# Thymeleaf 的基本用法

属于个人整理的文档，大部分内容来源自网络

在这里我们没有打算使用SpringMVC进行整合使用或者说跟Spring Boot 一起使用

我们在这里单独使用Servelet版本-算是为了给一些初学者提供部分代码

Thymeleaf是一款用于渲染XML/XHTML/HTML5内容的模板引擎，类似JSP，Velocity，FreeMaker等，它也可以轻易的与Spring MVC等Web框架进行集成作为Web应用的模板引擎。

**Thymeleaf最大的特点是能够直接在浏览器中打开并正确显示模板页面，而不需要启动整个Web应用，但是总是看到说其效率有点低**

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。

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、改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。

## 1.引入提示

在html页面中引入thymeleaf命名空间，即，此时在html模板文件中动态的属性使用th:命名空间修饰 。

<html lang="en" xmlns:th="http://www.thymeleaf.org">

这样才可以在其他标签里面使用th:这样的语法.这是下面语法的前提\*.

## 2.变量表达式(获取变量值)

<div th:text="'你是否读过，'+${session.book}+'!!'">

同EL表达式有些相似的效果，如果有数据，被替换

完成前后端分离效果(美工代码)</div>

代码分析：

1.可以看出获取变量值用$符号,对于javaBean的话使用变量名.属性名方式获取,这点和EL表达式一样

2.它通过标签中的th:text属性来填充该标签的一段内容，意思是$表达式只能写在th标签内部,不然不会生效,上面例子就是使用th:text标签的值替换div标签里面的值,至于div里面的原有的值只是为了给前端开发时做展示用的.这样的话很好的做到了前后端分离.意味着div标签中的内容会被表达式${session.book}的值所替代，无论模板中它的内容是什么，之所以在模板中“多此一举“地填充它的内容，完全是为了它能够作为原型在浏览器中直接显示出来。

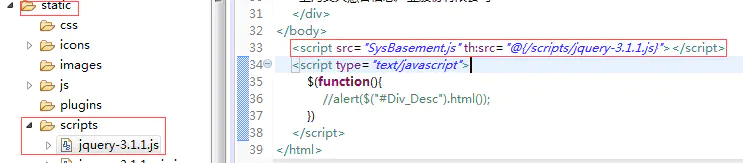
3.访问spring-mvc中model的属性，语法格式为“${}”，如${user.id}可以获取model里的user对象的id属性

4.牛叉的循环<li th:each="book : ${books}" >

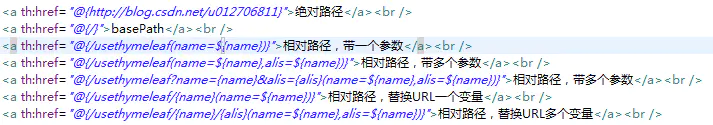
## 3.URL表达式(引入URL)

重点！重点！重点！

引用静态资源文件(CSS使用th:href，js使用使用th:src)



href链接URL(使用th:href)



代码分析

1.最终解析的href为：

/seconddemo/

/seconddemo/usethymeleaf?name=Dear 相对路径，带一个参数

/seconddemo/usethymeleaf?name=Dear&alis=Dear 相对路径，带多个参数

/seconddemo/usethymeleaf?name=Dear&alis=Dear 相对路径，带多个参数

/seconddemo/usethymeleaf/Dear 相对路径，替换URL一个变量

/seconddemo/usethymeleaf/Dear/Dear 相对路径，替换URL多个变量

2.URL最后的(name=${name})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

3.@{/usethymeleaf}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seconddemo，那么结果应该是/seconddemo/usethymeleaf，即URL中以”/“开头的路径(比如/usethymeleaf将会加上服务器地址和域名和应用cotextpath，形成完整的URL。

4.th:href属性修饰符:它将计算并替换使用href链接URL 值，并放入的href属性中。

5.th:href中可以直接使用静态地址

## 4.选择或星号表达式

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\*{customer.name}

<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>

//等价于<div>

<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>

1.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。上面的\*{title}表达式可以理解为${book.title}。(父对象)

2.当然，美元符号和星号语法可以混合使用

**小插曲：三和四的变量表达式、URL表达式所对应的属性都可以使用统一的方式：th.attr=“属性名=属性值”的方式来设置，参考第“七.设置属性值”部分**

## 5.文字国际化表达式

j简单看一下就可以，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:text="#{header.address.country}"></table>

## 6. 表达式支持的语法

字面量（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

文本操作（Text operations）

字符串连接（String concatenation）: +

文本替换（Literal substitutions）: |The name is ${name}|

<div th:class="'content'">...</div>

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

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

<span th:text="${onevar} + ' ' + |${twovar}, ${threevar}|">

算术运算（Arithmetic operations）

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

布尔操作（Boolean operations）

* + Binary operators: and , or
  + Boolean negation (unary operator): ! , not

比较和等价(Comparisons and equality)

* + Comparators: > , < , >= , <= ( gt , lt , ge , le )
  + Equality operators: == , != ( eq , ne )

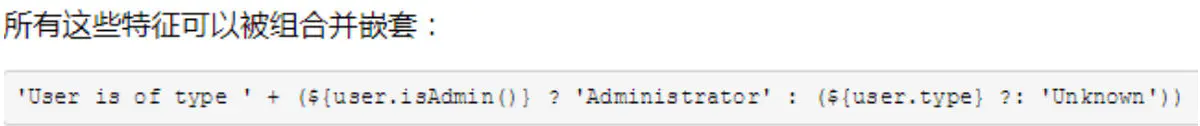
条件运算符（Conditional operators）三元运算符

* + If-then: (if) ? (then)
  + If-then-else: (if) ? (then) : (else)
  + Default: (value) ?: (defaultvalue)
* 示例一：
* <h2 th:text="${expression} ? 'Hello' : 'Something else'"></h2>
* 示例二：
* <!-- IF CUSTOMER IS ANONYMOUS -->
* <div th:if="${customer.anonymous}">
* <div>Welcome, Gues!</div>
* </div>
* <!-- ELSE -->
* <div th:unless="${customer.anonymous}">
* <div th:text=" 'Hi,' + ${customer.name}">Hi, User</div>
* </div>

Special tokens:

* + No-Operation: \_

Switch

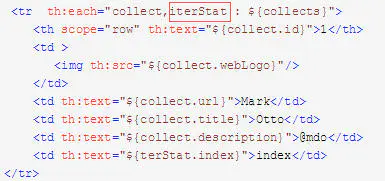
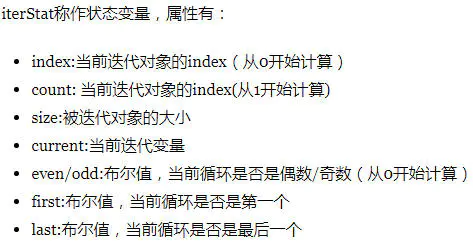


循环

渲染列表数据是一种非常常见的场景，例如现在有n条记录需要渲染成一个表格或li列表标签该数据集合必须是可以遍历的，使用th:each标签

代码分析：

循环，在html的标签中，加入th:each=“value:${list}”形式的属性，如可以迭代prods的数据 又如带状态变量的循环：



​ 利用状态变量判断：

7.设置属性值

1. th:attr

任何属性值，语法格式：th:attr="属性名=属性值,[属性名=属性值]"

属性值如果是使用表达式的话：通常有URL表达式@{}和变量表达式${}

但此标签语法不太优雅

示例：

th:attr="action=@{/subscribe}" //当然也可以直接使用th:action

th:attr="src=@{/images/gtvglogo.png},title=#{logo},alt=#{logo}" //可直接使用th:src

th:attr="value=#{subscribe.submit}"//可直接使用th:value

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

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

2.前置和后置添加属性值

th:attrappend 和 th:attrprepend

主要对class和style两个属性

class="btn" th:attrappend="class=${' ' + cssStyle}"

转换后：class="btn warning"

3.还有两个特定的添加属性

th:classappend 和 th:styleappend

与上面的attrappend功能一样

class="row" th:classappend="${prodStat.odd}? 'odd'"

转换后：奇数行class="row odd"，偶数行class="row"

8.内嵌变量Utilities

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

代码示例：

${#dates.format(dateVar, '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')}

${#dates.createNow()}

${#dates.createToday()}

${#strings.isEmpty(name)}

${#strings.arrayIsEmpty(nameArr)}

${#strings.listIsEmpty(nameList)}

${#strings.setIsEmpty(nameSet)}

${#strings.startsWith(name,'Don')}

// also array\*, list\* and set\*

${#strings.endsWith(name,endingFragment)}

// also array\*, list\* and set\*

${#strings.length(str)}

${#strings.equals(str)}

${#strings.equalsIgnoreCase(str)}

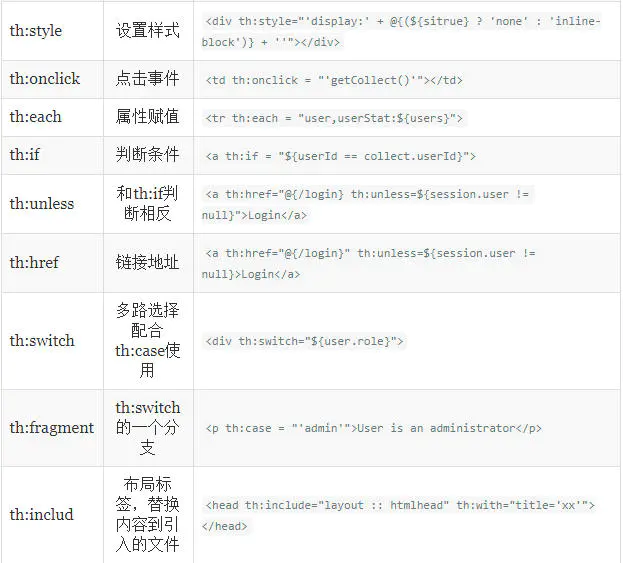
${#strings.concat(str)}

${#strings.concatReplaceNulls(str)}

${#strings.randomAlphanumeric(count)}//产生随机字符串

9.thymeleaf布局

10.附录



作者：胖先森

链接：https://www.jianshu.com/p/d1370aeb0881

来源：简书

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。