Thymeleaf Commands

Thymeleaf is a template engine, and Thymeleaf templates make up the views of our MVC application. In order to create our templates, we use attributes to pass data from the model, through the controllers, and to our views.

Attributes

Thymeleaf has many attributes that we could use. In this book, we focus on a few key ones.

Displaying Data

th:text dynamically populates the contents of an HTML element.

```
<p th:text = "Hello World!"></p>
```

If we want to pull in a value of a variable from the controller, we can use the **variable expressions** syntax, ${}. Let's say we want to pull in the value of a variable called hello.

```
<p th:text = "${hello}"></p>
```

Conditionally Displaying Data

In order to conditionally display data, we need to use th:if and th:unless. th:if will display the value of the element if the expression evaluates to true. If the expression evaluates to false, then the value of the element is NOT displayed. th:unless will display the value of the element if the expression evaluates to false. If the expression evaluates to true, then the value of the element is NOT displayed.

Example

Let's say we want to conditionally display our grocery list. We do have a variable, pantryStatus, that is a String. If pantryStatus is empty, we want to display a list that includes staples like flour, sugar, and rice. If pantryStatus is not empty, we want our grocery list to include our favorite fresh fruits and veggies, like bananas, strawberries, and broccoli.

```
<ol id = "groceryList">
   <li th:if = "${pantryStatus == 'empty'}">Flour</li>
   <li th:if = "${pantryStatus == 'empty'}">Sugar</li>
   <li th:if = "${pantryStatus == 'empty'}">Rice</li>
   <li th:unless = "${pantryStatus == 'empty'}">Bananas</li>
   <li th:unless = "${pantryStatus == 'empty'}">Strawberries</li>
   <li th:unless = "${pantryStatus == 'empty'}">Broccoli</li>
</ol>
```

Iteration

What if our grocery list is large? Typing out each item would be frustrating and inefficient. Instead we could use Thymeleaf to print the values of our grocery list as we iterate through them. th:each is used to iterate through items in an ArrayList. th:block is the command that creates an attribute container around the section we want to use the iteration for.

Example

Our grocery list is stored in an ArrayList called groceries. Each item in our list is an object of type foodItem and has a property called name.

```
<ol id = "groceryList" th:each = "item: ${groceries}">
   <li th:text = "${item.name}></li>
</ol>
```

Template Fragments

A *fragment* in Thymeleaf is a section of HTML that is reusable. This could be a section of our site that includes the grocery store's name, address, and phone number.

th:fragment defines template fragments. We can then use th:replace to denote a piece of HTML to be replaced by the fragment.

Example

For the grocery store application, we may want to keep the grocery store's info in a separate file, fragments.html.

```
<div th:fragment = "groceryStoreInfo">
   <h3>Cool Grocery Store</h3>
   <p>We are at 123 N. 4th Avenue</p>
   <p>Call us at (123) 456-7890</p>
</div>
```

If we want to use the fragment, groceryStoreInfo, in a separate template, index.html, we could use th:replace.

```
<div th:replace = "fragments :: groceryStoreInfo"></div>
```

th:replace tells Thymeleaf to *replace* the div element in index.html with the one in fragments.html named groceryStoreInfo.

Static Resources

We can use th:src to include static resources such as JavaScript files or images in your template. To include external URLs, we can use th:href.

Example

For our grocery store application, we have a few things we want to add.

1. A link to our favorite grocery store's site so we can order online.
2. A link to our styles, styles.css.
3. A picture of our local grocery store.

```
<head>
   <link rel = "stylesheet" type = "text/css" th:href = "@{/css/styles.css}"/>
</head>
<body>
   <h1>My favorite grocery store!</h1>
   <a th:href = "www.grocerystore.com">Grocery Store's Site</a>
   <img th:src = "@{/images/storefront.png}">
</body>
```


- ← Assignment #3: Tech Jobs (MVC Edition)
- TODOs →