# 一、引用命名空间

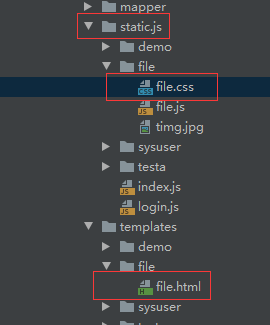
<html xmlns:th="http://www.thymeleaf.org">

在html中引入此命名空间，可避免编辑器出现html验证错误，虽然加不加命名空间对Thymeleaf的功能没有任何影响。

## 引入css

<link th:href="@{/font-awesome/css/font-awesome.css}" rel="stylesheet" />

文件结构如下所示



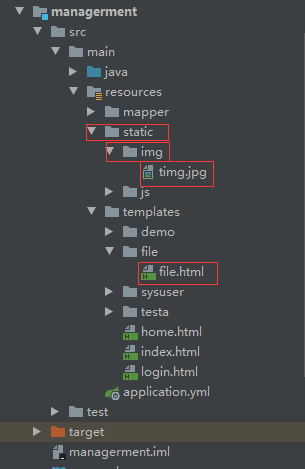
<link th:href="@{/js/file/file.css}" rel="stylesheet" />

引入js

<script type="text/javascript" th:src="@{/js/jquery-1.11.2.min.js}"></script>

## 引入img

目录结构



解释说明：

1.springboot2.0之后的静态资源文件需要添加/static/，在IDEA中测试路径写法是否正确是按住Ctrl键并用鼠标左键点击，若能跳转到资源则说明路径正确。

2.th:href 替换的是href属性，相对应的还有src，可以使用th:src=“@{/static/....}”进行替换。

eg:

|  |
| --- |
| <img th:src="@{/static/img/timg.jpg}" /> |

因为不知道公司给得案例配置了什么, 导致直接使用上述方式无法直接访问静态资源, 需要如下.

|  |
| --- |
| <img th:src="@{/img/timg.jpg}" /> |

# 二、输出内容

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

说明：

1. th:text 用来将内容输出到所在标签的body中。

2. #{home.welcome} 用来引入数据home对象中的 welcome属性。

3.可以用th:utext 用来显示“unescaped ” 的html内容。

2.2<p>Today is: <span th:text="${today}">13 February 2011</span></p>

说明：

${today} 用来引用 today 变量

# 三、访问对象

${param.x} 返回名为x 的 request参数。（可能有多个值）

${session.x} 返回名为x的Session参数。

${application.x} 返回名为 servlet context 的参数。

# 四、基本语法

4.1 #｛home.welcome｝ -- 访问数据

4.2 #{home.welcome(${session.user.name})} -- 格式化数据 当 home.welcome 为 "abcdegf{0}" 类似这种内容时。（多个参数以逗句分隔）。

4.3 ${today} --- 访问变量

4.4 访问基本对象

#ctx: the context object.

#vars: the context variables.

#locale: the context locale.

#request: (only in Web Contexts) the HttpServletRequest object.

#response: (only in Web Contexts) the HttpServletResponse object.

#session: (only in Web Contexts) the HttpSession object.

#servletContext: (only in Web Contexts) the ServletContext object.

其它公共对象参考:

http://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#appendix-a-expression-basic-objects

# 4.5 日期的输出

<span th:text="${#calendars.format(today,'dd MMMM yyyy')}">13 May 2011</span>

# 4.6 星号语法

<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>

# 4.7 输出URL

<a href="product/list.html" th:href="@{/product/list}">Product List</a>

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

# 4.8 使用代码段

<div th:insert="~{commons :: main}">...</div>

# 4.9 直接输出内容

<span th:text="'working web application'"> -- 输出字符

<span th:text="2013 + 2"> -- 输出数据表达式

<div th:if="${user.isAdmin()} == false"> --输出布尔表达式

<div th:if="${user.id}>2"> --输出判断表达式

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

# 4.10 条件表达式

<tr th:class="${user.id}? 'even' : 'odd'">

...

</tr>

<tr th:class="${user.id} > 2? 'even' : 'odd'">

...

</tr>

<tr th:class="${row.even}? 'alt'">

...省略 false 结果的表达方式

</tr>

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

...省略 true 结果的表达方式

<p>Age: <span th:text="\*{age}?: '(no age specified)'">27</span>.</p>

</div>

<span th:text="${user.name} ?: \_">no user authenticated</span> --不做任何处理时用下划线 \_ 表示

# 4.11 格式化

<td th:text="${{user.lastAccessDate}}">...</td> --${{.}} 调用默认的格式化器来输出结果。

# 4.12 预处理

<p th:text="${\_\_#{article.text('textVar')}\_\_}">Some text here...</p>

说明：thymeleaf 的处理模板内容的顺序与书写顺序无关，只能通过 \_\_${expression}\_\_ ，来将需要先一步计算出来后面 要用的变量指定为优化处理。

## th:selected动态下拉列表并返回选中值

参考文献

<https://www.cnblogs.com/person008/p/11434681.html>

功能说明

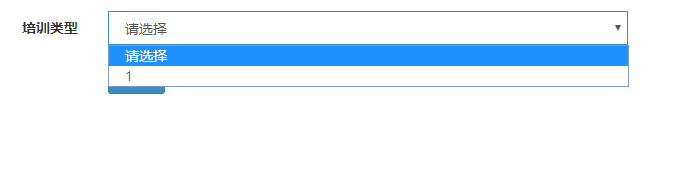
**[thymeleaf(th:each th:selected) 从后台动态获取下拉框数据回显及选中](https://www.cnblogs.com/person008/p/11434681.html)**

|  |
| --- |
| <div class="col-sm-8">  <select data-placeholder="请选择实验室" id="laboratoryId" name="laboratoryId" class="form-control chosen-select" style="width: 100%; " required >  <option value="" selected>请选择</option>  <option th:each="laboratoryDO:${laboratoryDOList}" th:value="${laboratoryDO.laboratoryId}" th:text="${laboratoryDO.labName}" ></option>  </select>  </div>  <script type="text/javascript" th:inline="javascript">  $().ready(function() {  var single = [[${laboratoryId}]];  $("#laboratoryId option[value= " + single + "]").prop("selected",true);  });</script> |

后台

|  |  |
| --- | --- |
| 1 | model.addAttribute("laboratoryId",laboratoryId); |

实现效果



# 五、设置 Attribute 值

5.1 设置任何Attribute 的方法

<input type="submit" value="Subscribe!" th:attr="value=#{subscribe.submit}"/> --设置单个

<img src="../../images/gtvglogo.png" th:attr="src=@{/images/gtvglogo.png},title=#{logo},alt=#{logo}" /> --一次设置多个

5.2 设置一些内置的Attribute的方法

<li><a href="product/list.html" th:href="@{/product/list}">Product List</a></li>

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

<input type="submit" value="Subscribe!" th:value="#{subscribe.submit}"/>

<img src="../../images/gtvglogo.png" th:src="@{/images/gtvglogo.png}" th:alt-title="#{logo}" /> -- 一次设置多个（alt title）的方法

其它的可用属性：http://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#setting-value-to-specific-attributes

5.3 设置html里没有指的任何属性的语法

<span th:whatever="${user.name}">...</span> ---whatever 可以换成任何你想设的属性

# 六、循环输出的语法

6.1 基本循环

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

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

<td th:text="${prod.price}">2.41</td>

<td th:text="${prod.inStock}? #{true} : #{false}">yes</td>

</tr>

6.2 循环状态的使用

<table>

<tr>

<th>NAME</th>

<th>PRICE</th>

<th>IN STOCK</th>

</tr>

<tr th:each="prod,iterStat : ${prods}" th:class="${iterStat.odd}? 'odd'">

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

<td th:text="${prod.price}">2.41</td>

<td th:text="${prod.inStock}? #{true} : #{false}">yes</td>

</tr>

</table>

关于状态的其它信息的使用详细参考：http://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#keeping-iteration-status

# 七、条件判断

7.1 if 和 unless

<a href="comments.html" th:href="@{/comments(prodId=${prod.id})}" th:unless="${#lists.isEmpty(prod.comments)}">view</a>

<a href="comments.html" th:href="@{/product/comments(prodId=${prod.id})}" th:if="${not #lists.isEmpty(prod.comments)}">view</a>

7.2 switch 语句

<div th:switch="${user.role}">

<p th:case="'admin'">User is an administrator</p>

<p th:case="${roles.manager}">User is a manager</p>

<p th:case="\*">User is some other thing</p> --默认的 case 相当于default

</div>

# 八、模板 include

8.1 定义和引用代码块

定义代码块

<!DOCTYPE html>

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

<body>

<div th:fragment="copy">

&copy; 2011 The Good Thymes Virtual Grocery

</div>

</body>

</html>

引用代码块

<body>

...

<div th:insert="~{footer :: copy}"></div>

</body>

引用未用fragment 标注的代码块

<div id="copy-section">

&copy; 2011 The Good Thymes Virtual Grocery

</div>

<body>

...

<div th:insert="~{footer :: #copy-section}"></div>

</body>

8.2 th:insert th:replace th:include 之间的区别

th:insert --- 插入代码块 th:replace -- 替换代码块会替换掉容器标签 th:include ---- 和insert相似但只会插入fragment标注body内的内容。

需要替换的片段内容：

<footer th:fragment="copy">

<script type="text/javascript" th:src="@{/plugins/jquery/jquery-3.0.2.js}"></script>

</footer>

导入片段：

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

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

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

结果为：

<div>

<footer>

<script type="text/javascript" th:src="@{/plugins/jquery/jquery-3.0.2.js}"></script>

</footer>

</div>

<footer>

<script type="text/javascript" th:src="@{/plugins/jquery/jquery-3.0.2.js}"></script>

</footer>

<div>

<script type="text/javascript" th:src="@{/plugins/jquery/jquery-3.0.2.js}"></script>

</div>

8.3 带参数的代码段

<div th:fragment="frag (onevar,twovar)">

<p th:text="${onevar} + ' - ' + ${twovar}">...</p>

</div>

<div th:replace="::frag (${value1},${value2})">...</div>

<div th:replace="::frag (onevar=${value1},twovar=${value2})">...</div>

# 九、局部变量的使用示例

<div th:with="firstPer=${persons[0]}">

<p>

The name of the first person is <span th:text="${firstPer.name}">Julius Caesar</span>.

</p>

</div>

<div th:with="firstPer=${persons[0]},secondPer=${persons[1]}">

<p>

The name of the first person is <span th:text="${firstPer.name}">Julius Caesar</span>.

</p>

<p>

But the name of the second person is

<span th:text="${secondPer.name}">Marcus Antonius</span>.

</p>

</div>

# 十、注释

<!-- ... -->

# 十一、说明

以上只列出Thymeleaf了简要常用的语法和使用方式，更多详情的说明和规则请参见：

http://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#introducing-thymeleaf