处理方法，这个架构需要使用更大的大量的内存空间来执行模板,这意味着你不应该使用其创建大数据的XML文档(而不是web文件)。一般的经验(根据JVM的内存大小),如果你生成单个XML文件大小在一个几十兆字节时,我劝你最好不要使用Thymeleaf。

# 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}

## 表达式案例

<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'))

例如：页面上有个 th:value="${user.id} ，而controller没有返回user对象，这是页面会抛出异常

EL1007E: Property or field 'id' cannot be found on null

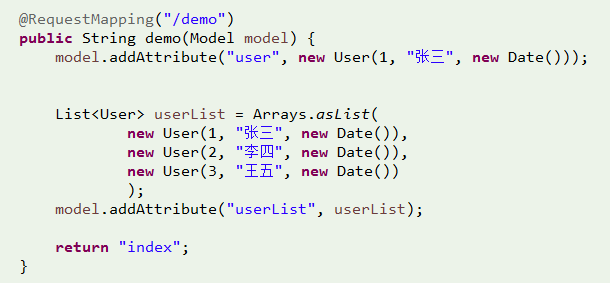
这个时候我们需要加个?即可。 th:value="${user?.id} ?用于判断对象是否存在，存在则调用后面的属性

# 局部变量

Thymeleaf的局部变量定义在模块里，并且只有在此模块生效

for循环：<tr th:each="prod : ${prods}"></tr>

Thymeleaf提供一种定义变量的方式来取代迭代。当th:with被加工后，firstPer的局部变量被创建，并且有效范围是此div内。



<div th:with="uu=${userList[0]}">

<span th:text="${uu.name}"></span>

</div>

你同时可以定义多个局部变量

<div th:with="uu1=${userList[0]},uu2=${userList[1]}">

<span th:text="${uu1.name}"></span>

<span th:text="${uu2.name}"></span>

</div>

# 属性的优先级列表

当你在一个tag里面定义多个属性后。优先级就比较重要了。

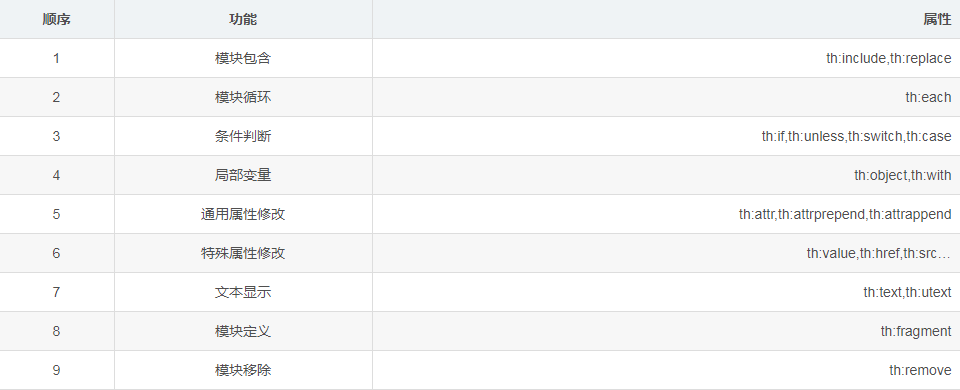
<ul>

<li th:each="item : ${items}" th:text="${item.description}">Item description here...</li>

</ul>

上述代码必须先执行each，再执行text，否则就会出错。为了保证上述优先级，Thymeleaf给自己的属性都定义了一个顺序。

Thymeleaf的优先级定义是通过数字标记的升序来定义，这个顺序是：



# 注释

1. 标准的html/xhtml注释

标准的html/xhtml注释可以在模板文件中任意使用。所有在<!– –>里面的内容都不会被thymeleaf和浏览器解析。

<!-- User info follows -->

<div th:text="${...}">

...

</div>

2.Thymeleaf级别的注释

thymeleaf级别的注释，指的是那些在引擎解析的时候会去掉的注释部分。

<!--/\* 这些代码会被引擎解析时去掉 \*/-->

引擎会去掉所有<!–/\* 和 \*/–>之间的内容。所以你也可以用它来显示静态文本。

<!--/\*-->

<div>

you can see me only before thymeleaf processes me!

</div>

<!--\*/-->

你可以用它来注释其他的TR。

<table>

<tr th:each="x : ${xs}">

</tr>

<!--/\*-->

<tr>

</tr>

<tr>

</tr>

<!--\*/-->

</table>

3. 保留注释块的内容

thymeleaf中还有一种用法，在解析时用来保留注释块里的内容，并去掉注释标签。

<span>hello!</span>

<!--/\*/

<div th:text="${...}">

...

</div>

/\*/-->

<span>goodbye!</span>

Thymeleaf解析系统会删除< !–/ \* /,/ \* / –>标记,保留里面的内容。

<span>hello!</span>

<div th:text="${...}">

...

</div>

<span>goodbye!</span>

4. th:block标签

th:block用来定义一个代码块。并执行里面的属性。这将在循环的时候特别有用。

<table>

<th:block th:each="user : ${users}">

<tr>

<td th:text="${user.login}">...</td>

<td th:text="${user.name}">...</td>

</tr>

<tr>

<td colspan="2" th:text="${user.address}">...</td>

</tr>

</th:block>

</table>

尤其在和保留内容的注释同时使用时：

<table>

<!--/\*/ <th:block th:each="user : ${users}"> /\*/-->

<tr>

<td th:text="${user.login}">...</td>

<td th:text="${user.name}">...</td>

</tr>

<tr>

<td colspan="2" th:text="${user.address}">...</td>

</tr>

<!--/\*/ </th:block> /\*/-->

</table>

# 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。

事件传值：th:onclick=*"'goUpdate(' + ${tfw.id} + ')'"*

# 操作集合

操作map。使用thymelead提供的工具类，maps，可以操作基本属性和方法。

使用get：<span th:test="${map.get(year)}"></span>

遍历：<span th:each="m : ${map}" th:text="${m.key} + '=' + ${m.value}"></span>

# 几种常用的使用方法

### 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}!|">

### 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

比如这里我们需要判断属性

<input type="radio" name="callType" th:each="callType,iterStat:${allCallTypes}"

th:value="${callType.value}" th:title="${callType.name}"

th:checked="${iterStat.count==1} ? 'true' : 'false'"

/>

<select name="osCode">

<option th:each="oo,index : ${os}" th:value="${oo.code}" th:text="${oo.name}"

th:selected="${tfw?.osCode == oo.code} ? 'true' : 'false'"

></option>

</select>

### 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

### 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

### 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';

/\* -]\*/

### 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

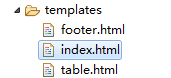
\*/

${#strings.randomAlphanumeric(count)}

# 使用thymeleaf布局

### 快速使用

创建3个页面



1. footer.html 内容

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

<div th:fragment="copy">

© 2011 The Good Thymes Virtual Grocery

</div>

</body>

</html>

2.table.html内容

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

<div th:fragment="table\_number" >

<table style="border: 1px solid red;">

<tr><td>111</td><td>222</td></tr>

<tr><td>111</td><td>222</td></tr>

<tr><td>111</td><td>222</td></tr>

<tr><td>111</td><td>222</td></tr>

</table>

</div>

</body>

</html>

3. index.html 内容

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

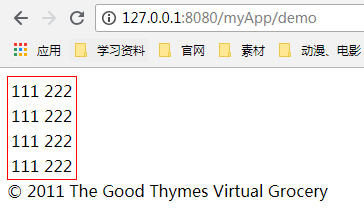
<div th:include="table :: table\_number"></div>

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

</body>

</html>

4. 效果



### 语法

include的表达式想当简洁。这里有三种写法：

这里 templatename 是页面的名称， domselector是使用 th:fragment 修饰的DOM节点

1. “templatename::domselector” 或者 “templatename::[domselector]”引入模板页面中的某个模块。

2. “templatename”引入模板页面。

“::domselector” 或者 “this::domselector” 引入自身模板的模块

3.上面所有的templatename和domselector的写法都支持表达式写法：

<div th:include="footer :: (${user.isAdmin}? #{footer.admin} : #{footer.normaluser})"></div>

另外，thymeleaf还支持使用css选择器，引入模板

比如 footer 页面，有个div的id是 copy-section

<div id="copy-section">© 2011 The Good Thymes Virtual Grocery</div>

index页面想要引用的话，如下：domselector = #copy-section

<body>

<div th:include="footer :: #copy-section"></div>

</body>

### th:include 和 th:replace的区别

th:include和th:replace都可以引入模块，两者的区别在于

th:include：引入模板的children， 保留引用者的节点。

th:replace：引入模板的所有节点， 将引用者的节点覆盖掉。

比如：

<footer th:fragment="copy">© 2011 The Good Thymes Virtual Grocery</footer>

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

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

结果：

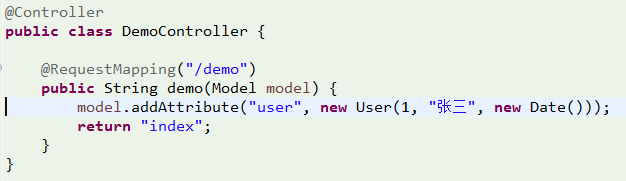
<div>© 2011 The Good Thymes Virtual Grocery</div > 原来的div保留，之引入了模板的子节点。（也就是内容）

<footer >© 2011 The Good Thymes Virtual Grocery</footer > 原来的div被模板覆盖了

### 给引入模块添加参数

我们的模块当中肯定有需要有参数的需求

比如，我们向页面中返回相应数据。user对象，到index页面中，这个对象要在args.html页面中使用。



1. 先定义args.html

<div th:fragment="params(user)">

<p th:text="${user.id} + '-' + ${user.name} + '-' + ${user.brith}">...</p>

</div>

2. 引用模板

<div th:include="args :: params(${user})"></div>

3. 页面效果

clipboard.png

4. 多参数用法

<div th:fragment="handler(id, name)">

<p th:text="${id} + '~~~~~~~~~~~~' + ${name}"></p>

</div>

调用的时候：

<div th:include="args :: handler(${user.id}, ${user.name})"></div>

<div th:include="args :: handler(id = ${user.id}, name = ${user.name})"></div>

<div th:include="args :: handler(name =${user.name} , id = ${user.id})"></div>

第二种用法中参数顺序并不重要

5. 异常数据处理

如果传入的参数是null。需要特殊处理下。在传参的后面加个?。防止异常出现

<div th:fragment="params(user)">

<p th:text="${user?.id} + '-' + ${user?.name} + '-' + ${user?.brith}"></p>

</div>

6. 引入没有被定义参数的模板

比如，模板如下。

<div th:fragment="noParams">

<p th:text="${id} + '~~~~~~~~~~~~' + ${name}"></p>

</div>

由于模板中使用了数据，所以我们可以传入参数

不带参数： <div th:include="args :: noParams"></div>

一个参数： <div th:include="args :: noParams" th:with="id = ${user.id}"></div>

两个参数： <div th:include="args :: noParams" th:with="id = ${user.id}, name = ${user.name}"></div>

### 解析式删除不需要的内容

这才是此技术最吸引人的地方，可以让前端开发人员和后端开发人员使用同一个模板，并且都能看到自己想要的效果

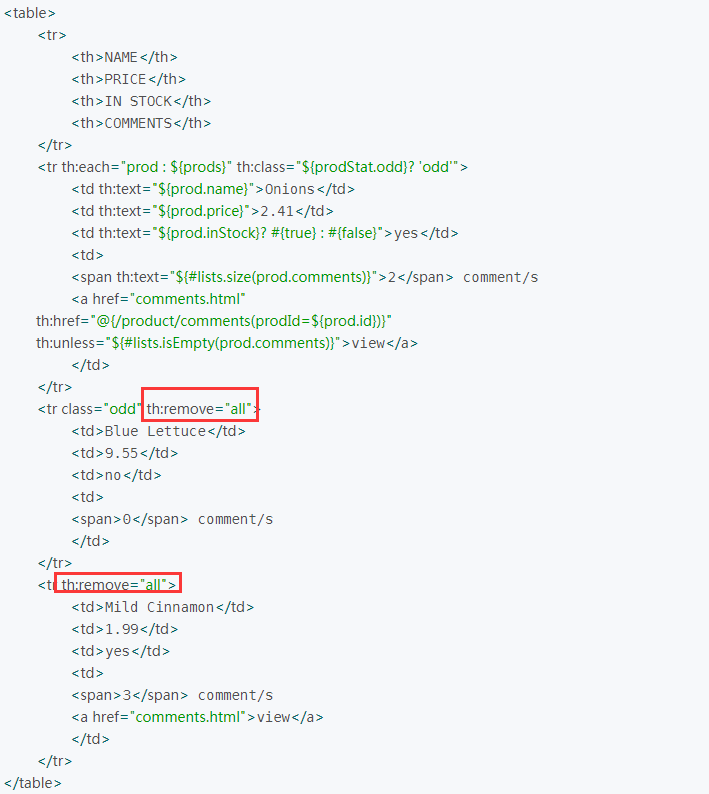
后端开发人员使用的界面



前端开发人员。里面的内容，主要是前端的同学使用



为了解决这个问题th:remove华丽登场了



这个模板在后端开发通过thymeleaf解析后会移除掉有th：remove的标签，满足后端的预期。同时在前端眼中，也是自己预览的效果。

th:remove总共有五种属性：

1. all : 移除tag标记和children。
2. body：保留tag标记和移除children。
3. tag :移除tag和保留children.
4. all-but-first :保留tag和移除除了第一个外的所有children。
5. none :什么都不做。

当然属性也支持表达式：

<a href="/something" th:remove="${condition}? tag : none">Link text not to be removed</a>

<a href="/something" th:remove="${condition}? tag">Link text not to be removed</a>

### 项目案例实践

使用thymeleaf布局非常的方便

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"

# 工具对象用法

### #dates

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Dates

\* ======================================================================

\*/

/\*

\* Format date with the standard locale format

\* Also works with arrays, lists or sets

\*/

${#dates.format(date)}

${#dates.arrayFormat(datesArray)}

${#dates.listFormat(datesList)}

${#dates.setFormat(datesSet)}

/\*

\* Format date with the ISO8601 format

\* Also works with arrays, lists or sets

\*/

${#dates.formatISO(date)}

${#dates.arrayFormatISO(datesArray)}

${#dates.listFormatISO(datesList)}

${#dates.setFormatISO(datesSet)}

/\*

\* 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')}

/\*

\* Obtain date properties

\* Also works with arrays, lists or sets

\*/

${#dates.day(date)} // also arrayDay(...), listDay(...), etc.

${#dates.month(date)} // also arrayMonth(...), listMonth(...), etc.

${#dates.monthName(date)} // also arrayMonthName(...), listMonthName(...), etc.

${#dates.monthNameShort(date)} // also arrayMonthNameShort(...), listMonthNameShort(...), etc.

${#dates.year(date)} // also arrayYear(...), listYear(...), etc.

${#dates.dayOfWeek(date)} // also arrayDayOfWeek(...), listDayOfWeek(...), etc.

${#dates.dayOfWeekName(date)} // also arrayDayOfWeekName(...), listDayOfWeekName(...), etc.

${#dates.dayOfWeekNameShort(date)} // also arrayDayOfWeekNameShort(...), listDayOfWeekNameShort(...), etc.

${#dates.hour(date)} // also arrayHour(...), listHour(...), etc.

${#dates.minute(date)} // also arrayMinute(...), listMinute(...), etc.

${#dates.second(date)} // also arraySecond(...), listSecond(...), etc.

${#dates.millisecond(date)} // also arrayMillisecond(...), listMillisecond(...), etc.

/\*

\* Create date (java.util.Date) objects from its components

\*/

${#dates.create(year,month,day)}

${#dates.create(year,month,day,hour,minute)}

${#dates.create(year,month,day,hour,minute,second)}

${#dates.create(year,month,day,hour,minute,second)}

${#dates.create(year,month,day,hour,minute,second,millisecond)}

/\*

\* 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()}

### #Calendars

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Calendars

\* ======================================================================

\*/

/\*

\* Format calendar with the standard locale format

\* Also works with arrays, lists or sets

\*/

${#calendars.format(cal)}

${#calendars.arrayFormat(calArray)}

${#calendars.listFormat(calList)}

${#calendars.setFormat(calSet)}

/\*

\* Format calendar with the ISO8601 format

\* Also works with arrays, lists or sets

\*/

${#calendars.formatISO(cal)}

${#calendars.arrayFormatISO(calArray)}

${#calendars.listFormatISO(calList)}

${#calendars.setFormatISO(calSet)}

/\*

\* Format calendar with the specified pattern

\* Also works with arrays, lists or sets

\*/

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

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

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

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

/\*

\* Obtain calendar properties

\* Also works with arrays, lists or sets

\*/

${#calendars.day(date)} // also arrayDay(...), listDay(...), etc.

${#calendars.month(date)} // also arrayMonth(...), listMonth(...), etc.

${#calendars.monthName(date)} // also arrayMonthName(...), listMonthName(...), etc.

${#calendars.monthNameShort(date)} // also arrayMonthNameShort(...), listMonthNameShort(...), etc.

${#calendars.year(date)} // also arrayYear(...), listYear(...), etc.

${#calendars.dayOfWeek(date)} // also arrayDayOfWeek(...), listDayOfWeek(...), etc.

${#calendars.dayOfWeekName(date)} // also arrayDayOfWeekName(...), listDayOfWeekName(...), etc.

${#calendars.dayOfWeekNameShort(date)} // also arrayDayOfWeekNameShort(...), listDayOfWeekNameShort(...), etc.

${#calendars.hour(date)} // also arrayHour(...), listHour(...), etc.

${#calendars.hour(date)} // also arrayHour(...), listHour(...), etc.

${#calendars.minute(date)} // also arrayMinute(...), listMinute(...), etc.

${#calendars.second(date)} // also arraySecond(...), listSecond(...), etc.

${#calendars.millisecond(date)} // also arrayMillisecond(...), listMillisecond(...), etc.

/\*

\* Create calendar (java.util.Calendar) objects from its components

\*/

${#calendars.create(year,month,day)}

${#calendars.create(year,month,day,hour,minute)}

${#calendars.create(year,month,day,hour,minute,second)}

${#calendars.create(year,month,day,hour,minute,second,millisecond)}

/\*

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

\*/

${#calendars.createNow()}

/\*

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

\*/

${#calendars.createToday()}

### #numbers

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Numbers

\* ======================================================================

\*/

/\*

\* ==========================

\* Formatting integer numbers

\* ==========================

\*/

/\*

\* Set minimum integer digits.

\* Also works with arrays, lists or sets

\*/

${#numbers.formatInteger(num,3)}

${#numbers.arrayFormatInteger(numArray,3)}

${#numbers.listFormatInteger(numList,3)}

${#numbers.setFormatInteger(numSet,3)}

/\*

\* Set minimum integer digits and thousands separator:

\* 'POINT', 'COMMA', 'WHITESPACE', 'NONE' or 'DEFAULT' (by locale).

\* Also works with arrays, lists or sets

\*/

${#numbers.formatInteger(num,3,'POINT')}

${#numbers.arrayFormatInteger(numArray,3,'POINT')}

${#numbers.listFormatInteger(numList,3,'POINT')}

${#numbers.setFormatInteger(numSet,3,'POINT')}

/\*

\* ==========================

\* Formatting decimal numbers

\* ==========================

\*/

/\*

\* Set minimum integer digits and (exact) decimal digits.

\* Also works with arrays, lists or sets

\*/

${#numbers.formatDecimal(num,3,2)}

${#numbers.arrayFormatDecimal(numArray,3,2)}

${#numbers.listFormatDecimal(numList,3,2)}

${#numbers.setFormatDecimal(numSet,3,2)}

/\*

\* Set minimum integer digits and (exact) decimal digits, and also decimal separator.

\* Also works with arrays, lists or sets

\*/

${#numbers.formatDecimal(num,3,2,'COMMA')}

${#numbers.arrayFormatDecimal(numArray,3,2,'COMMA')}

${#numbers.listFormatDecimal(numList,3,2,'COMMA')}

${#numbers.setFormatDecimal(numSet,3,2,'COMMA')}

/\*

\* Set minimum integer digits and (exact) decimal digits, and also thousands and

\* decimal separator.

\* Also works with arrays, lists or sets

\*/

${#numbers.formatDecimal(num,3,'POINT',2,'COMMA')}

${#numbers.arrayFormatDecimal(numArray,3,'POINT',2,'COMMA')}

${#numbers.listFormatDecimal(numList,3,'POINT',2,'COMMA')}

${#numbers.setFormatDecimal(numSet,3,'POINT',2,'COMMA')}

/\*

\* ==========================

\* Utility methods

\* ==========================

\*/

/\*

\* Create a sequence (array) of integer numbers going

\* from x to y

\*/

${#numbers.sequence(from,to)}

${#numbers.sequence(from,to,step)}

### #strings

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Strings

\* ======================================================================

\*/

/\*

\* Null-safe toString()

\*/

${#strings.toString(obj)} // also array\*, list\* and set\*

/\*

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

\* Also works with arrays, lists or sets

\* \* Also works with arrays, lists or sets

\*/

${#strings.isEmpty(name)}

${#strings.arrayIsEmpty(nameArr)}

${#strings.listIsEmpty(nameList)}

${#strings.setIsEmpty(nameSet)}

/\*

\* Perform an 'isEmpty()' check on a string and return it if false, defaulting to

\* another specified string if true.

\* Also works with arrays, lists or sets

\*/

${#strings.defaultString(text,default)}

${#strings.arrayDefaultString(textArr,default)}

${#strings.listDefaultString(textList,default)}

${#strings.setDefaultString(textSet,default)}

/\*

\* Check whether a fragment is contained in a String

\* Also works with arrays, lists or sets

\*/

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

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

/\*

\* 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\*

/\*

\* Substring-related operations

\* Also works with arrays, lists or sets

\*/

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

${#strings.substring(name,3,5)} // also array\*, list\* and set\*

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

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

${#strings.replace(name,'las','ler')} // also array\*, list\* and set\*

/\*

\* Append and prepend

\* Also works with arrays, lists or sets

\*/

${#strings.prepend(str,prefix)} // also array\*, list\* and set\*

${#strings.append(str,suffix)} // also array\*, list\* and set\*

/\*

\* Change case

\* Also works with arrays, lists or sets

\*/

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

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

/\*

\* Split and join

\*/

${#strings.arrayJoin(namesArray,',')}

${#strings.listJoin(namesList,',')}

${#strings.setJoin(namesSet,',')}

${#strings.arraySplit(namesStr,',')} // returns String[]

${#strings.listSplit(namesStr,',')} // returns List<String>

${#strings.setSplit(namesStr,',')} // returns Set<String>

/\*

\* Trim

\* Also works with arrays, lists or sets

\*/

${#strings.trim(str)} // also array\*, list\* and set\*

/\*

\* Compute length

\* Also works with arrays, lists or sets

\*/

${#strings.length(str)} // also array\*, list\* and set\*

/\*

\* Abbreviate text making it have a maximum size of n. If text is bigger, it

\* will be clipped and finished in "..."

\* Also works with arrays, lists or sets

\*/

${#strings.abbreviate(str,10)} // also array\*, list\* and set\*

/\*

\* Convert the first character to upper-case (and vice-versa)

\*/

${#strings.capitalize(str)} // also array\*, list\* and set\*

${#strings.unCapitalize(str)} // also array\*, list\* and set\*

/\*

\* Convert the first character of every word to upper-case

\*/

${#strings.capitalizeWords(str)} // also array\*, list\* and set\*

${#strings.capitalizeWords(str,delimiters)} // also array\*, list\* and set\*

/\*

\* Escape the string

\*/

${#strings.escapeXml(str)} // also array\*, list\* and set\*

${#strings.escapeJava(str)} // also array\*, list\* and set\*

${#strings.escapeJavaScript(str)} // also array\*, list\* and set\*

${#strings.unescapeJava(str)} // also array\*, list\* and set\*

${#strings.unescapeJavaScript(str)} // also array\*, list\* and set\*

/\*

\* Null-safe comparison and concatenation

\*/

${#strings.equals(first, second)}

${#strings.equalsIgnoreCase(first, second)}

${#strings.concat(values...)}

${#strings.concatReplaceNulls(nullValue, values...)}

/\*

\* Random

\*/

${#strings.randomAlphanumeric(count)}

### #objects

${#objects.nullSafe(obj,default)}

${#objects.arrayNullSafe(objArray,default)}

${#objects.listNullSafe(objList,default)}

${#objects.setNullSafe(objSet,default)}

### #bools

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Bools

\* ======================================================================

\*/

/\*

\* Evaluate a condition in the same way that it would be evaluated in a th:if tag

\* (see conditional evaluation chapter afterwards).

\* Also works with arrays, lists or sets

\*/

${#bools.isTrue(obj)}

${#bools.arrayIsTrue(objArray)}

${#bools.listIsTrue(objList)}

${#bools.setIsTrue(objSet)}

/\*

\* Evaluate with negation

\* Also works with arrays, lists or sets

\*/

${#bools.isFalse(cond)}

${#bools.arrayIsFalse(condArray)}

${#bools.listIsFalse(condList)}

${#bools.setIsFalse(condSet)}

/\*

\* Evaluate and apply AND operator

\* Receive an array, a list or a set as parameter

\*/

${#bools.arrayAnd(condArray)}

${#bools.listAnd(condList)}

${#bools.setAnd(condSet)}

/\*

\* Evaluate and apply OR operator

\* Receive an array, a list or a set as parameter

\*/

${#bools.arrayOr(condArray)}

${#bools.listOr(condList)}

${#bools.setOr(condSet)}

### #arrays

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Arrays

\* ======================================================================

\*/

/\*

\* Converts to array, trying to infer array component class.

\* Note that if resulting array is empty, or if the elements

\* of the target object are not all of the same class,

\* this method will return Object[].

\*/

${#arrays.toArray(object)}

/\*

\* Convert to arrays of the specified component class.

\*/

${#arrays.toStringArray(object)}

${#arrays.toIntegerArray(object)}

${#arrays.toLongArray(object)}

${#arrays.toDoubleArray(object)}

${#arrays.toFloatArray(object)}

${#arrays.toBooleanArray(object)}

/\*

\* Compute length

\*/

${#arrays.length(array)}

/\*

\* Check whether array is empty

\*/

${#arrays.isEmpty(array)}

/\*

\* Check if element or elements are contained in array

\*/

${#arrays.contains(array, element)}

${#arrays.containsAll(array, elements)}

### #lists

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Lists

\* ======================================================================

\*/

/\*

\* Converts to list

\*/

${#lists.toList(object)}

/\*

\* Compute size

\*/

${#lists.size(list)}

/\*

\* Check whether list is empty

\*/

${#lists.isEmpty(list)}

/\*

\* Check if element or elements are contained in list

\*/

${#lists.contains(list, element)}

${#lists.containsAll(list, elements)}

/\*

\* Sort a copy of the given list. The members of the list must implement

\* comparable or you must define a comparator.

\*/

${#lists.sort(list)}

${#lists.sort(list, comparator)}

### #sets

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Sets

\* ======================================================================

\*/

/\*

\* Converts to set

\*/

${#sets.toSet(object)}

/\*

\* Compute size

\*/

${#sets.size(set)}

/\*

\* Check whether set is empty

\*/

${#sets.isEmpty(set)}

/\*

\* Check if element or elements are contained in set

\*/

${#sets.contains(set, element)}

${#sets.containsAll(set, elements)}

### #maps

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Maps

\* ======================================================================

\*/

/\*

\* Compute size

\*/

${#maps.size(map)}

/\*

\* Check whether map is empty

\*/

${#maps.isEmpty(map)}

/\*

\* Check if key/s or value/s are contained in maps

\*/

${#maps.containsKey(map, key)}

${#maps.containsAllKeys(map, keys)}

${#maps.containsValue(map, value)}

${#maps.containsAllValues(map, value)}

### #aggregates

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Aggregates

\* ======================================================================

\*/

/\*

\* Compute sum. Returns null if array or collection is empty

\*/

${#aggregates.sum(array)}

${#aggregates.sum(collection)}

/\*

\* Compute average. Returns null if array or collection is empty

\*/

${#aggregates.avg(array)}

${#aggregates.avg(collection)}

### #messages

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Messages

\* ======================================================================

\*/

/\*

\* Obtain externalized messages. Can receive a single key, a key plus arguments,

\* or an array/list/set of keys (in which case it will return an array/list/set of

\* externalized messages).

\* If a message is not found, a default message (like '??msgKey??') is returned.

\*/

${#messages.msg('msgKey')}

${#messages.msg('msgKey', param1)}

${#messages.msg('msgKey', param1, param2)}

${#messages.msg('msgKey', param1, param2, param3)}

${#messages.msgWithParams('msgKey', new Object[] {param1, param2, param3, param4})}

${#messages.arrayMsg(messageKeyArray)}

${#messages.listMsg(messageKeyList)}

${#messages.setMsg(messageKeySet)}

/\*

\* Obtain externalized messages or null. Null is returned instead of a default

\* message if a message for the specified key is not found.

\*/

${#messages.msgOrNull('msgKey')}

${#messages.msgOrNull('msgKey', param1)}

${#messages.msgOrNull('msgKey', param1, param2)}

${#messages.msgOrNull('msgKey', param1, param2, param3)}

${#messages.msgOrNullWithParams('msgKey', new Object[] {param1, param2, param3, param4})}

${#messages.arrayMsgOrNull(messageKeyArray)}

${#messages.listMsgOrNull(messageKeyList)}

${#messages.setMsgOrNull(messageKeySet)}

### #ids

/\*

\* ======================================================================

\* See javadoc API for class org.thymeleaf.expression.Ids

\* ======================================================================

\*/

/\*

\* Normally used in th:id attributes, for appending a counter to the id attribute value

\* so that it remains unique even when involved in an iteration process.

\*/

${#ids.seq('someId')}

/\*

\* Normally used in th:for attributes in <label> tags, so that these labels can refer to Ids

\* generated by means if the #ids.seq(...) function.

\*

\* Depending on whether the <label> goes before or after the element with the #ids.seq(...)

\* function, the "next" (label goes before "seq") or the "prev" function (label goes after

\* "seq") function should be called.

\*/

${#ids.next('someId')}

${#ids.prev('someId')}

### 基础对象和web参数访问方式

<p>param :获取请求的参数.</p>

<a href="http://127.0.0.1:8080/myApp/demo2?userName=张三&userId=10">访问连接：http://127.0.0.1:8080/myApp/demo2?userName=张三&userId=10</a>

<p th:text="${param.userName}"></p>

<p th:text="${param.containsKey('userName')}"></p>

<p th:text="${param.size()}"></p>

<p th:text="${param.isEmpty()}"></p>

<hr/>

<p>session :获取请求的参数.</p>

<p th:text="${session.userName}"></p>

<p th:text="${session.containsKey('userName')}"></p>

<p th:text="${session.size()}"></p>

<p th:text="${session.isEmpty()}"></p>

<hr/>

<p>application :获取请求的参数.</p>

<p th:text="${application.userName}"></p>

<p th:text="${application.containsKey('userName')}"></p>

<p th:text="${application.size()}"></p>

<p th:text="${application.isEmpty()}"></p>

<hr/>

<p>web环境对象</p>

#httpServletRequest :javax.servlet.http.HttpServletRequest对象实例

<p th:text="${#httpServletRequest.getAttribute('user')}"></p> <!-- 获得响应值 -->

<p th:text="${#httpServletRequest.getParameter('userName')}"></p> <!-- 获得请求值 -->

<p th:text="${#httpServletRequest.getContextPath()}"></p> <!-- 应用的根路径 -->

<hr/>

#httpSession：javax.servlet.http.HttpSession实例。

<p th:text="${#httpSession.getAttribute('user')}"></p>

<p th:text="${#httpSession.id}"></p>

<p th:text="${#httpSession.lastAccessedTime}"></p>

<hr/>

spring环境对象

<p>#themes : 提供和“ spring:theme JSP tag.”同样的功能。</p>

${#themes.code('foo')}

<hr/>

直接访问spring注册对象

<div th:text="${@userService.getName(1)}"></div>

# 默认配置

spring.thymeleaf.cache=false

spring.thymeleaf.mode=LEGACYHTML5

spring.thymeleaf.encoding=utf-8

spring.thymeleaf.servlet.content-type=text/html

spring.thymeleaf.prefix=classpath:/templates/

spring.thymeleaf.suffix=.html