thymeleaf 中文文档

（一）Thymeleaf 是个什么？

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。

（二）应用

1.简单的 Thymeleaf 应用

1 ） 只 需 加 入 thymeleaf-2.1.4.RELEASE.jar

（://.thymeleaf.org/download.html ）包，若用 maven，则加入如下配置

```
<dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf</artifactId>
    <version>2.1.4</version>
</dependency>
```

2）然后增加头文件（如下）

```
<!DOCTYPE html>
<html xmlns="://.w3.org/1999/xhtml"
      xmlns:th="://.thymeleaf.org">
```

3）就可以用 th 标签动态替换掉静态数据了。如下图，后台传出的 message 会将静态数据“Red Chair”替换掉，若访问静态页面，则显示数据“Red Chair”。

```
<td th:text="${message}">Red Chair</td>
```

2.整合 spring

1）加入 thymeleaf-spring4-2.1.4.RELEASE.jar（://.thymeleaf.org/download.html ）包，若用 maven，则加入如下配置

```
<dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf-spring3</artifactId>
```

```
    <version>2.1.4</version>
</dependency>
```

2）在 servlet 配置文件中加入如下代码

```
<!-- Scans the classpath of this application for @Components to deploy as beans -->
        <context:component-scan base-package="com.test.thymeleaf.controller" />

        <!-- Configures the @Controller programming model -->
        <mvc:annotation-driven />

         <!--Resolves view names to protected .jsp resources within the /WEB-INF/views directory -->
         <!--springMVC+jsp 的跳转页面配置-->
        <!--<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">-->
                <!--<property name="prefix" value="/WEB-INF/views/" />-->
                <!--<property name="suffix" value=".jsp" />-->
```

```
        <!--</bean>-->

        <!--springMVC+thymeleaf 的跳转页面配置-->
        <bean id="templateResolver"

class="org.thymeleaf.templateresolver.ServletContextTemplat
eResolver">
            <property  name="prefix"  value="/WEB-INF/views/"
/>
            <property name="suffix" value=".html" />
            <property   name="templateMode"   value="HTML5"
/>
        </bean>

        <bean id="templateEngine"

class="org.thymeleaf.spring4.SpringTemplateEngine">
             <property                    name="templateResolver"
ref="templateResolver" />
        </bean>

        <bean
```

```
class="org.thymeleaf.spring4.view.ThymeleafViewResolver">
        <property name="templateEngine"
ref="templateEngine" />
    </bean>
```

3）将静态页面加到项目中，更改文件头，加入 th 标签即可。

3.th 标签整理

1）简单表达式

--变量表达式 ${……}

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

上述代码为引用 user 对象的 name 属性值。

--选择/星号表达式 *{……}

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

选择表达式一般跟在 th:object 后，直接取 object 中的属性。

--文字国际化表达式 #{……}

```
<p th:utext="#{home.welcome}">Welcome to our grocery
store!</p>
```

调用国际化的 welcome 语句,国际化资源文件如下

resource_en_US.properties：

home.welcome=Welcome to here！

resource_zh_CN.properties：

home.welcome=欢迎您的到来！

-- URL 表达式 @{……}

view

@{……}支持决定路径和相对路径。其中相对路径又支持跨上下文调用 url 和协议的引用（//code.jquery.com/jquery-2.0.3.min.js）。

当 URL 为后台传出的参数时，代码如下

2）常用的 th 标签

--简单数据转换（数字，日期）

<dt>价格</dt>

<dd th:text="${#numbers.formatDecimal(product.price, 1, 2)}">180</dd>

<dt>进货日期</dt>

<dd th:text="${#dates.format(product.availableFrom, 'yyyy-MM-dd')}">2014-12-01</dd>

--字符串拼接

<dd th:text="${'$'+product.price}">235</dd>

--转义和非转义文本

当后台传出的数据为“This is an HTML text. Enjoy yourself!”时，若页面代码如下则出现两种不同的结果

This is an HTML text. Enjoy yourself!

This is an HTML text. Enjoy yourself!

--表单中

<form th:action="@{/bb}" th:object="${user}" method="post"

th:method="post">

<input type="text" th:field="*{name}"/>

<input type="text" th:field="*{msg}"/>

<input type="submit"/>

</form>

--显示页面的数据迭代

//用 th:remove 移除除了第一个外的静态数据，用第一个 tr 标签进行循环迭代显示

<tbody th:remove="all-but-first">

//将后台传出的 productList 的集合进行迭代，用 product 参数接收，通过 product 访问属性值

<tr th:each="product:${productList}">

//用 count 进行统计，有顺序的显示

<td th:text="${productStat.count}">1</td>

<td th:text="${product.description}">Red Chair</td>

<td th:text="${'$' +

```
#numbers.formatDecimal(product.price, 1, 2)}">$123</td>
                        <td
th:text="${#dates.format(product.availableFrom,
'yyyy-MM-dd')}">2014-12-01</td>
                    </tr>
                    <tr>
                        <td>White table</td>
                        <td>$200</td>
                        <td>15-Jul-2013</td>
                    </tr>
                    <tr>
                        <td>Reb table</td>
                        <td>$200</td>
                        <td>15-Jul-2013</td>
                    </tr>
                    <tr>
                        <td>Blue table</td>
                        <td>$200</td>
                        <td>15-Jul-2013</td>
                    </tr>
            </tbody>
```

--条件判断

Special offer!

不能用"<”，">"等符号，要用"lt"等替代

```
<!-- 当 gender 存在时，选择对应的选项；若 gender 不存在或为 null 时，取得 customer 对象的 name-->
<td th:switch="${customer.gender?.name()}">
    <img th:case="'MALE'" src="../../../images/male.png" th:src="@{/images/male.png}" alt="Male" /> <!-- Use "/images/male.png" image -->
    <img th:case="'FEMALE'" src="../../../images/female.png" th:src="@{/images/female.png}" alt="Female" /> <!-- Use "/images/female.png" image -->
    <span th:case="*">Unknown</span>
</td>
```

```
<!--在页面先显示，然后再在显示的数据基础上进行修改-->
<div class="form-group col-lg-6">
    <label>姓名<span> </span></label>
     < !--除非 resume 对象的 name 属性值为 null ,否则就用 name
```

```
的值作为 placeholder 值-->
      <input          type="text"          class="form-control"
th:unless="${resumes.name} eq '' or ${resumes.name} eq null"
              data-required="true"
th:placeholder="${resumes.name}" />
       <!--除非 resume 对象的 name 属性不为空，否则就定义一个
field 方便封装对象，并用 placeholder 提示-->
      <input        type="text"        th:field="${resume.name}"
class="form-control" th:unless="${resumes.name} ne null"
              data-required="true"  th:placeholder="请填写您的
真实姓名"  />
</div>
```

```
<!-- 增加 class="enhanced"当 balance 大雨 10000 -->
<td  th:class="${customer.balance  gt  10000}  ?  'enhanced'"
th:text="${customer.balance}">350</td>
      --根据后台数据选中 select 的选项
```

```
 <div class="form-group col-lg-6">
        <label >性别<span> Sex:</span></label>
        <select                        th:field="${resume.gender}"
```

```
class="form-control" th:switch="${resumes.gender.toString()}"
                data-required="true">
                  <option   value="  男  "   th:case="'  男  '"
th:selected="selected" >男</option>
                  <option   value="  女  "   th:case="'  女  '"
th:selected="selected" >女</option>
                  <option value="">请选择</option>
        </select>
  </div>
```

因为 gender 是定义的 Enum（枚举）类型，所以要用 toString 方法。用 th:switch 指定传出的变量,用 th:case 对变量的值进行匹配。!"请选择"放在第一项会出现永远选择的是这个选项。或者用 th:if

```
<div class='form-group col-lg-4'>
            <select                          class='form-control'
name="skill[4].proficiency">
                  <option >掌握程度</option>
                  <option   th:if="${skill.level   eq   '  一  般  '}"
th:selected="selected">一般</option>
                   <option   th:if="${skill.level   eq   '  熟  练  '}"
```

```
th:selected="selected">熟练</option>
                    <option  th:if="${skill.level  eq  '精 通 '}"
th:selected="selected">精通</option>
            </select>
</div>
```

--spring 表达式语言

```
<!DOCTYPE html>
<html xmlns:th="://.thymeleaf.org">
    <head>
        <title>Thymeleaf tutorial: exercise 10</title>
        <link rel="stylesheet" href="../../../css/main-static.css"
th:href="@{/css/main.css}" />
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Thymeleaf  tutorial  -  Solution  for  exercise  10:
Spring Expression language</h1>

        <h2>Arithmetic expressions</h2>
        <p  class="label">Four  multiplied  by  minus  six
```

```
multiplied by minus two module seven:</p>
        <p class="answer" th:text="${4 * -6 * -2 % 7}">123</p>

        <h2>Object navigation</h2>
        <p class="label">Description field of paymentMethod field of the third element of customerList bean:</p>
        <p class="answer" th:text="${customerList[2].paymentMethod.description}">Credit card</p>

        <h2>Object instantiation</h2>
        <p class="label">Current time milliseconds:</p>
        <p class="answer" th:text="${new java.util.Date().getTime()}">22-Jun-2013</p>

        <h2>T operator</h2>
        <p class="label">Random number:</p>
        <p class="answer" th:text="${T(java.lang.Math).random()}">123456</p>
    </body>
```

</html>

--内联

<label for="body">Message body:</label>

<textarea id="body" name="body" th:inline="text">

Dear [[${customerName}]],

it is our sincere pleasure to congratulate your in your birthday:

Happy birthday [[${customerName}]]!!!

See you soon, [[${customerName}]].

Regards,

The Thymeleaf team

</textarea>

--内联 JS <js 起止加入如下代码，否则引号嵌套或者"<"">"等不能用>

/*<![CDATA[*/

……

/*]]>*/

--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';

/* -]*/

4.不常用

--表达式

2）文字

a)文本文字 'one text','Another one',……

b)数字文字 0,34,3.0,12.3，……

c)布尔文字 true，flase

d)空文字 null

e)文字标记 one，sometext，main，……

3）文本处理

a)字符串连接 +

b)文字替换 | The name id ${name} |

4）算术表达式

a)基本表达式 +，-，*，/，%

b)减号（一元运算符） -

5）布尔表达式

a)基本表达式 and，or

b)布尔否定（一元运算符） !,not

6）比较和相等

a)比较 >，<，>=，<=(gt，lt，ge，le)

b)相等表达式 == ,! =(eq，ne)

7）条件表达式

a)If-then (if) ? (then)

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

c)Default (value) ? : (defaltvalue)

所有这些标签能够结合和嵌套：

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

--表达式基本对象

在上下文变量评估 OGNL 表达式时，一些对象表达式可获得更高的灵活性。这些对象将由#号开始引用。

- #ctx: 上下文对象.

- #vars: 上下文变量.

- #locale: 上下文语言环境.

- #ServletRequest: (仅在 web 上文)ServletRequest 对象.

- #Session: (仅在 web 上文) Session 对象.

--表达式功能对象

- #dates:java.util.Date 对象的实用方法。

- #calendars:和 dates 类似，但是 java.util.Calendar 对象.

- #numbers: 格式化数字对象的实用方法。

- #strings: 字符创对象的实用方法： contains，startsWith, prepending/appending 等.

- #objects: 对 objects 操作的实用方法。

- #bools: 对布尔值求值的实用方法。

- #arrays: 数组的实用方法。

- #lists: list 的实用方法。

- #sets: set 的实用方法。

- #maps: map 的实用方法。

- #aggregates: 对数组或集合创建聚合的实用方法。

- #messages: 在表达式中获取外部信息的实用方法。

- #ids: 处理可能重复的 id 属性的实用方法（比如：迭代的结果)。

--给特定的属性设值

下面是用 th:action 给 action 设值。

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

还有很多这样的属性，它们每一个都针对一个特定的 XHTML 或者 HTML5 属性：

th:text="${data}"

将 data 的值替换该属性所在标签的 body。字符常量要用引号，比如 th:text="'hello world'",th:text="2011+3",th:text="'my name is '+${user.name}"

th:utext

和 th:text 的区别是"unescaped text"。

th:with

定义变量，th:with="isEven=${prodStat.count}%2==0"，定义多个变量可以用逗号分隔。

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。

th:[tagAttr]

设置标签的各个属性，比如 th:value,th:action 等。

可以一次设置两个属性，比如：th:alt-title="#{logo}"

对属性增加前缀和后缀，用 th:attrappend，

th:attrprepend,比如：th:attrappend="class=${' '+cssStyle}"

对于属性是有些特定值的，比如 checked 属性，thymeleaf 都采用 bool 值，比如 th:checked=${user.isActive}

th:each

循 环 ， <tr th:each="user,userStat:${users}">,userStat 是状态变量，有 index,count,size,current,even,odd,first,last 等属性 ,如果没有显示设置状态变量， thymeleaf 会默 认给个 “变量名+Stat"的状态变量。

th:if or th:unless

条件判断，支持布尔值，数字（非零为 true)，字符，字符串等。

th:switch，th:case

选择语句。 th:case="*"表示 default case。