bean values
context object: product
```
<td data-th-text="${product.description}">Red chair</td>
<td th:text="${product.description}">Red chair</td>
```

formatting
```
“${#numbers.formatDecimal(product.price, 1, 2)}”
“${#dates.format(product.availableFrom, 'dd-MMM-yyyy')}”
```

string concatenation
```
“${'string1' + product.price}”
“|string1 + ${product.price}|”
```

inlining
In order for inlining to work, we must activate it by using the th:inline attribute, which has three possible values or modes (*text*, *javascript* and *none*).
```
<p>Price [[${product.price}]]</p>
```

escaped and unescaped text
```
th:text="${html}"
<div>This is an &lt;em&gt;HTML&lt;/em&gt; text.
th:utext="${html}"
<div>This is an <em>HTML</em> text.
```

i18n
```
"#{product.name}"
```

iteration
```
<tr th:each="product : ${productList}">
<tr th:each="prod,iterStat : ${prods}">
        index, count, size, current, even, odd,first, last
```

conditions
```
th:if="${product.name?: ‘default name’}"
// outputs name of enum if enumType is not null
th:text="${product.enumType?.name() }"
th:if="${product.price lt 100}"
th:unless="${product.price lt 100}"
th:remove="all-but-first"
        all – tag and its children
        body – only children
        tag – only tag
        all-but-first – children except first
        none – do nothing
```

Spring expression language
Description field of paymentMethod field of the third element of customerList bean
```
th:text="${customerList[2].paymentMethod.description}"
```

links
```
thymelaef:
        th:href="@{/exercise11/product.html(action='view')}"
html output:    href="/exercise11/product.html?action=view"
```

template fragments
define:
```
<div th:fragment="banner" id="bannerElement"> … </div>
```
use:
```
<div th:include="this :: banner" />
```
using xpath:
```
<div th:include="this :: [//div[@id='bannerElement']]" />
```

parameterizable fragments
define:
```
<div th:fragment="banner(color, text)" >
<div th:attrappend="class=${color}"> … </div>
<span th:text="${text}">The Thymeleaf tutorial</span> </div>
```
use:
```
<div th:include="this :: banner('blue', 'Some text')" />
```

forms
```
<form action="saveCustomer.html"
      th:action="@{/exercise12/saveCustomer.html}"
      th:object="${customer}"   method="post">

       <input type="hidden" th:field="*{id}" />
       <input type="text" th:field="*{firstName}"
 th:class="${#fields.hasErrors('age')}? 'error'" />

       <div th:each="gender : ${genders}" >
          <input type="radio"
                 th:value="${gender}"
                 th:field="*{gender}" />
          <label th:for="${#ids.prev('gender')}"
th:text="${gender.description}">Male</label>
       </div>

      <select th:field="*{paymentMethod}" >
          <option th:each="paymentMethod : ${paymentMethods}"
                 th:value="${paymentMethod}"
                 th:text="${paymentMethod.description}">Credit
           card</option>
      </select>

      <input type="submit" />
</form>
```

comments
```
<!--/* static: comment, runtime: removed */-->
<!--/* --> static: comment, runtime: removed <!--*/-->

<!--/*/  <div th:text="${...}"> static: comment, runtime: executed
</div> /*/-->

<!--/*/ <th:block th:each="user : ${users}"> /*/-->
   <td th:text="${user.login}">...</td>
<!--/*/ </th:block> /*/-->
```

conversion service
use: "${{releaseDate}}"
Conversion handler:
http://www.thymeleaf.org/whatsnew21.html#spconv

template uri variables
```
<a th:href="@{/buy/{id}/{name}.html(id=${productId},
name=${productName})}">Buy now!</a>
Output: <a href="/buy/273/Red-carpet.html">Buy now!</a>
```

preprocessing
And what is that preprocessing thing? It is an execution of the expressions done before the normal one, that allows the modification of the actual expression that will be eventually executed. Preprocessed expressions are exactly like normal ones, but appear surrounded by a double underscore symbol (like __${expression}__).
Ex.:
```
<p th:text="${__#{article.text('textVar')}__}">Some text here...</p>
<span th:text="'Hello __${customerName}__!'">Hello, Peter!</span>
```