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、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。

    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（[http://www.thymeleaf.org/download.html](http://www.thymeleaf.org/download.html" \t "https://blog.csdn.net/qq_33381918/article/details/_blank) ）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="http://www.w3.org/1999/xhtml"

xmlns:th="http://www.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（[http://www.thymeleaf.org/download.html](http://www.thymeleaf.org/download.html" \t "https://blog.csdn.net/qq_33381918/article/details/_blank) ）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

IMG_256

<!-- 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.ServletContextTemplateResolver">

<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>

IMG_257

　　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

IMG_256

resource\_en\_US.properties：

home.welcome=Welcome to here！

resource\_zh\_CN.properties：  
  
home.welcome=欢迎您的到来！

IMG_257

　　　-- URL表达式  @{……}

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

          @{……}支持绝对路径和相对路径。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

<img src="../../static/assets/images/qr-code.jpg" th:src="@{${path}}" alt="二维码" />

　　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 &lt;em&gt;HTML&lt;/em&gt; text. &lt;b&gt;Enjoy yourself!&lt;/b&gt;”时，若页面代码如下则出现两种不同的结果

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

　　This is an &lt;em&gt;HTML&lt;/em&gt; text. &lt;b&gt;Enjoy yourself!&lt;/b&gt;

</div>   
<div th:utext="${html}">

　　This is an <em>HTML</em> text. <b>Enjoy yourself!</b>

</div>

　　　--表单中

IMG_258

<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>

IMG_259

　　 --显示页面的数据迭代

IMG_260

//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>

IMG_261

　　--条件判断

<span th:if="${product.price lt 100}" class="offer">Special offer!</span>

　　不能用"<”，">"等符号，要用"lt"等替代

IMG_262

<!-- 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>

IMG_263

IMG_264

<!--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>&nbsp;</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>

IMG_265

<!-- 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

IMG_266

<div class="form-group col-lg-6">

<label >性别<span>&nbsp;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>

IMG_267

　　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

IMG_268

<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>

IMG_269

　　--spring表达式语言

IMG_270

<!DOCTYPE html>

<html xmlns:th="http://www.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>

IMG_271

　 --内联

IMG_272

<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>

IMG_273

　--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.

        - #httpServletRequest: (仅在web上文)HttpServletRequest 对象.

        - #httpSession: (仅在web上文)  HttpSession 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