[Bài đọc] Template Layout

**1. Định nghĩa và tham chiếu fragments**

Trong các template, ta thường có nhu cầu sử dụng các phần từ những template khác, chẳng hạn như các footer, các header, các menu...

Để làm được điều đó, Thymeleaf cần chúng ta định ra các phần đó, được gọi là các "fragment", việc này có thể thực hiện được bằng cách sử dụng thuộc tính th:fragment.

Giả sử, chúng ta muốn thêm bản quyền của tác giả vào footer tất cả các trang của cửa hàng, do đó chúng ta tạo ra tệp /WEB-INF/templates/footer.html chứa khối mã này:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<body>  
<div th:fragment="copy">  
&copy; 2011 The Good Thymes Virtual Grocery  
</div>  
</body>  
</html>

 Khối lệnh trên định nghĩa 1 fragment tên là copy, chúng ta có thể sử dụng nó ở trang home bằng cách sử dụng thuộc tính th:insert hoặc th:replace như sau (hoặc th:include, tuy nhiên thuộc tính th:include không được khuyến khích sử dụng từ Thymeleaf 3.0)

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

Cú pháp của Fragment

Cú pháp trong Fragment khá đơn giản, có 3 dạng như sau:

**"~{templatename::selector}"** Chèn vào một Fragment có tên là “selector” được định nghĩa trong tamplate có tên là “templatename”

**"~{templatename}"** Chèn vào toàn bộ template có tên là “templatename”

Chú ý: tên của template được sử dụng trong thuộc tính th:insert/th:replate sẽ được nhận biết bằng Template Resolver, hiện nay chúng ta thường sử dụng Template Engine.

**~{::selector}"** hoặc **"~{this::selector}"**: được sử dụng trong trường hợp chèn 1 fragment từ cùng 1 template. Nếu không tìm thấy trên template thì sẽ gọi đến template khác cho đến khi tìm thấy selector trùng khớp.

Cả templatename và selector đều có đầy đủ tư cách tham gia vào các biểu thức, cả cả biểu thức điều kiện, như ví dụ sau:

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

Lưu ý: bộ ký tự ~{} là tùy chọn bắt buộc trong thuộc tính th:inser/th:replace

Fragment có thể bao gồm bất kỳ thuộc tính th:\* nào. Các thuộc tính này sẽ được đánh giá khi nó được đưa vào template đích, và chúng có thể tham chiếu đến bất kỳ ngữ cảnh nào trong template đích.

Một lợi thế của việc sử dụng phương pháp này là bạn có thể viết các fragment trong các trang, sau đó cấu trúc sắp xếp lại chúng để hiển thị lên trình duyệt. bằng cách sử dụng Thymeleaf, ta vẫn có khả năng sử dụng các fragment đó thành 1 template khác.

Tham chiếu đến fragment không sử dụng th:fragment

Ta có thể tham chiếu đến fragment mà không sử dụng bất kỳ thuộc tính th:fragment nào. Nó thậm chí có thể được tham chiếu đến mà không cần dùng đến kiến thức thymeleaf. Ví dụ

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

Ta có thể sử dụng fragment trên bằng cách sử dụng thuộc tính id của nó:

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

Điểm khác nhau giữa th:insert và th:replace (và th:include)

th:insert: Thực hiện chèn fragment như 1 thẻ con của thẻ có thuộc tính th:insert

th:replace: Thực hiện chèn fragment thay vào vị trí của thẻ có thuộc tính th:replace

th:include: Tương tự như  th:insert nhưng thay vì chèn toàn bộ fragment, nó chỉ chèn vào nội dung bên trong của thẻ fragment (thuộc tính này không được khuyến khích sử dụng từ Thymleaf 3.0)

Ta cùng xem ví dụ sau. Chúng ta có fragment “copy”

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

Ta thực hiện sử dụng 3 thẻ <div> có các thuộc tính th:insert, th:repalce, th:include như sau:

<body>  
...  
<div th:insert="footer :: copy"></div>  
<div th:replace="footer :: copy"></div>  
<div th:include="footer :: copy"></div>  
</body>

Và đây là kết quả:

<body>  
...  
 <div>  
 <footer>  
 &copy; 2011 The Good Thymes Virtual Grocery  
 </footer>  
 </div>  
 <footer>  
 &copy; 2011 The Good Thymes Virtual Grocery  
 </footer>  
 <div>  
 &copy; 2011 The Good Thymes Virtual Grocery  
 </div>  
</body>

**2. Sử dụng tham số trong fragments**

Để tạo một cơ chế cho fragment như 1 function, ta sử dụng 1 bộ tham số cho thẻ th:fragment.

<div th:fragment="frag (onevar,twovar)">  
 <p th:text="${onevar} + ' - ' + ${twovar}">...</p>  
</div>

Vậy khi tham chiếu đến fragment trên yêu cầu thẻ có thuộc tính th:inser, th:replace cần sử dụng 2 tham số để tham chiếu đến fragment này:

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

Chú ý trong tùy chọn cuối cùng, thứ tự các tham số không quan trọng, ví dụ ta có thể sử dụng khối mã sau để thay thế

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

**3. Sử dụng biến cục bộ trong fragment**

Kể cả với những fragment không có tham số như sau:

<div th:fragment="frag">  
 ...  
</div>

Chúng ta có thể dùng cách thứ 2 đã nói tới ở trên để tham chiếu đến chúng.

**Cách 1:**

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

Điều này tương đương với việc sử dụng kết hợp th:replace và th:with:

<div th:replace="::frag" th:with="onevar=${value1},twovar=${value2}">

Chú ý, với kỹ thuật sử dụng biến cục bộ cho fragment, bất kể có đối số hay không, chỉ cần làm cho các tham số bị trống trước khi thực thi, fragment sẽ vẫn truy cập đến các tham số được gọi đến bằng template hiện tại.

**4. th:assert cho những phép assertions trong template**

Trong thuộc tính th:assert, danh sách các tham số được phân cách bằng dấu “;”  chúng sẽ được đánh giá là true trong trường hợp không có ngoại lệ nào xảy ra.

<div th:assert="${onevar},(${twovar} != 43)">...</div>

Chúng ta có thể sử dụng thuộc tính này vào việc validate các tham số trong fragment, ví dụ sau:

<header th:fragment="contentheader(title)" th:assert="${!#strings.isEmpty(title)}">  
 ...  
</header>

**5. Layout linh hoạt**

Nhờ có thể sử dụng fragment như các biểu thức, chúng ta không chỉ có thể truyền những tham số text, đối số cho fragment, mà còn có thể là các fragment khác.

Kết quả là chúng ta có một cơ chế dàn template rất linh hoạt.

Ví dụ cách sử dụng title và link sau:

<head th:fragment="common\_header(title,links)">  
 <title th:replace="${title}">The awesome application</title>  
 <!-- Common styles and scripts -->  
 <link rel="stylesheet" type="text/css" media="all" th:href="@{/css/awesomeapp.css}">  
 <link rel="shortcut icon" th:href="@{/images/favicon.ico}">  
 <script type="text/javascript" th:src="@{/sh/scripts/codebase.js}"></script>  
 <!--/\* Per-page placeholder for additional links \*/-->  
 <th:block th:replace="${links}" />  
</head>

Chúng ta có thể gọi đến fragment như sau:

<head th:replace="base :: common\_header(~{::title},~{::link})">  
 <title>Awesome - Main</title>  
 <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}">  
 <link rel="stylesheet" th:href="@{/themes/smoothness/jquery-ui.css}">  
</head>

Và kết quả ta sẽ có các thẻ title và link như sau:

<head>  
 <title>Awesome - Main</title>  
 <!-- Common styles and scripts -->  
 <link rel="stylesheet" type="text/css" media="all" href="/awe/css/awesomeapp.css">  
 <link rel="shortcut icon" href="/awe/images/favicon.ico">  
 <script type="text/javascript" src="/awe/sh/scripts/codebase.js"></script>  
 <link rel="stylesheet" href="/awe/css/bootstrap.min.css">  
 <link rel="stylesheet" href="/awe/themes/smoothness/jquery-ui.css">  
</head>

Sử dụng fragment trống

Một empty fragment (~{}) có thể được sử đụng dể đánh dấu no markup. Như ví dụ sau:

<head th:replace="base :: common\_header(~{::title},~{})">  
 <title>Awesome - Main</title>  
</head>

Tham số thứ 2 của fragment được đặt là empty fragment (link) được đặt thành ~{} và không có gì được viết cho khối mã <th:block th:replace="${links}" />. Ta có kết quả:

<head>  
 <title>Awesome - Main</title>  
 <!-- Common styles and scripts -->  
 <link rel="stylesheet" type="text/css" media="all" href="/awe/css/awesomeapp.css">  
 <link rel="shortcut icon" href="/awe/images/favicon.ico">  
 <script type="text/javascript" src="/awe/sh/scripts/codebase.js"></script>  
</head>

**Sử dụng no-operation token**

No-op được sử dụng như 1 tham số cho fragment nếu chúng ra muốn nó nhận giá trị mặc định. Ví dụ về việc sử dụng common\_header:

<head th:replace="base :: common\_header(\_,~{::link})">  
 <title>Awesome - Main</title>  
 <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}">  
 <link rel="stylesheet" th:href="@{/themes/smoothness/jquery-ui.css}">  
</head>

Title trong fragment common\_header được set no-op (\_) ví dụ  như sau:

<title th:replace="${title}">The awesome application</title>

Kết quả

<head>  
 <title>The awesome application</title>  
 <!-- Common styles and scripts -->  
 <link rel="stylesheet" type="text/css" media="all" href="/awe/css/awesomeapp.css">  
 <link rel="shortcut icon" href="/awe/images/favicon.ico">  
 <script type="text/javascript" src="/awe/sh/scripts/codebase.js"></script>  
 <link rel="stylesheet" href="/awe/css/bootstrap.min.css">  
 <link rel="stylesheet" href="/awe/themes/smoothness/jquery-ui.css">  
</head>

Tùy chỉnh chèn có điều kiện Fragment

Fragment trống và no-operation token cho phép chúng ta chèn các fragment có điều kiện 1 cách dễ dàng.

Ví dụ, chúng ta chèn fragment common :: adminhead khi người dùng có quyền admin và chèn fragment trống trong các trường hợp khác:

<div th:insert="${user.isAdmin()} ? ~{common :: adminhead} : ~{}">  
 ...  
</div>

Hoặc chúng ta sử dụng no-operation token như sau:

<div th:insert="${user.isAdmin()} ? ~{common :: adminhead} : \_">  
 Welcome [[${user.name}]], click   
 <a th:href="@{/support}">here</a>   
 for help-desk support.  
</div>

Ngoài ra nếu chúng ta đã cấu hình template resolvers để kiểm tra sự có mặt của  template resources, ta có thể sử dụng sự có mặt fragment như một điều kiện trong một hoạt động mặc định.

<!-- The body of the <div> will be used if the "common :: salutation" fragment -->  
<!-- does not exist (or is empty). -->  
<div th:insert="~{common :: salutation} ?: \_">  
 Welcome [[${user.name}]], click <a th:href="@{/support}">here</a> for help-desk support.  
</div>

**6. Xóa các Fragments**

Ví dụ:

<table>  
 <tr>  
 <th>NAME</th>  
 <th>PRICE</th>  
 <th>IN STOCK</th>  
 <th>COMMENTS</th>  
 </tr>  
 <tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">  
 <td th:text="${prod.name}">Onions</td>  
 <td th:text="${prod.price}">2.41</td>  
 <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>  
 <td>  
 <span th:text="${#lists.size(prod.comments)}">2</span> comment/s  
 <a href="comments.html"  
 th:href="@{/product/comments(prodId=${prod.id})}"  
 th:unless="${#lists.isEmpty(prod.comments)}">view</a>  
 </td>  
 </tr>  
</table>

Đoạn mã này giống như 1 đoạn mẫu, nhưng trong trường hợp web tĩnh, không có Thymeleaf hỗ trợ nó sẽ không được xử lý.

Chúng ta cần thêm nhiều hàng hơn để thử nghiệm:

<table>  
 <tr>  
 <th>NAME</th>  
 <th>PRICE</th>  
 <th>IN STOCK</th>  
 <th>COMMENTS</th>  
 </tr>  
 <tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">  
 <td th:text="${prod.name}">Onions</td>  
 <td th:text="${prod.price}">2.41</td>  
 <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>  
 <td>  
 <span th:text="${#lists.size(prod.comments)}">2</span> comment/s  
 <a href="comments.html"  
 th:href="@{/product/comments(prodId=${prod.id})}"  
 th:unless="${#lists.isEmpty(prod.comments)}">view</a>  
 </td>  
 </tr>  
 <tr class="odd">  
 <td>Blue Lettuce</td>  
 <td>9.55</td>  
 <td>no</td>  
 <td>  
 <span>0</span> comment/s  
 </td>  
 </tr>  
 <tr>  
 <td>Mild Cinnamon</td>  
 <td>1.99</td>  
 <td>yes</td>  
 <td>  
 <span>3</span> comment/s  
 <a href="comments.html">view</a>  
 </td>  
 </tr>  
</table>

Giờ chúng ta có 3 hàng, chúng ta sẽ xử lý Thymeleaf:

<table>  
 <tr>  
 <th>NAME</th>  
 <th>PRICE</th>  
 <th>IN STOCK</th>  
 <th>COMMENTS</th>  
 </tr>  
 <tr>  
 <td>Fresh Sweet Basil</td>  
 <td>4.99</td>  
 <td>yes</td>  
 <td>  
 <span>0</span> comment/s  
 </td>  
 </tr>  
 <tr class="odd">  
 <td>Italian Tomato</td>  
 <td>1.25</td>  
 <td>no</td>  
 <td>  
 <span>2</span> comment/s  
 <a href="/gtvg/product/comments?prodId=2">view</a>  
 </td>  
 </tr>  
 <tr>  
 <td>Yellow Bell Pepper</td>  
 <td>2.50</td>  
 <td>yes</td>  
 <td>  
 <span>0</span> comment/s  
 </td>  
 </tr>  
 <tr class="odd">  
 <td>Old Cheddar</td>  
 <td>18.75</td>  
 <td>yes</td>  
 <td>  
 <span>1</span> comment/s  
 <a href="/gtvg/product/comments?prodId=4">view</a>  
 </td>  
 </tr>  
 <tr class="odd">  
 <td>Blue Lettuce</td>  
 <td>9.55</td>  
 <td>no</td>  
 <td>  
 <span>0</span> comment/s  
 </td>  
 </tr>  
 <tr>  
 <td>Mild Cinnamon</td>  
 <td>1.99</td>  
 <td>yes</td>  
 <td>  
 <span>3</span> comment/s  
 <a href="comments.html">view</a>  
 </td>  
 </tr>  
</table>

Thymeleaf đã không bỏ qua 2 hàng cuối, chúng ta sử dụng thuộc tính th:remove vào thẻ <tr> thứ 3:

<table>  
 <tr>  
 <th>NAME</th>  
 <th>PRICE</th>  
 <th>IN STOCK</th>  
 <th>COMMENTS</th>  
 </tr>  
 <tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">  
 <td th:text="${prod.name}">Onions</td>  
 <td th:text="${prod.price}">2.41</td>  
 <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>  
 <td>  
 <span th:text="${#lists.size(prod.comments)}">2</span> comment/s  
 <a href="comments.html"  
 th:href="@{/product/comments(prodId=${prod.id})}"  
 th:unless="${#lists.isEmpty(prod.comments)}">view</a>  
 </td>  
 </tr>  
 <tr class="odd" th:remove="all">  
 <td>Blue Lettuce</td>  
 <td>9.55</td>  
 <td>no</td>  
 <td>  
 <span>0</span> comment/s  
 </td>  
 </tr>  
 <tr th:remove="all">  
 <td>Mild Cinnamon</td>  
 <td>1.99</td>  
 <td>yes</td>  
 <td>  
 <span>3</span> comment/s  
 <a href="comments.html">view</a>  
 </td>  
 </tr>  
</table>

Sau xử lý ta sẽ có mã:

<table>  
 <tr>  
 <th>NAME</th>  
 <th>PRICE</th>  
 <th>IN STOCK</th>  
 <th>COMMENTS</th>  
 </tr>  
 <tr>  
 <td>Fresh Sweet Basil</td>  
 <td>4.99</td>  
 <td>yes</td>  
 <td>  
 <span>0</span> comment/s  
 </td>  
 </tr>  
 <tr class="odd">  
 <td>Italian Tomato</td>  
 <td>1.25</td>  
 <td>no</td>  
 <td>  
 <span>2</span> comment/s  
 <a href="/gtvg/product/comments?prodId=2">view</a>  
 </td>  
 </tr>  
 <tr>  
 <td>Yellow Bell Pepper</td>  
 <td>2.50</td>  
 <td>yes</td>  
 <td>  
 <span>0</span> comment/s  
 </td>  
 </tr>  
 <tr class="odd">  
 <td>Old Cheddar</td>  
 <td>18.75</td>  
 <td>yes</td>  
 <td>  
 <span>1</span> comment/s  
 <a href="/gtvg/product/comments?prodId=4">view</a>  
 </td>  
 </tr>  
</table>

Thuộc tính th:remove hoạt động theo 5 cách khác nhau dựa vào giá trị của nó:

* All: Xóa bỏ tất cả các thẻ bao gồm cả thẻ con của nó
* Body: Không xóa thẻ chứa th:remove , chỉ xóa thẻ con của nó
* Tag: Xóa thẻ chứa thuộc tính th:remove, không xóa thẻ con của nó
* All-but-first: Xóa tất cả thẻ con trừ thẻ con đầu tiên
* None: không xóa.

Ví dụ: sau đây là ví dụ về sử dụng phương thức all-but-first

<table>  
 <thead>  
 <tr>  
 <th>NAME</th>  
 <th>PRICE</th>  
 <th>IN STOCK</th>  
 <th>COMMENTS</th>  
 </tr>  
 </thead>  
 <tbody th:remove="all-but-first">  
 <tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">  
 <td th:text="${prod.name}">Onions</td>  
 <td th:text="${prod.price}">2.41</td>  
 <td th:text="${prod.inStock}? #{true} : #{false}">yes</td>  
 <td>  
 <span th:text="${#lists.size(prod.comments)}">2</span> comment/s  
 <a href="comments.html"  
 th:href="@{/product/comments(prodId=${prod.id})}"  
 th:unless="${#lists.isEmpty(prod.comments)}">view</a>  
 </td>  
 </tr>  
 <tr class="odd">  
 <td>Blue Lettuce</td>  
 <td>9.55</td>  
 <td>no</td>  
 <td>  
 <span>0</span> comment/s  
 </td>  
 </tr>  
 <tr>  
 <td>Mild Cinnamon</td>  
 <td>1.99</td>  
 <td>yes</td>  
 <td>  
 <span>3</span> comment/s  
 <a href="comments.html">view</a>  
 </td>  
 </tr>  
 </tbody>  
</table>

Thuộc tính th:remove có thể lấy bất kỳ giá trị nào của Thymeleaf Standard Expression, miễn là nó trả về 1 trong các giá trị String nhận giá trị (all, tag, body, all-but-first or none)

Ví dụ với thuộc tính th:remove có điều kiện như sau:

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

Hoặc ví dụ sau đây:

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

Trong trường hợp này, ${condition} trả về false, không có thẻ nào được remove

Last modified: Saturday, 23 November 2019, 3:27 PM