Thymeleaf Tutorial II

Interactive Tutorial

Project version	2.1.2.	
Project web site	http://www.thymeleaf.org	
Generation info.	Generated by Kunner	
	v1.0	2013.12.13

Translated by kunner, all rights are **not** reserved.

본 자습서는 Thymeleaf Interactive Tutorial(http://itutorial.thymeleaf.org)을 기초로 작성되었습니다. Thymeleaf에서 제공하는 자습서의 내용과 함께 각 예제와 관련된 구문의 사용법, 현업의 적용 방안 등에 대해 추가 기술하였습니다. – Kunner.

Translated by kunner, all rights are **not** reserved.

Exercise 1. Bean values

매뉴얼 관련 정보: Using Thymeleaf 4장

원본 HTML의 소스코드를 확인하고 XXX 라고 되어 있는 부분이 Product(Appendix 1 참조)의 bean 값으로 제대로 대체될 수 있도록 코드를 작성하는 예제 입니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 2</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 1: bean values</h1>
10         <h2>Product information</h2>
11         <dl>
12             <dt>Product name</dt>
13             <dd>XXX</dd>
14             <dt>Product price</dt>
15             <dd>XXX</dd>
16             <dt>Product available from</dt>
17             <dd>XXX</dd>
18         </dl>
19     </body>
20 </html>
```

코드 13, 15, 17 라인의 XXX 를 Product Class의 Bean으로 교체하기 위해 th:text="${class.bean}" 구문을 이용합니다.

적용

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 2</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
```

Translated by kunner, all rights are **not** reserved.

```
 9          <h1>Thymeleaf tutorial - Answer for exercise 1: bean values</h1>
10          <h2>Product information</h2>
11          <dl>
12              <dt>Product name</dt>
13              <dd th:text="${product.description}">Red chair</dd>
14              <dt>Product price</dt>
15              <dd th:text="${product.price}">350</dd>
16              <dt>Product available from</dt>
17              <dd th:text="${product.availableFrom}">28-Jun-2013</dd>
18          </dl>
19      </body>
20  </html>
```

13번째 라인의 <dd th:text="${product.description}">Red Chair</dd> 가 서버에서 실행되면 Red Chair 가 product.description 값으로 대체 됩니다. 이는 각각 15, 17번째 라인에도 동일하게 적용됩니다.

Exercise 2. Simple 포맷 변환

매뉴얼 관련 정보: Using Thymeleaf 16장

Thymeleaf의 포매팅을 이용해 날짜와 금액을 적절한 포맷으로 변환해 표기합니다.

원본 HTML

```
 1  <!DOCTYPE html>
 2  <html xmlns:th="http://www.thymeleaf.org">
 3      <head>
 4          <title>Thymeleaf tutorial: exercise 2</title>
 5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
 6          <meta charset="utf-8" />
 7      </head>
 8      <body>
 9          <h1>Thymeleaf tutorial - Answer for exercise 2: bean values</h1>
10          <h2>Product information</h2>
11          <dl>
12              <dt>Product name</dt>
13              <dd>XXX</dd>
14              <dt>Product price</dt>
15              <dd>XXX</dd>
```

Translated by kunner, all rights are **not** reserved.

```
16                  <dt>Product available from</dt>
17                  <dd>XXX</dd>
18              </dl>
19          </body>
20      </html>
```

Exercise 1 에서 했던 것처럼 먼저 th:text 로 13, 15, 17 라인의 각 항목을 Bean 값으로 교체합니다.
그리고 금액을 표기하는 price 와 날짜를 표기하는 available from에 Thymeleaf formatter를 적용합니다.

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 2</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 2: bean values</h1>
10          <h2>Product information</h2>
11          <dl>
12              <dt>Product name</dt>
13              <dd th:text="${product.description}">Red chair</dd>
14              <dt>Product price</dt>
15              <dd th:text="${#numbers.formatDecimal(product.price, 1, 2)}">350</dd>
16              <dt>Product available from</dt>
17              <dd th:text="${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}">28-Jun-
    2013</dd>
18          </dl>
19      </body>
20  </html>
```

보다 자세한 formatter의 사용 방법은 Using Thymeleaf 매뉴얼 16장을 확인하시기 바랍니다.

Exercise 3. 문자열 결합

매뉴얼 관련 정보: Using Thymeleaf 4-6장

th:text에 모델, 또는 변수에 문자열을 결합해 출력할 수 있습니다. 이 기능을 이용하여 웹사이트에서
금액을 표기할 때 통화를 표시할 수 있습니다. Thymeleaf의 문자열 결합은 자바스크립트의 문자열 결합과

Translated by kunner, all rights are **not** reserved.

동일한 방법으로 처리할 수 있습니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 3</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 3: string concatenation</h1>
10         <h2>Product information</h2>
11         <dl>
12             <dt>Product price</dt>
13             <dd>XXX</dd>
14         </dl>
15     </body>
16 </html>
```

먼저 13번째 라인의 XXX를 th:text 구문을 이용하여 product.price로 교체합니다. 그 후 문자열 결합 기능을 이용하여 product.price 앞에 $ 를 표기 합니다.

적용

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 3</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 3: string concatenation</h1>
10         <h2>Product information</h2>
11         <dl>
12             <dt>Product price</dt>
13             <dd th:text="${'$' + #numbers.formatDecimal(product.price, 1, 2)}">$350</dd>
14         </dl>
15     </body>
16 </html>
```

Translated by kunner, all rights are **not** reserved.

Exercise 4. 국제화(다국어 메시지 처리)

매뉴얼 관련 정보: Using Thymeleaf 3장

Thymeleaf는 message.properties 파일을 이용해 다국어 메시지를 처리할 수 있습니다. 다국어 메시지의 위치는 org.thymeleaf.messageresolver.IMessageResolver를 구현하여 지정할 수 있습니다. (매뉴얼 참조)

본 예제에서는 message.properties가 이미 만들어져 있고(Appendix 2 참조), 사용자의 요청에 따라 locale 정보가 넘어 왔다고 가정한 후, Thymeleaf가 message.properties의 메시지를 호출하는 방법을 다루게 될 것입니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 4</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 4: internationalization</h1>
10         <h2>Product information</h2>
11         <dl>
12             <dt>Product name</dt>
13             <dd th:text="${product.description}">Red chair</dd>
14             <dt>Product price</dt>
15             <dd th:text="${#numbers.formatDecimal(product.price, 1, 2)}">350</dd>
16             <dt>Product available from</dt>
17             <dd th:text="${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}">28-Jun-
   2013</dd>
18         </dl>
19     </body>
20 </html>
```

위 예제에서 지역화 메시지로 처리될 부분은 4, 9, 10, 12, 14, 16번 라인 입니다.

지역화 메시지를 표시하려면 th:text="#{메시지key}" 구문을 이용하면 됩니다.

적용

Translated by kunner, all rights are **not** reserved.

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title th:text="#{tutorial.exercise4}">Thymeleaf tutorial: exercise 4</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1 th:text="#{tutorial.exercise4}">Thymeleaf tutorial - Solution for exercise 4:
   internationalization</h1>
10         <h2 th:text="#{product.info}">Product information</h2>
11         <dl>
12             <dt th:text="#{product.name}">Product name</dt>
13             <dd th:text="${product.description}">Red chair</dd>
14             <dt th:text="#{product.price}">Product price</dt>
15             <dd th:text="${#numbers.formatDecimal(product.price, 1, 2)}">350</dd>
16             <dt th:text="#{product.available}">Product available from</dt>
17             <dd th:text="${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}">28-Jun-
   2013</dd>
18         </dl>
19     </body>
20 </html>
```

Exercise 5. 문자열 이스케이프

매뉴얼 관련 정보: Using Thymeleaf 3-2장

이스케이프란 문자열에 <나 > 등 HTML 태그에 해당하는 문자가 포함된 경우 화면에 출력할 때 HTML이 표시되지 않도록 <를 <로 >를 >로 변경하거나 스크립트나 특수 문자열을 화면에 제대로 출력되도록 변환하는 것을 말합니다. 반대로 HTML을 그대로 출력하고자 하는 경우에는 <를 <로 바꾸고 이를 언이스케이프 라고 합니다.

Thymeleaf에서는 자동으로 문자열을 이스케이프 처리 하지만, 필요에 따라 언이스케이프해야 하는 경우 th:utext="${문자열}" 구문을 이용해 처리할 수 있습니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 5</title>
```

Translated by kunner, all rights are **not** reserved.

```
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 5: escaped and unescaped text</h1>
10          <div>
11              Some escaped text
12          </div>
13          <div>
14              Some unescaped text
15          </div>
16      </body>
17  </html>
```

위 예제에서 10, 13번 라인을 각각 th:text 와 th:utext로 처리합니다. 예제에서 사용된 변수는 html 입니다.

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 5</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 5: escaped and unescaped text</h1>
10          <div th:text="${html}">
11              Some escaped text
12          </div>
13          <div th:utext="${html}">
14              Some unescaped text
15          </div>
16      </body>
17  </html>
```

이렇게 언이스케이프 처리가 필요한 부분에 대해 th:utext 를 이용하면 됩니다.

Exercise 6. 반복

매뉴얼 관련 정보: Using Thymeleaf 6장

Translated by kunner, all rights are **not** reserved.

Thymeleaf은 java.util.List, java.util.Iterable, java.util.Map 등 배열이나 리스트 형태의 객체에 대해 반복 처리할 수 있는 기능을 가지고 있습니다. 이 기능은 게시판이나 상품 목록 등 반복되는 형태의 목록을 출력 할 때 유용합니다.

Thymeleaf의 반복 처리의 사용법은 매우 간단합니다.

th:each="식별자: ${객체명}" 로 반복 처리의 시작점과 반복에 사용할 객체를 선언하고, 실제 출력되는 부분에 th:text="${식별자.빈}" 로 내용을 출력하면 됩니다.

또 프로토타입(정적 HTML)을 위해 반복 행을 HTML로 하드 코딩했을 경우, th:remove 어트리뷰트를 사용하면 서버 실행 시 해당 반복 행을 자동으로 제거해 줍니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 6</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 6: iteration</h1>
10         <h2>Product list</h2>
11         <table>
12             <thead>
13                 <tr>
14                     <th>Description</th>
15                     <th>Price</th>
16                     <th>Available from</th>
17                 </tr>
18             </thead>
19             <tbody>
20                 <tr>
21                     <td>XXX</td>
22                     <td>XXX</td>
23                     <td>XXX</td>
24                 </tr>
25             </tbody>
26         </table>
27     </body>
28 </html>
```

위 예제에서 반복이 시작되는 20번째 라인의 <tr>에 th:each 구문을 적용합니다. 또, 19번째 라인의 <tbody>에 th:remove를 적용하여 20~24번째 라인이 자동으로 제거 되도록 합니다. 이에 따라 코드를

Translated by kunner, all rights are **not** reserved.

다음과 같이 변경합니다.

적용

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 6</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Solution for exercise 6: iteration</h1>
10         <h2>Product list</h2>
11         <table>
12             <thead>
13                 <tr>
14                     <th>Description</th>
15                     <th>Price</th>
16                     <th>Available from</th>
17                 </tr>
18             </thead>
19             <tbody th:remove="all-but-first">
20                 <tr th:each="product : ${productList}">
21                     <td th:text="${product.description}">Red chair</td>
22                     <td th:text="${'$' + #numbers.formatDecimal(product.price, 1,
   2)}">$350</td>
23                     <td th:text="${#dates.format(product.availableFrom, 'dd-MMM-
   yyyy')}">28-Jun-2013</td>
24                 </tr>
25                 <tr>
26                     <td>White table</td>
27                     <td>$200</td>
28                     <td>15-Jul-2013</td>
29                 </tr>
30                 <tr>
31                     <td>Reb table</td>
32                     <td>$200</td>
33                     <td>15-Jul-2013</td>
34                 </tr>
35                 <tr>
36                     <td>Blue table</td>
37                     <td>$200</td>
38                     <td>15-Jul-2013</td>
```

Translated by kunner, all rights are **not** reserved.

```
39                  </tr>
40              </tbody>
41          </table>
42      </body>
43  </html>
```

위 코드에서 25 ~ 39번째 라인은 하드 코딩된 프로토타입용 HTML 입니다. 브라우저에서 정적으로 페이지를 열었을 때는 25~39 라인이 출력되어 보여집니다. 그런데 서버에서 실행됐을 때는 해당 부분이 제거되어야 하므로 19번째 라인에 th:remove 어트리뷰트를 적용해 자동 제거하도록 하였습니다.

th:remove 어트리뷰트의 속성값 all-but-first는 제목 표시줄인 첫번째 행만 남기고 나머지는 제거하라는 옵션입니다. 이외의 다른 속성값에 대해서는 Using Thymeleaf 매뉴얼 8-3장을 참고하시기 바랍니다.

Exercise 7. 반복 상태 변수

매뉴얼 관련 정보: Using Thymeleaf 6-2장

반복 상태 변수(iter status variable)를 이용해 반복 행의 현재 상태를 확인할 수 있습니다. 이 기능은 반복 행을 처리할 때 특정 행에 대해서만 독특한 처리를 할 수 있게 합니다. 이를테면 짝수 행에는 배경색을 넣고, 홀수 행에는 배경색을 제거하는 등의 처리가 가능합니다.

또 첫번째 행을 제목 행으로 다른 CSS를 적용하는 등 리스트를 출력할 때 다양한 방법으로 활용할 수 있습니다. 심지어 리스트 내 특정 요소 값의 총합을 구하는 처리도 가능합니다.

사용법은 역시 매우 간단해서 th:each 에서 반복 상태 변수를 선언하기만 하면 리스트에서 바로 사용 가능합니다.

원본 HTML

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 7</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 7: iteration stats</h1>
10          <h2>Product list</h2>
11          <table>
12              <thead>
13                  <tr>
14                      <th>Index</th>
```

Translated by kunner, all rights are **not** reserved.

```
15                      <th>Description</th>
16                      <th>Price</th>
17                      <th>Available from</th>
18                  </tr>
19              </thead>
20              <tbody>
21                  <tr th:each="product : ${productList}">
22                      <td>XXX</td>
23                      <td th:text="${product.description}">Red chair</td>
24                      <td th:text="${'$' + #numbers.formatDecimal(product.price, 1, 2)}">$350</td>
25                      <td th:text="${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}">28-Jun-
    2013</td>
26                  </tr>
27              </tbody>
28          </table>
29      </body>
30  </html>
```

위 코드에서 20번째 라인 <tbody>에 th:remove를 적용하고, 27번째 라인 뒤에 프로토타입을 위한 HTML 코드를 삽입합니다. th:remove의 속성값은 앞선 예제와 같이 all-but-first를 적용합니다.

또 21번째 라인의 th:each에 반복 상태 변수를 선언합니다. 반복 상태 변수는 식별자 뒤에 변수명만 선언해 주면 되며, 생략 가능합니다. 만약 상태 변수 선언은 생략하면, Thymeleaf는 자동으로 식별자명에 Stat를 붙여 상태 변수를 만들어 줍니다.

명시적 선언	th:each="식별자, 반복상태변수명: ${객체명}"
묵시적 선언(생략가능)	th:each="식별자, ${객체명}" → 식별자Stat 이라는 이름의 상태변수가 자동으로 생성됨

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 7</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 7: iteration stats</h1>
10          <h2>Product list</h2>
11          <table>
12              <thead>
13                  <tr>
```

Translated by kunner, all rights are **not** reserved.

```
14                      <th>Index</th>
15                      <th>Description</th>
16                      <th>Price</th>
17                      <th>Available from</th>
18                  </tr>
19              </thead>
20              <tbody th:remove="all-but-first">
21                  <tr th:each="product : ${productList}">
22                      <td th:text="${productStat.count}">1</td>
23                      <td th:text="${product.description}">Red chair</td>
24                      <td th:text="${'$' + #numbers.formatDecimal(product.price, 1, 2)}">$350</td>
25                      <td th:text="${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}">28-Jun-
    2013</td>
26                  </tr>
27                  <tr>
28                      <td>2</td>
29                      <td>White table</td>
30                      <td>$200</td>
31                      <td>15-Jul-2013</td>
32                  </tr>
33                  <tr>
34                      <td>3</td>
35                      <td>Reb table</td>
36                      <td>$200</td>
37                      <td>15-Jul-2013</td>
38                  </tr>
39                  <tr>
40                      <td>4</td>
41                      <td>Blue table</td>
42                      <td>$200</td>
43                      <td>15-Jul-2013</td>
44                  </tr>
45              </tbody>
46          </table>
47      </body>
48  </html>
```

21번째 라인에서 명시적으로 반복상태변수를 선언하지 않아도, Thymeleaf는 자동으로 반복상태변수를 생성합니다. 이때 생성되는 반복상태변수의 이름은 식별자Stat 입니다. 따라서 본 예제에서 반복상태변수는 productStat 으로 생성되었습니다.

22번째 라인에서 반복 순서에 따라 productStat의 값을 출력하도록 했습니다. Thymeleaf는 출력값에 대한 기본 연산을 지원하므로, 이를 이용해 게시판의 목록 번호를 페이지 번호와 연동해 처리할 수도 있습니다.

반복상태변수의 활용에 대한 자세한 설명은 Using Thymeleaf 매뉴얼 6-2장을 참고하시기 바랍니다.

Translated by kunner, all rights are **not** reserved.

Exercise 8. 조건 제어 I

매뉴얼 관련 정보: Using Thymeleaf 7장

출력값 중 price가 $100 미만인 경우 "Special Offer!" 라는 문자열을 함께 출력하도록 하는 예제로, 가격(price)에 대해 설정된 조건을 평가하고, 결과에 따라 서로 다르게 처리되도록 해야 합니다.

이를 위해 th:if 구문을 이용합니다. 이외에도 switch 구문도 사용할 수 있습니다. 자세한 사용 방법은 매뉴얼의 7장을 참고하시기 바랍니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 8</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 8: conditions</h1>
10         <h2>Product list</h2>
11         <table>
12             <thead>
13                 <tr>
14                     <th>Description</th>
15                     <th>Price</th>
16                     <th>Available from</th>
17                     <th></th>
18                 </tr>
19             </thead>
20             <tbody>
21                 <tr th:each="product : ${productList}">
22                     <td th:text="${product.description}">Red chair</td>
23                     <td th:text="${'$' + #numbers.formatDecimal(product.price, 1, 2)}">$350</td>
24                     <td th:text="${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}">28-Jun-
   2013</td>
25                     <td>
26                         <span class="offer">Special offer!</span>
27                     </td>
28                 </tr>
29             </tbody>
```

Translated by kunner, all rights are **not** reserved.

```
30          </table>
31      </body>
32  </html>
```

위 코드에서 조건의 판별 결과에 따라 변경될 부분은 26번째 라인입니다. price가 $100 미만인 경우에만 출력하도록 적용해 보겠습니다.

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 8</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 8: conditions</h1>
10          <h2>Product list</h2>
11          <table>
12              <thead>
13                  <tr>
14                      <th>Description</th>
15                      <th>Price</th>
16                      <th>Available from</th>
17                      <th></th>
18                  </tr>
19              </thead>
20              <tbody th:remove="all-but-first">
21                  <tr th:each="product : ${productList}">
22                      <td th:text="${product.description}">Red chair</td>
23                      <td th:text="${'$' + #numbers.formatDecimal(product.price, 1, 2)}">$350</td>
24                      <td th:text="${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}">28-Jun-
    2013</td>
25                      <td>
26                          <span th:if="${product.price lt 100}" class="offer">Special
    offer!</span>
27                      </td>
28                  </tr>
29                  <tr>
30                      <td>White table</td>
31                      <td>$200</td>
32                      <td>15-Jul-2013</td>
```

Translated by kunner, all rights are **not** reserved.

```
33                      <td></td>
34                  </tr>
35                  <tr>
36                      <td>Reb table</td>
37                      <td>$150</td>
38                      <td>15-Jul-2013</td>
39                      <td><span class="offer">Special offer!</span></td>
40                  </tr>
41                  <tr>
42                      <td>Blue table</td>
43                      <td>$200</td>
44                      <td>15-Jul-2013</td>
45                      <td></td>
46                  </tr>
47              </tbody>
48          </table>
49      </body>
50  </html>
```

만약 100$ 미만인 경우에는 "Special Offer!"를, 200$ 미만인 경우에는 "Cool Offer!"를 출력해야 하는 경우는 어떻게 해야 할까요? 이렇게 복수의 조건을 판별해야 하는 경우에 대해 다음 예제에서 확인해 보겠습니다.

이와 같은 비즈니스 로직(조건 처리)이 템플릿 엔진에서 처리되도록 하는 것은 데이터와 비즈니스 로직의 분리에 도움이 됩니다. 비즈니스 정책이 변경되는 경우, 즉 $100 이 아니라 $80 미만 상품에 대해서만 문자열을 출력하라는 요구사항이 접수되는 경우 기존처럼 개발 코드를(극단적인 예로 Class 변경 등) 변경하거나 하지 않아도 되기 때문입니다. 특히 관리자도구에서 동적으로 생성한 코드 조각(i.e. Structured Contents)과 연동하는 경우 매우 유연한 시스템이 될 수 있습니다. 다만, DAO와 서비스가 강하게 결합되어 있는 기존 시스템에서 이를 즉각 반영하기는 어렵습니다. 이에 대해서는 템플릿 개발/수정/관리의 업무 정책과 함께 고민해야 할 것입니다.

그러나, DAO와 서비스를 분리하고 서비스가 UI를 동적으로 재구성할 수 있도록 설계된 아키텍처는 업무 생산성을 비약적으로 향상 시킬 수 있을 것입니다. Thymeleaf는 이를 가능하게 하는 템플릿 엔진입니다.

Exercise 9. 조건 제어 Ⅱ

매뉴얼 관련 정보: Using Thymeleaf 7-2장

단일 조건이 아니라 복수의 조건을 판별해야 하는 경우 Switch – Case 구문을 이용해 처리할 수 있습니다.

이 장에서는 앞서 알아 보았던 th:if 외에 Thymeleaf에서 제공하는 다른 조건 제어 구문, 즉 th:unless, th:switch ~ th:case, 그리고 별도 어트리뷰트 없이 th:text에 바로 조건식을 사용하는 방법에 대해 알아

Translated by kunner, all rights are **not** reserved.

보겠습니다.

본 예제에서 사용한 객체 정보는 Appendix 3~5에 기술되어 있습니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 9</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 9: more on conditions</h1>
10         <h2>Customer list</h2>
11         <table>
12             <thead>
13                 <tr>
14                     <th>First name</th>
15                     <th>Last name</th>
16                     <th>Gender</th>
17                     <th>Payment method</th>
18                     <th>Balance</th>
19                 </tr>
20             </thead>
21             <tbody>
22                 <tr th:each="customer : ${customerList}">
23                     <td th:text="${customer.firstName}">Peter</td>
24                     <td th:text="${customer.lastName}">Jackson</td>
25                     <!--
26                         Use th:switch for selecting content based on ${customer.gender}.
27                         As genre can be null if unknown, better use ${customer.gender?.name()}
28                         for obtaining its name.
29                     -->
30                     <td>
31                         <img alt="Male" /> <!-- Use "/images/male.png" image -->
32                         <img alt="Female" /> <!-- Use "/images/female.png" image -->
33                         <span>Unknown</span>
34                     </td>
35                     <td>
36                         <span th:text="${customer.paymentMethod.description}">Direct
   debit</span>
37                         <!-- Show next message only when paymentMethod is not CREDIT_CARD
   -->
```

Translated by kunner, all rights are **not** reserved.

```
38                          <span class="warn">
39                              Payment must be done in the next 4 days
40                          </span>
41                      </td>
42                      <!-- Add class="enhanced" when balance is greater than 10000 -->
43                      <td th:text="${customer.balance}">350</td>
44                  </tr>
45              </tbody>
46          </table>
47      </body>
48  </html>
```

1) 25~29번째 라인의 주석 내용을 보면 고객의 목록을 출력할 때 고객 성별에 따라 다른 아이콘을 출력하고, 성별을 모르는 경우 Unknown을 출력합니다.
2) 37번째 라인의 주석 내용은 고객의 결제 방식이 카드가 아닌 경우, 안내 메시지를 출력합니다.
3) 42번째 라인 주석에 따라 고객의 잔고가 10000을 넘는 경우 enhanced 라는 CSS 클래스를 적용합니다.

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 9</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 9: more on conditions</h1>
10          <h2>Customer list</h2>
11          <table>
12              <thead>
13                  <tr>
14                      <th>First name</th>
15                      <th>Last name</th>
16                      <th>Gender</th>
17                      <th>Payment method</th>
18                      <th>Balance</th>
19                  </tr>
20              </thead>
21              <tbody th:remove="all-but-first">
22                  <tr th:each="customer : ${customerList}">
```

Translated by kunner, all rights are **not** reserved.

```
23                      <td th:text="${customer.firstName}">Peter</td>
24                      <td th:text="${customer.lastName}">Jackson</td>
25                      <td th:switch="${customer.gender?.name()}">
26                          <img th:case="'MALE'" src="../../../images/male.png"
    th:src="@{/images/male.png}" alt="Male" />
27                          <img th:case="'FEMALE'" src="../../../images/female.png"
    th:src="@{/images/female.png}" alt="Female" />
28                          <span th:case="*">Unknown</span>
29                      </td>
30                      <td>
31                          <span th:text="${customer.paymentMethod.description}">Direct
    debit</span>
32                          <span th:unless="${customer.paymentMethod.name() ==
    'CREDIT_CARD'}" class="warn">
33                              Payment must be done in the next 4 days
34                          </span>
35                      </td>
36                      <td th:class="${customer.balance gt 10000} ? 'enhanced'"
    th:text="${customer.balance}">350</td>
37                  </tr>
38                  <tr>
39                      <td>Mary</td>
40                      <td>Johanson</td>
41                      <td><img src="../../../images/female.png" /></td>
42                      <td><span>Credit card</span></td>
43                      <td>5000</td>
44                  </tr>
45                  <tr>
46                      <td>Robert</td>
47                      <td>Allen</td>
48                      <td><img src="../../../images/male.png" /></td>
49                      <td>
50                          <span>Credit card</span>
51                          <span class="warn">Payment must be done in the next 4 days</span>
52                      </td>
53                      <td class="enhanced">50000</td>
54                  </tr>
55              </tbody>
56          </table>
57      </body>
58  </html>
```

1) 25~28번째 라인

Translated by kunner, all rights are **not** reserved.

고객 성별은 남/여/불명 의 세 가지 조합으로 구성되어 있습니다. 이를 위해 th:switch – th:case를 사용합니다. 해당 구문의 사용 방법은 다음과 같습니다.

```
th:switch="${판별대상}"
  th:case="'판별값'"          <!-- // 조건 1 ~ ... //-->
  ...
  th:case="*"                 <!-- // 기본값 //-->
```

이때 각 case로 묶인 태그는 해당 case가 선택되지 않는 경우 실행 결과에서 제거됩니다.

따라서 본 예제의 성별 이미지는 판별된 값에 따라 하나만 출력되고 나머지는 출력되지 않습니다.

2) 32번째 라인

th:if 대신 th:unless를 사용해 if not() 을 처리할 수 있습니다. th:if 와 th:unless는 정확히 반대의 역할을 수행하며, 사용 방법은 동일합니다. 또, 어트리뷰트의 속성값에 ${not 조건식} 을 입력하면 반대의 결과를 얻을 수 있습니다.

3) 36번째 라인

어트리뷰트 속성값에 표현식을 직접 써 주는 방법입니다. Else에 Null을 반환하는 Elvis 연산자도 동일한 방법으로 사용합니다. 주어진 조건에 따라 선별적으로 class에 CSS class를 적용해야 하므로, th:class 어트리뷰트를 사용하였습니다.

사용 방법은 다음과 같습니다.

th:class="${조건절} ? ' TRUE반환값' : 'FALSE반환값' "

Exercise 10. SpEL

Thymeleaf에서 SpEL(Spring Expression Language)을 이용하는 방법에 대해 알아 보겠습니다.

사용법은 앞서 살펴본 것과 크게 다르지 않습니다. th:text 구문에 SpEL을 그대로 입력해 주면 됩니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 10</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
```

Translated by kunner, all rights are **not** reserved.

```
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 10: Spring Expression language</h1>
10          <h2>Arithmetic expressions</h2>
11          <p class="label">Four multiplied by minus six multiplied by minus two module seven:</p>
12          <p class="answer">123</p>
13           <h2>Object navigation</h2>
14          <p class="label">Description field of paymentMethod field of the third element of
    customerList bean:</p>
15          <p class="answer">Credit card</p>
16          <h2>Object instantiation</h2>
17          <p class="label">Current time milliseconds:</p>
18          <p class="answer">22-Jun-2013</p>
19          <h2>T operator</h2>
20          <p class="label">Random number:</p>
21          <p class="answer">123456</p>
22      </body>
23  </html>
```

 사이의 내용을 SpEL로 처리하는 예제 입니다.

1) Four multiplied by minus six multiplied by minus two module seven

   = ${4 * -6 -2 % 7}

2) Description field of paymentMethod field of the third element of customerList bean

   = ${customerList[2].paymentMethod.description}

3) Current time milliseconds

   = ${new java.util.Date().getTime()}

4) Random number

   = ${T(java.lang.Math).random()}

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 10</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 10: Spring Expression language</h1>
10          <h2>Arithmetic expressions</h2>
```

Translated by kunner, all rights are **not** reserved.

```
11          <p class="label">Four multiplied by minus six multiplied by minus two module seven:</p>
12          <p class="answer" th:text="${4 * -6 * -2 % 7}">123</p>
13          <h2>Object navigation</h2>
14          <p class="label">Description field of paymentMethod field of the third element of
    customerList bean:</p>
15          <p class="answer" th:text="${customerList[2].paymentMethod.description}">Credit
    card</p>
16          <h2>Object instantiation</h2>
17          <p class="label">Current time milliseconds:</p>
18          <p class="answer" th:text="${new java.util.Date().getTime()}">22-Jun-2013</p>
19          <h2>T operator</h2>
20          <p class="label">Random number:</p>
21          <p class="answer" th:text="${T(java.lang.Math).random()}">123456</p>
22      </body>
23  </html>
```

Exercise 11. 링크

매뉴얼 관련 정보: Using Thymeleaf 4-4장

th:href="@{ ... }" 구문을 이용하여 Thymeleaf에서 링크를 동적으로 생성할 수 있습니다.

@{ ... } 내부의 속성값에는 대체될 링크 정보를 입력합니다. 파라미터는 괄호로 감싸서 콤마(,)로 구분해 URI 뒤에 붙여 주면 됩니다. 링크 정보는 다른 구문과 마찬가지로 표현식을 이용하거나 객체 정보를 받아올 수 있으며, 다음과 같이 사용할 수 있습니다.

th:href="@{/order/orderList(execId=${execId}, orderId=${o.id}, execType='FAST')}"

※ 참고로 링크의 target은 th:target 으로 설정할 수 있습니다.

원본 HTML

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 11</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 11: links</h1>
10          <h2>Product actions</h2>
11          <ul>
12              <li><a>View product</a></li>
```

Translated by kunner, all rights are **not** reserved.

```
13                  <li><a>Edit product</a></li>
14              </ul>
15          </body>
16      </html>
```

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 11</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 11: links</h1>
10          <h2>Product actions</h2>
11          <ul>
12              <li><a href="#" th:href="@{/exercise11/product.html(action='view')}">View
    product</a></li>
13              <li><a href="#" th:href="@{/exercise11/product.html(action='edit')}">Edit
    product</a></li>
14          </ul>
15      </body>
16  </html>
```

Exercise 12. 폼 컨트롤

매뉴얼 관련 정보: Using Thymeleaf 4-3장

웹페이지에서 제공하는 입력폼은 보통 Bean과 강하게 결합된 형태로 개발됩니다. 각 입력 항목은 특정 오브젝트의 Bean 으로 get/set 됩니다. 기존 개발 작업에서는 폼에 이러한 Bean을 매핑시키느라 매번 별도의 코드를 추가해야 했습니다. 하지만 Thymeleaf의 폼 컨트롤 기능을 이용하면 쉽게 처리할 수 있습니다.

<form> 태그 내에 th:object 어트리뷰트를 이용해 오브젝트를 지정하고 th:action으로 입력 처리 페이지를 지정합니다. 또 input 항목에 각 Bean 요소들을 th:field로 매핑하면 나머지는 Thymeleaf이 알아서 처리합니다.

원본 HTML

```
1   <!DOCTYPE html>
```

Translated by kunner, all rights are **not** reserved.

```
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 12</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 12: forms</h1>
10         <h2>Customer edition</h2>
11         <!--
12             Set the action attribute to "/exercise12/saveCustomer.html", and use
13             th:object to bind the ${customer} variable as the form-backing bean.
14         -->
15         <form method="post" action="saveCustomer.html">
16             <!-- Use th:field to set field 'id' here -->
17             <input type="hidden" />
18             <!-- Use th:field to show *{firstName} -->
19             <label for="firstName">First name:</label>
20             <input type="text" />
21             <!-- Use th:field to show *{lastName} -->
22             <label for="lastName">Last name:</label>
23             <input type="text" />
24             Genre:
25             <!-- Iterate on ${genders} -->
26             <div class="radio">
27                 <!-- Use th:field to check *{gender} -->
28                 <input type="radio" />
29                 <label>Male</label>
30             </div>
31             <label for="paymentMethod">Payment method:</label>
32             <!-- Use th:field to select *{paymentMethod} -->
33             <select>
34                 <!-- Iterate on ${paymentMethods} -->
35                 <option>Credit card</option>
36             </select>
37             <!-- Use th:field to show *{balance} -->
38             <label for="balance">Balance (dollars):</label>
39             <input type="text" />
40             <input type="submit" />
41         </form>
42     </body>
43 </html>
```

Translated by kunner, all rights are **not** reserved.

주석의 지시에 따라 코드를 추가합니다.

<form>에 사용할 객체와 이동할 액션 페이지를 각각 th:object와 th:action으로 지정하고 각 입력 항목에 th:field를 지정하여 Bean과 입력폼을 연결합니다. <select>의 하위 <option>의 경우 th:each를 이용해 반복하고 th:value로 값을 지정합니다.

Thymeleaf에서 폼을 컨트롤하는 자세한 방법은 매뉴얼을 참고하시기 바랍니다.

적용

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 12</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Solution for exercise 12: forms</h1>
10         <h2>Customer edition</h2>
11         <form action="saveCustomer.html" th:action="@{/exercise12/saveCustomer.html}"
   th:object="${customer}" method="post">
12             <input type="hidden" th:field="*{id}" />
13             <label for="firstName">First name:</label>
14             <input type="text" th:field="*{firstName}" value="John" />
15             <label for="lastName">Last name:</label>
16             <input type="text" th:field="*{lastName}" value="Wayne" />
17             Genre:
18             <div th:each="gender : ${genders}" class="radio">
19                 <input type="radio" th:value="${gender}" th:field="*{gender}" />
20                 <label th:for="${#ids.prev('gender')}"
   th:text="${gender.description}">Male</label>
21             </div>
22             <div th:remove="all" class="radio">
23                 <input type="radio" />
24                 <label>Female</label>
25             </div>
26             <label for="paymentMethod">Payment method:</label>
27             <select th:field="*{paymentMethod}" th:remove="all-but-first">
28                 <option th:each="paymentMethod : ${paymentMethods}"
                           th:value="${paymentMethod}"
   th:text="${paymentMethod.description}">Credit card</option>
29                 <option>Another payment method</option>
30                 <option>Another payment method</option>
31             </select>
```

Translated by kunner, all rights are **not** reserved.

```
32              <label for="balance">Balance (dollars):</label>
33              <input type="text" th:field="*{balance}" size="10" value="2500" />
34              <input type="submit" />
35          </form>
36      </body>
37  </html>
```

각 입력폼에 지정된 기본값(value)는 설정된 th:field에 값이 존재하는 경우 해당 값으로 대체됩니다. (값이 없다면 기본값이 유지됩니다)

Exercise 13. Inlining

매뉴얼 관련 정보: Using Thymeleaf 12장

문자열 내의 특정 문자열을 대체하는 방법에 대해 알아 보겠습니다.

예를 들어 템플릿을 이용한 이메일 발송 시 회원 DB와 연동하여 고객 이름을 표시할 필요가 있습니다. 이때 Thymeleaf의 th:inline 어트리뷰트를 이용하면 별도의 메시지 처리기 필요 없이 바로 문자열을 변수로 대체할 수 있습니다.

다음 예제를 통해 textarea 내의 Peter 라는 문자열을 customerName 변수로 대체하여 출력하는 방법에 대해 알아 보겠습니다.

원본 HTML

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 13</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 13: inlining</h1>
10          <h2>Birthday email</h2>
11          <form action="#" method="post">
12              <label for="body">Message body:</label>
13  <textarea id="body" name="body">
14  Dear Peter,
15
16  it is our sincere pleasure to congratulate your in your birthday:
```

Translated by kunner, all rights are **not** reserved.

```
17      Happy birthday Peter!!!
18
19  See you soon, Peter.
20
21  Regards,
22      The Thymeleaf team
23  </textarea>
24              <input type="submit" value="Send mail" />
25          </form>
26      </body>
27  </html>
```

13 ~ 23번째 라인의 textarea 내에 삽입된 문자열에서 Peter 라는 이름이 세 번 나옵니다. (14, 17, 19)

textarea 태그에 th:inline 어트리뷰트를 지정합니다.

th:inline 어트리뷰트의 속성값은 "text", "javascript", "none"이 있습니다. 익히 예상할 수 있는 것처럼, text는 단순 문자열 대체, javascript 는 자바스크립트 내에서의 문자열 대체를 의미 합니다. 여기서는 text 속성값을 이용합니다.

대체될 부분에는 [[${표현식}]] 을 입력하면 됩니다.

'표현식' 에는 th:text가 지원하는 모든 구문의 사용이 가능합니다. 이는 th:inline의 속성값 종류와 관계 없이 모두 동일합니다.

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 13</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 13: inlining</h1>
10          <h2>Birthday email</h2>
11          <form action="#" method="post">
                <label for="body">Message body:</label>
12  <textarea id="body" name="body" th:inline="text">
13  Dear [[${customerName}]],
14
15  it is our sincere pleasure to congratulate your in your birthday:
16      Happy birthday [[${customerName}]]!!!
17
18  See you soon, [[${customerName}]].
```

Translated by kunner, all rights are **not** reserved.

```
19
20  Regards,
        The Thymeleaf team
21  </textarea>
22              <input type="submit" value="Send mail" />
23          </form>
24      </body>
25  </html>
```

위와 같이 th:inline 의 문자열 대체는 매우 편리하게 사용할 수 있습니다. 하지만 Thymeleaf의 핵심 사상인 "프로토타입을 정적인 페이지로 열었을 때 문제 없이 보여야 한다"는 원칙에 위배합니다. 위와 같이 변환된 코드를 브라우저에서 열어 보면 Peter 대신 [[${customerName}]] 으로 보일 것이기 때문입니다. 일반적으로는 큰 문제가 없지만 경우에 따라 프로토타입의 가치를 잃을 수도 있게 됩니다.

따라서 th:inline="text"을 사용할 때는 프로토타입의 완성도를 고려하여야 합니다.

이런 경우 th:inline="javascript"를 사용하는 방법에 대해 고려해야 합니다. Javascript를 사용하면 정적 브라우저에서 보여 줄 기본값의 지정이 가능합니다. 이에 대한 자세한 사용법은 매뉴얼 12-2장을 참고하시기 바랍니다.

Exercise 14. 동일 템플릿 내의 템플릿조각 사용하기

매뉴얼 관련 정보: Using Thymeleaf 8장

Thymeleaf v2.1에서는 템플릿 내의 특정 부분을 템플릿 조각으로 지정하는 기능을 지원합니다. 이렇게 템플릿조각으로 지정되면, 화면 내 어디서든 호출해 사용 가능합니다. 화면 내 동일한 요소가 중복으로 표시되어야 하는 경우 효과적으로 사용할 수 있습니다.

예를 들어, 모양은 동일하지만 파라미터나 CSS에 따라 표시 내용이나 형식이 달라지는 경우, 코드를 중복으로 작성할 필요 없이 템플릿조각으로 지정한 후 호출하면 됩니다. 이 기능은 '매개변수 지정 템플릿조각' 기능과 함께 사용하면 효과가 극대화 됩니다. (파라미터에 대한 사용법은 예제 15번에서 다루게 될 것입니다)

다만, 이 기능은 작성된 코드를 정적 페이지의 프로토타입으로 사용할 때는 적절하지 않습니다. 이런 경우에는 th:include 보다 th:replace가 적절합니다. 이와 관한 자세한 사용 방법에 대해서는 매뉴얼을 참고하시기 바랍니다.

다음 예제에서는 bannerElement를 id로 가진 DIV를 템플릿조각으로 지정한 후 지시에 따라 템플릿조각을 호출하는 방법을 살펴 보겠습니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
```

Translated by kunner, all rights are **not** reserved.

```
 3      <head>
 4          <title>Thymeleaf tutorial: exercise 14</title>
 5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
 6          <meta charset="utf-8" />
 7      </head>
 8      <body>
 9          <h1>Thymeleaf tutorial - Answer for exercise 14: same-template fragments</h1>
10          <div id="bannerElement">
11              <div class="banner">
12                  <img alt="Thymeleaf logo" />
13                  <span>The Thymeleaf tutorial</span>
14              </div>
15          </div>
16          <h2>Product information</h2>
17          <dl>
18              <dt>Product name</dt>
19              <dd th:text="${product.description}">Red chair</dd>
20              <dt>Product price</dt>
21              <dd th:text="${product.price}">350</dd>
22              <dt>Product available from</dt>
23              <dd th:text="${product.availableFrom}">28-Jun-2013</dd>
24          </dl>
25          <!-- Include the banner here -->
26          <h2>Customer information</h2>
27          <dl>
28              <dt>First name</dt>
29              <dd th:text="${customer.firstName}">John</dd>
30              <dt>Last name</dt>
31              <dd th:text="${customer.lastName}">Smith</dd>
32              <dt>Genre</dt>
33              <dd th:text="${customer.gender?.description}">Male</dd>
34              <dt>Payment method</dt>
35              <dd th:text="${customer.paymentMethod?.description}">Credit card</dd>
36              <dt>Balance</dt>
37              <dd th:text="'$' + ${customer.balance}">$350</dd>
38          </dl>
39          <!-- Include the banner here -->
40      </body>
41  </html>
```

예제에서 10~15번째 라인의 DIV 요소가 템플릿조각으로 지정될 부분입니다. (id="bannerElement")

템플릿에서 특정 부분을 템플릿조각으로 지정할 때는 th:fragment 어트리뷰트를 이용합니다. 그리고 지정된 템플릿조각을 호출할 때는 th:include 어트리뷰트를 이용합니다.

Translated by kunner, all rights are **not** reserved.

일반적으로 th:include 어트리뷰트를 사용하는 방법은 다음과 같습니다.

하지만 본 예제에서는 동일 템플릿 내의 템플릿조각을 호출해야 하므로 다음과 같이 입력해야 합니다.

파일명 대신 this를 입력하고 구분자 :: 뒤에 [//div[@id='조각명']] 을 붙여 준다는 것에 유의하시기 바랍니다.

th:include 어트리뷰트의 속성값에 문자열 대신 표현식이나 조건문을 쓸 수 있습니다. 또 th:include에 th:with 어트리뷰트를 조합하여 파라미터를 전달할 수도 있습니다.

<div th:include="this :: [//div[@id='ID']]" th:with="onevar=${value1}, twovar=${value2}" />

적용

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 14</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Solution for exercise 14: same-template fragments</h1>
10         <div th:fragment="banner" id="bannerElement">
11             <div class="banner">
12                 <img src="../../../images/logo.png" th:src="@{/images/logo.png}"
   alt="Thymeleaf logo" />
13                 <span>The Thymeleaf tutorial</span>
14             </div>
15         </div>
16         <h2>Product information</h2>
17         <dl>
18             <dt>Product name</dt>
19             <dd th:text="${product.description}">Red chair</dd>
20             <dt>Product price</dt>
21             <dd th:text="${product.price}">350</dd>
22             <dt>Product available from</dt>
23             <dd th:text="${product.availableFrom}">28-Jun-2013</dd>
24         </dl>
25         <div th:include="this :: banner">Banner</div>
26         <h2>Customer information</h2>
27         <dl>
28             <dt>First name</dt>
29             <dd th:text="${customer.firstName}">John</dd>
```

Translated by kunner, all rights are **not** reserved.

```
30                <dt>Last name</dt>
31                <dd th:text="${customer.lastName}">Smith</dd>
32                <dt>Genre</dt>
33                <dd th:text="${customer.gender?.description}">Male</dd>
34                <dt>Payment method</dt>
35                <dd th:text="${customer.paymentMethod?.description}">Credit card</dd>
36                <dt>Balance</dt>
37                <dd th:text="'$' + ${customer.balance}">$350</dd>
38            </dl>
39            <div th:include="this :: [//div[@id='bannerElement']]">Banner</div>
40        </body>
41    </html>
```

앞서 언급한 바와 같이 템플릿조각은 그 자체로는 그다지 대단하지 않습니다. 무엇보다 단지 정적인 코드를 조각화 하는 것은 그다지 효율적이지 않습니다. 그러나 몇 가지 기능과 함께 결합하면 템플릿조각이야말로 Thymeleaf 템플릿엔진이 제공하는 가장 파워풀한 기능입니다. 다음 예제를 통해 템플릿조각의 활용성을 크게 높이는 방법에 대해 학습해 보겠습니다.

Exercise 15. 파라미터와 함께 템플릿조각 사용하기

매뉴얼 관련 정보: Using Thymeleaf 8장

앞선 예제에서 잠깐 언급했던 템플릿조각에 파라미터를 넘겨 사용하는 방법에 대해 알아 보겠습니다. 쉽게 예상할 수 있는 것처럼, 파라미터에 따라 템플릿조각의 실행 결과가 달라지게 될 것입니다.

이에 대해서는 앞서 설명한 만큼 별다른 설명이 필요없을 테니 바로 예제로 넘어 가겠습니다.

본 예제에서는 동일한 템플릿조각을 세 번에 걸쳐 호출하는데, 파라미터에 따라 서로 다른 CSS 클래스를 호출해 템플릿 처리 결과 3가지 다른 배너가 페이지에 표시되도록 할 것입니다.

예제의 지시사항은 다음과 같습니다.

1) 1번 배너 파라미터: "white" , "Product information"

2) 2번 배너 파라미터: "red" , "Customer information"

3) 3번 배너 파라미터: "blue" , "Copyright 2013"

각 파라미터는 순서대로 적용할 css클래스명과 대체할 문자열을 의미합니다.

원본 HTML

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 15</title>
```

Translated by kunner, all rights are **not** reserved.

```
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 15: parameterizable fragments</h1>
10          <div th:fragment="banner" th:remove="all">
11              <div class="banner white">
12                  <img src="../../../images/logo.png" th:src="@{/images/logo.png}" alt="Thymeleaf
    logo" />
13                  <span>The Thymeleaf tutorial</span>
14              </div>
15          </div>
16          <div th:include="this :: banner">Banner</div>
17          <h2>Product information</h2>
18          <dl>
19              <dt>Product name</dt>
20              <dd th:text="${product.description}">Red chair</dd>
21              <dt>Product price</dt>
22              <dd th:text="${product.price}">350</dd>
23              <dt>Product available from</dt>
24              <dd th:text="${product.availableFrom}">28-Jun-2013</dd>
25          </dl>
26          <div th:include="this :: banner">Banner</div>
27          <h2>Customer information</h2>
28          <dl>
29              <dt>First name</dt>
30              <dd th:text="${customer.firstName}">John</dd>
31              <dt>Last name</dt>
32              <dd th:text="${customer.lastName}">Smith</dd>
33              <dt>Genre</dt>
34              <dd th:text="${customer.gender?.description}">Male</dd>
35              <dt>Payment method</dt>
36              <dd th:text="${customer.paymentMethod?.description}">Credit card</dd>
37              <dt>Balance</dt>
38              <dd th:text="'$' + ${customer.balance}">$350</dd>
39          </dl>
40          <div th:include="this :: banner">Banner</div>
41      </body>
42  </html>
```

10~15번째 라인을 보면, 템플릿조각을 지정하는 구문이 작성되어 있습니다. (앞서 살펴 봤던 동일 템플릿 내의 템플릿조각 지정입니다) 그리고 11번과 13번째 라인의 코드가 각각 th:include에서 넘겨 받은 파라미터와 매핑해야 할 부분입니다. – *위에서 이야기 한 것처럼 첫번째 파라미터는 css클래스명, 두번째*

Translated by kunner, all rights are **not** reserved.

파라미터는 대체할 문자열입니다.

적용

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 15</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Solution for exercise 15: parameterizable fragments</h1>
10         <div th:fragment="banner(color, text)" th:remove="all">
11             <div class="banner " th:attrappend="class=${color}">
12                 <img src="../../../images/logo.png" th:src="@{/images/logo.png}"
   alt="Thymeleaf logo" />
13                 <span th:text="${text}">The Thymeleaf tutorial</span>
14             </div>
15         </div>
16         <div th:include="this :: banner('white', 'Product information')">Banner</div>
17         <h2>Product information</h2>
18         <dl>
19             <dt>Product name</dt>
20             <dd th:text="${product.description}">Red chair</dd>
21             <dt>Product price</dt>
22             <dd th:text="${product.price}">350</dd>
23             <dt>Product available from</dt>
24             <dd th:text="${product.availableFrom}">28-Jun-2013</dd>
25         </dl>
26         <div th:include="this :: banner('red', 'Customer information')">Banner</div>
27         <h2>Customer information</h2>
28         <dl>
29             <dt>First name</dt>
30             <dd th:text="${customer.firstName}">John</dd>
31             <dt>Last name</dt>
32             <dd th:text="${customer.lastName}">Smith</dd>
33             <dt>Genre</dt>
34             <dd th:text="${customer.gender?.description}">Male</dd>
35             <dt>Payment method</dt>
36             <dd th:text="${customer.paymentMethod?.description}">Credit card</dd>
37             <dt>Balance</dt>
38             <dd th:text="'$' + ${customer.balance}">$350</dd>
39         </dl>
```

Translated by kunner, all rights are **not** reserved.

```
40          <div th:include="this :: banner('blue', 'Copyright 2013')">Banner</div>
41      </body>
    </html>
```

10번째 라인의 th:fragment 에서 조각명에 괄호를 묶어 넘겨 받을 인수를 지정합니다.

11번째 라인은 css클래스를 넘겨 받은 파라미터에 따라 지정하는 코드 입니다. th:attrappend 어트리뷰트는 속성값의 내용을 별도의 어트리뷰트로 지정합니다. 여기서는 넘겨 받은 파라미터를 바탕으로 class="css클래스명"라는 코드를 생성하게 될 것이라는 것을 유추할 수 있습니다.

16, 26, 40번째 라인은 이렇게 지정한 조각을 호출해 화면에 적용하는 코드 입니다. 앞선 14번 예제에서 학습한 내용이니 어렵지 않게 이해할 수 있을 것입니다.

특정 코드를 템플릿조각으로 지정해 호출하는 것은 비단 Thymeleaf 뿐 아니라 대다수의 템플릿엔진에서 지원하는 기능입니다. 사실 템플릿엔진을 사용하는 가장 큰 이유이기도 합니다. 심지어는 정적 JSP 에서조차 <jsp:include /> 구문을 이용해 템플릿을 재활용 합니다.

그러나 Thymeleaf는 템플릿팅을 할 때 예제와 같이 동일 템플릿에서 특정 영역을 조각화 한다든가, 템플릿조각에 파라미터를 넘긴다든가 하는 방식으로 다른 템플릿엔진이나 jsp include에 비해 훨씬 더 강력한 기능을 지원합니다. 그리고 이 조각을 html이나 jsp 같은 물리 파일이 아니라 영속화 하고(i.e. DBMS, XML 등), UI에서 이를 불러들여 사용하는 방식으로 서비스 하면 별도의 CMS 엔진 없이도 콘텐츠를 동적으로 관리할 수 있게 됩니다.

Thymeleaf를 이용한 동적 UI 관리에 대해 좀 더 설명하면 다음과 같습니다. 템플릿조각 내에 들어갈 요소를 관리자도구를 통해 입력한 후 이를 DB에 저장하여 버전 관리를 하고, 최종 버전을 템플릿의 캐시에 저장하여 서비스 하는 구조를 갖추게 되면 서비스 내 모든 페이지를 동적으로 관리할 수 있게 됩니다. 템플릿을 캐시에 저장하는 것은 Thymeleaf의 기본 구조이며, 한 서비스 내에서 템플릿의 크기는 충분히 예상 가능하므로 성능(H/W, S/W)에 전혀 지장이 없을 것입니다.

Exercise 16. 리터럴 대체

매뉴얼 관련 정보: Using Thymeleaf 4.6, 4.7, 4.12장

이번 예제에서는 문자열을 결합하거나 대체하는 다양한 방법을 학습합니다. Thymeleaf에서는 문자열과 관련해 다음 세 가지 방법으로 처리할 수 있습니다.

1	'문자열1' + '문자열2'	가장 기본적인 문자열 결합 방법 – 매뉴얼 4.6장 참조 자바스크립트에서 문자열을 결합하는 방법과 동일합니다.
2	'문자열1 __문자열2__'	전처리를 이용한 문자열 결합 방법 – 매뉴얼 4.12장 참조 더블 언더스코어(__)로 둘러싸인 부분을 먼저 처리하고 그 결과를 출력합니다. 보통 message.properties와 함께 사용합니다.
3	|문자열1 문자열2|	리터럴 대체를 이용한 문자열 결합 방법 – 매뉴얼 4.7장 참조 |로 둘러싸인 부분에 홑따옴표(') 없이 문자열을 나열합니다.

Translated by kunner, all rights are **not** reserved.

		리터럴 대체 안의 문자열에는 반드시 ${}을 이용한 스트링 변수만 사용 가능합니다. 불린, 숫자, 조건식 등의 다른 리터럴은 사용할 수 없습니다.

예제에서는 위 세 가지 방법으로 템플릿이 Hello, Peter를 출력할 수 있게 변경하도록 지시하고 있습니다. 자세한 내용은 아래 예제를 통해 확인하시기 바랍니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 16</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 16: literal substitutions</h1>
10         <h2>Concatenation</h2>
11         <span>Hello, Peter!</span>
12         <h2>Preprocessing</h2>
13         <span>Hello, Peter!</span>
14         <h2>Literal substitution</h2>
15         <span>Hello, Peter!</span>
16     </body>
17 </html>
```

적용

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 16</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Solution for exercise 16: literal substitutions</h1>
10         <h2>Concatenation</h2>
11         <span th:text="'Hello ' + ${customerName} + '!'">Hello, Peter!</span>
12         <h2>Preprocessing</h2>
           <span th:text="'Hello __${customerName}__!'">Hello, Peter!</span>
13         <h2>Literal substitution</h2>
14         <span th:text="|Hello ${customerName}!|">Hello, Peter!</span>
```

Translated by kunner, all rights are **not** reserved.

```
15      </body>
16  </html>
```

Exercise 17. 주석

매뉴얼 관련 정보: Using Thymeleaf 11장

Thymeleaf 에서 템플릿을 작성할 때 사용할 수 있는 주석 처리 구문을 학습합니다.

Thymeleaf는 아직 프로그래밍되지 않은 프로토타입과 프로그램을 입힌 후의 서버 실행 결과가 동일하게 보여 지는 것을 핵심 사상으로 표방하고 있습니다. 그런데 실제 작업을 하다보면 간혹 어떤 것은 반드시 서버의 실행 결과에 따라서만 출력될 수 있기 때문에 정적 HTML에서 이를 처리하기 위해 일일히 코딩하면 프로그램을 입힐 때 해당 코드를 제거해야 하는 문제가 생깁니다. 물론 앞서 학습한 th:remove 구문을 이용하는 방법도 있지만, 특정 블록에 대해 처리하기 위해서는 다른 방법을 이용해야 합니다.

이를 위해 Thymeleaf에서는 다음과 같은 독특한 주석 처리를 지원합니다.

#	종류	사용 방법	프로토타입	서버 실행 결과
1	HTML Comment	<!-- 내용 -->	주석 처리	주석 처리
2	Thymeleaf Comment	<!--/* 내용 */-->	주석 처리	제거
3	Generated-only Comment	<!--/*/ 내용 /*/-->	주석 처리	내용만 출력
4	Prototype-only Comment	<!--/*--> 내용 <!--*/-->	출력	제거

각 주석의 자세한 사용방법은 매뉴얼을 참고하시기 바랍니다.

이번 예제에서는 원본 HTML의 [1] ~ [4]를 지시에 따라 위 네 가지 주석 처리 방법으로 변환해 보도록 하겠습니다.

원본 HTML

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 17</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 17: comments</h1>
10          [1] Start of the HTML main content
11          <h2>Product information</h2>
12          <dl>
13              [2] The bean product is set in the controller
14              <dt>Product name</dt>
```

Translated by kunner, all rights are **not** reserved.

```
15                <dd th:text="${product.description}">Red chair</dd>
16
17                <dt>[3] Product price</dt>
18                <dd th:text="${product.price}">350</dd>
19
20                <dt>Product available from</dt>
21                <dd th:text="${product.availableFrom}">28-Jun-2013</dd>
22            </dl>
23            <footer>[4] Development mode!</footer>
24        </body>
25    </html>
```

[1] 은 단순 HTML 주석이므로 해당 행을 <!-- --> 로 감싸 주면 됩니다.

[2] 는 프로토타입으로 볼 때는 주석으로 남고, 서버 실행 시에는 제거되는 Thymeleaf Comment를 사용합니다.

[3] 은 해당 행 뿐 아니라 부속 <dd> 태그까지 한번에 주석 처리 합니다. 정적 HTML에서는 해당 내용 전체가 보이지 않아야 하므로 Generated-only Comment를 사용합니다.

[4] 는 정적 HTML에서만 보이고, 서버 실행 시에는 아예 제거되는 부분입니다. <footer> 태그 전체를 Prototype-only Comment로 감싸줍니다.

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 17</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 17: comments</h1>
10          <!-- [1] Start of the HTML main content -->
11          <h2>Product information</h2>
12          <dl>
                <!--/* [2] The bean product is set in the controller */-->
13              <dt>Product name</dt>
14              <dd th:text="${product.description}">Red chair</dd>
15
16              <!--/*/
17              <dt>[3] Product price</dt>
18              <dd th:text="${product.price}">350</dd>
19              /*/-->
```

Translated by kunner, all rights are **not** reserved.

```
20
21              <dt>Product available from</dt>
22              <dd th:text="${product.availableFrom}">28-Jun-2013</dd>
23          </dl>
24          <!--/*-->
25              <footer>[4] Development mode!</footer>
26          <!--*/-->
27      </body>
28  </html>
```

Thymeleaf의 주석 처리 기능을 이용하면 디자이너/퍼블리셔와 프로그래머의 협업 시 큰 도움이 됩니다. 하지만 누군가 주석의 일반적인 의미에 따라 '지워도 좋다' 라는 생각을 하게 되면, 작업에 큰 혼란을 줄 수 있습니다. 따라서 이 기능은 매우 유용하지만 현업에서 활용할 때는 반드시 충분한 학습이 필요합니다.

Exercise 18. data-* 구문

매뉴얼 관련 정보: Using Thymeleaf 5-6장

W3C는 Custom Element를 지정할 때 prefix – name의 형태로 명명하라고 권고 합니다. 이에 따라 Thymeleaf의 th:* 어트리뷰트 대신 data-th-* 로 표기할 수 있습니다. th:text 대신 data-th-text 를 쓰는 식입니다. 이렇게 하면 어트리뷰트 명에 콜론(:)이 붙지 않기 때문에 W3C 권고를 위배하지 않습니다. 하지만 th:* 구문을 data-th-*로 대체하겠다는 것은 아니라고, Thymeleaf는 밝히고 있습니다.

원본 HTML

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 18</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Answer for exercise 18: data-* syntax</h1>
10          <h2>Product list</h2>
11          <table>
12              <thead>
13                  <tr>
14                      <th>Description</th>
15                      <th>Price</th>
16                      <th>Available from</th>
```

Translated by kunner, all rights are **not** reserved.

```
17                      <th></th>
18                  </tr>
19              </thead>
20              <tbody>
21                  <tr th:each="product : ${productList}">
22                      <td th:text="${product.description}">Red chair</td>
23                      <td th:text="${'$' + #numbers.formatDecimal(product.price, 1, 2)}">$350</td>
24                      <td th:text="${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}">28-Jun-
25 2013</td>
26                      <td>
27                          <span th:if="${product.price lt 100}" class="offer">Special offer!</span>
28                      </td>
29                  </tr>
30              </tbody>
31          </table>
32      </body>
33 </html>
```

모든 th:* 구문을 data-th-*로 바꿔 줍니다.

적용

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 18</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Solution for exercise 18: data-* syntax</h1>
10         <h2>Product list</h2>
11         <table>
12             <thead>
               <tr>
13                 <th>Description</th>
14                 <th>Price</th>
15                 <th>Available from</th>
16                 <th></th>
17             </tr>
18         </thead>
19         <tbody>
20             <tr data-th-each="product : ${productList}">
```

Translated by kunner, all rights are **not** reserved.

```
21                      <td data-th-text="${product.description}">Red chair</td>
22                      <td data-th-text="${'$' + #numbers.formatDecimal(product.price, 1,
    2)}">$350</td>
23                      <td data-th-text="${#dates.format(product.availableFrom, 'dd-MMM-
    yyyy')}">28-Jun-2013</td>
24                      <td>
25                          <span data-th-if="${product.price lt 100}" class="offer">Special
    offer!</span>
26                      </td>
27                  </tr>
28              </tbody>
29          </table>
30      </body>
31  </html>
```

Exercise 19. 조건에 따른 th:remove

Thymeleaf 2.1 신규 기능

th:remove 어트리뷰트는 해당 태그(또는 태그 바디 포함)를 제거할 때 사용합니다. 그런데 th:remove 어트리뷰트의 속성값에 표현식으로 조건절을 입력하면 조건에 따라 th:remove가 작동합니다.

Link text not to be removed

여기서 th:remove의 평가식 결과가 true 일때 tag 가 입력되어 있는 것을 주목하시기 바랍니다. 이 경우 해당 태그가 유지 된다는 뜻입니다. 또, a 태그의 경우 table이나 div와는 달리, 태그 바디나 하위 태그는 그대로 두고 태그만 지워지는 것에 유의하시기 바랍니다.

다음은 Customer 객체(Appendix 3)를 이용해 화면에 고객 목록을 출력하는 예제 입니다.

고객의 웹사이트가 null이 아닌 경우 고객의 First Name, Last Name에 웹사이트 링크를 걸고, null 이면 그냥 이름만 출력하게 하도록 코드를 변경하겠습니다.

원본 HTML

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 19</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Answer for exercise 19: conditional th:remove</h1>
```

Translated by kunner, all rights are **not** reserved.

```
10          <h2>Customer list</h2>
11          <table>
12              <thead>
13                  <tr>
14                      <th>First name</th>
15                      <th>Last name</th>
16                      <th>Balance</th>
17                  </tr>
18              </thead>
19              <tbody th:remove="all-but-first">
20                  <tr th:each="customer : ${customerList}">
21                      <td th:text="${customer.firstName}">Peter</td>
22                      <td th:text="${customer.lastName}">Jackson</td>
23                      <td th:text="${customer.balance}">350</td>
24                  </tr>
25                  <tr>
26                      <td>
27                          <a href="http://mary.blogspot.com/">Mary</a>
28                      </td>
29                      <td>
30                          <a href="http://mary.blogspot.com/">Johanson</a>
31                      </td>
32                      <td>5000</td>
33                  </tr>
34                  <tr>
35                      <td>Robert</td>
36                      <td>Allen</td>
37                      <td>50000</td>
38                  </tr>
39              </tbody>
40          </table>
41      </body>
42  </html>
```

지시에 따라 고객명에 적용된 <a> 태그 내에 th:remove=" ${customer.personalWebsite == null} ? tag" 를 입력합니다.

적용

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 19</title>
```

Translated by kunner, all rights are **not** reserved.

```
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 19: conditional th:remove</h1>
10          <h2>Customer list</h2>
11          <table>
12              <thead>
                    <tr>
13                      <th>First name</th>
14                      <th>Last name</th>
15                      <th>Balance</th>
16                  </tr>
17              </thead>
18              <tbody th:remove="all-but-first">
19                  <tr th:each="customer : ${customerList}">
20                      <td>
21                          <a th:href="${customer.personalWebsite}"
22                             th:remove="${customer.personalWebsite == null} ? tag"
23                             th:text="${customer.firstName}">Peter</a>
24                      </td>
25                      <td>
26                          <a th:href="${customer.personalWebsite}"
27                             th:remove="${customer.personalWebsite == null} ? tag"
28                             th:text="${customer.lastName}">Jackson</a>
29                      </td>
30                      <td th:text="${customer.balance}">350</td>
31                  </tr>
32                  <tr>
33                      <td>
34                          <a href="http://mary.blogspot.com/">Mary</a>
35                      </td>
36                      <td>
37                          <a href="http://mary.blogspot.com/">Johanson</a>
38                      </td>
39                      <td>5000</td>
40                  </tr>
41                  <tr>
42                      <td>Robert</td>
43                      <td>Allen</td>
44                      <td>50000</td>
45                  </tr>
46              </tbody>
47          </table>
```

Translated by kunner, all rights are **not** reserved.

```
48      </body>
49  </html>
```

Exercise 20. Conversion Service

Thymeleaf 2.1 신규 기능

Spring에서 DAO를 주입 받은 빈을 Service로 구현하여 사용자 정의 데이터타입을 정의할 수 있습니다. Thymeleaf는 이 Spring의 Conversion Service와 통합하여 사용자 정의 데이터타입을 템플릿에서 쉽게 사용할 수 있게 합니다. 즉, 출력해야 하는 특정 빈값에 Conversion Service가 지정되어 있는 경우, Thymeleaf는 자동으로 해당 Conversion Service를 실행한 결과를 화면에 출력합니다.

기본적인 데이터 출력에는 ${...} 구문을 사용하고, Conversion Service와 통합할 때는 ${{ ... }} 를 사용하면 됩니다.

원본 HTML

```
1   <!DOCTYPE html>
2   <html xmlns:th="http://www.thymeleaf.org">
3       <head>
4           <title>Thymeleaf tutorial: exercise 20</title>
5           <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6           <meta charset="utf-8" />
7       </head>
8       <body>
9           <h1>Thymeleaf tutorial - Solution for exercise 20: conversion service</h1>
10          <h2>Product information</h2>
11          <dl>
12              <dt>Product price:</dt>
13              <dd th:text="${price}">$150</dd>
14              <dt>Product release date:</dt>
15              <dd th:text="${releaseDate}">23-Jan-2014</dd>
16          </dl>
17      </body>
18  </html>
```

여기서 13, 15번째 라인을 출력할 때, ${ ... } 대신 ${{ ... } }을 사용합니다.

물론 이에 앞서 반드시 Spring에 해당 필드에 대한 포맷팅이 설정된 Conversion Service가 등록되어 있어야 합니다.

적용

Translated by kunner, all rights are **not** reserved.

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>Thymeleaf tutorial: exercise 20</title>
5          <link rel="stylesheet" href="../../../css/main-static.css" th:href="@{/css/main.css}" />
6          <meta charset="utf-8" />
7      </head>
8      <body>
9          <h1>Thymeleaf tutorial - Solution for exercise 20: conversion service</h1>
10         <h2>Product information</h2>
11         <dl>
12             <dt>Product price:</dt>
13             <dd th:text="${{price}}">$150</dd>
14             <dt>Product release date:</dt>
15             <dd th:text="${{releaseDate}}">23-Jan-2014</dd>
16         </dl>
17     </body>
18 </html>
```

위 구문을 서버에서 실행하면, 각각 price와 releaseDate에 정의된 사용자 정의 데이터타입에 따라 포맷팅된 결과가 화면에 출력될 것입니다. 이 구문이 유용한 이유는 동일한 데이터를 템플릿 작성 내용에 따라 서로 다른 표기로 보여 줄 수 있기 때문입니다. 이는 Thymeleaf을 포함한 모든 템플릿엔진들의 지향인 데이터와 표현의 분리를 위해 중요합니다.

Translated by kunner, all rights are **not** reserved.

Appendix

1. Product.java

```
package org.thymeleaf.itutorial.beans;

import java.util.Date;

public class Product {

    private String description;
    private Integer price;
    private Date availableFrom;

    public Product(final String description, final Integer price, final Date availableFrom) {
        this.description = description;
        this.price = price;
        this.availableFrom = availableFrom;
    }

    public Date getAvailableFrom() {
        return this.availableFrom;
    }

    public void setAvailableFrom(final Date availableFrom) {
        this.availableFrom = availableFrom;
    }

    public String getDescription() {
        return this.description;
    }

    public void setDescription(final String description) {
        this.description = description;
    }

    public Integer getPrice() {
        return this.price;
    }

    public void setPrice(final Integer price) {
        this.price = price;
    }
```

Translated by kunner, all rights are **not** reserved.

```
}
```

2. Internationalization – message.properties

2.1. messages_en.properties: English Text

```
tutorial.exercise4=Thymeleaf tutorial - exercise 4: internationalization
product.info=Product information
product.name=Product name
product.price=Product price
product.available=Product available from
back=Back
```

2.2. messages_es.properties: Spanish Text

```
tutorial.exercise4=Tutorial de Thymeleaf - ejercicio 4: internacionalizaci₩u00f3n
product.info=Informaci₩u00f3n del producto
product.name=Nombre del producto
product.price=Precio del producto
product.available=Producto disponible desde
back=Volver
```

2.3. messages_fr.properties: French Text

```
tutorial.exercise4=Tutorial De Thymeleaf - exercice 4: l'internationalisation
product.info=Information du produit
product.name=Nom du produit
product.price=Prix du produit
product.available=Produit disponible depuis
back=Revenir
```

3. Customer.java

```
package org.thymeleaf.itutorial.beans;

/**
 * Customer information.
 */
public class Customer {
```

Translated by kunner, all rights are **not** reserved.

```
private Integer id;
private String firstName;
private String lastName;
private Gender gender;
private PaymentMethod paymentMethod;
private int balance;
private String personalWebsite;

public Customer() {
    super();
}

public Customer(
        final Integer id, final String firstName, final String lastName,
        final Gender gender, final PaymentMethod paymentMethod,
        final int balance, final String personalWebsite) {
    super();
    this.id = id;
    this.firstName = firstName;
    this.lastName = lastName;
    this.gender = gender;
    this.paymentMethod = paymentMethod;
    this.balance = balance;
    this.personalWebsite = personalWebsite;
}

public Integer getId() {
    return this.id;
}

public void setId(final Integer id) {
    this.id = id;
}

public int getBalance() {
    return this.balance;
}

public void setBalance(final int balance) {
    this.balance = balance;
}

public String getFirstName() {
    return this.firstName;
```

Translated by kunner, all rights are **not** reserved.

```
    }

    public void setFirstName(final String firstName) {
        this.firstName = firstName;
    }

    public Gender getGender() {
        return this.gender;
    }

    public void setGender(final Gender gender) {
        this.gender = gender;
    }

    public String getLastName() {
        return this.lastName;
    }

    public void setLastName(final String lastName) {
        this.lastName = lastName;
    }

    public PaymentMethod getPaymentMethod() {
        return this.paymentMethod;
    }

    public void setPaymentMethod(final PaymentMethod paymentMethod) {
        this.paymentMethod = paymentMethod;
    }

    public String getPersonalWebsite() {
        return personalWebsite;
    }

    public void setPersonalWebsite(String personalWebsite) {
        this.personalWebsite = personalWebsite;
    }
}
```

4. Gender.java

```
package org.thymeleaf.itutorial.beans;
```

Translated by kunner, all rights are **not** reserved.

```
public enum Gender {

    FEMALE("Female gender"),
    MALE("Male gender");

    private String description;

    Gender(final String description) {
        this.description = description;
    }

    public String getDescription() {
        return this.description;
    }
}
```

5. PaymentMethod.java

```
package org.thymeleaf.itutorial.beans;

public enum PaymentMethod {

    CREDIT_CARD("Credit card payment"),
    DIRECT_DEBIT("Direct debit payment"),
    BANK_TRANSFER("Bank transfer payment");

    private String description;

    PaymentMethod(final String description) {
        this.description = description;
    }

    public String getDescription() {
        return this.description;
    }

}
```

Translated by kunner, all rights are **not** reserved.