ACE Inspiration

Java Web Development Course

Chapter 24 – Thymeleaf Tutorial

Subsidiary Company of ACE Data Systems

Objectives

- To understand modern server-side Java template engine for both web and standalone environments.

Content

What is Thymeleaf?

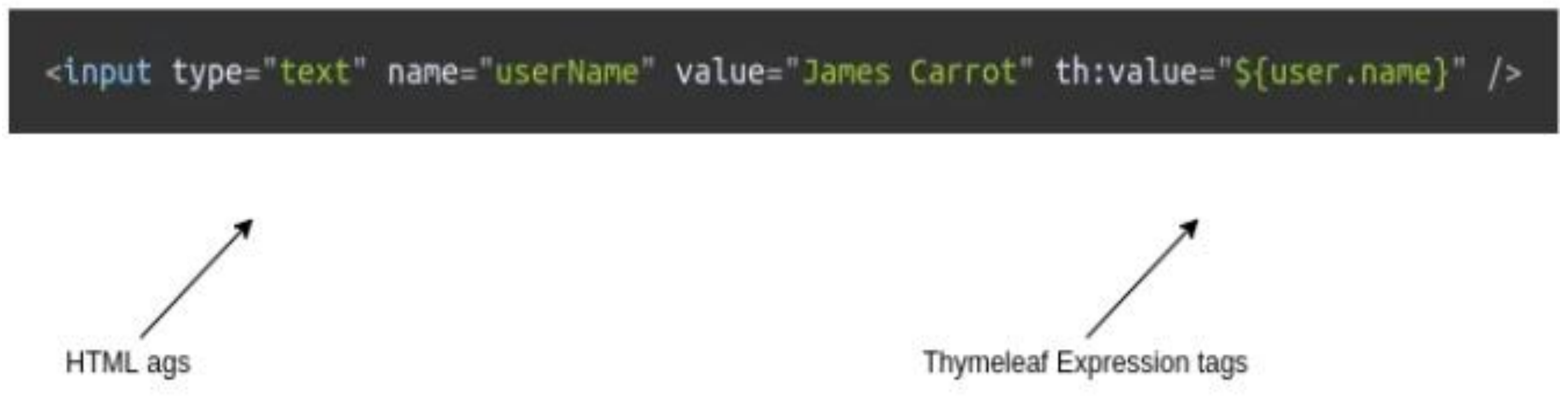
- Thymeleaf is a modern server-side Java template engine for both web and standalone environments, capable of processing HTML, XML, JavaScript, CSS, and even plain text.
- The main goal is to provide an elegant and highly-maintainable way of creating templates.
- It's commonly used to generate HTML views for web apps.
- Thymeleaf template engine is commonly used with the Spring MVC framework to develop web applications.

Thymeleaf Template Processing

- Thymeleaf can process six kinds of templates:
  1. **The HTML template mode** will allow any kind of HTML input, including HTML5, HTML 4 and XHTML.
  2. **The XML template mode** will allow XML input.
  3. **The TEXT template mode** will allow the use of a special syntax for templates of a non-markup nature. Examples of such templates might be text emails or templated documentation.
  4. **The JAVASCRIPT template mode** will allow the processing of JavaScript files in a Thymeleaf application.
  5. **The CSS template mode** will allow the processing of CSS files involved in a Thymeleaf application.
  6. **The RAW template mode** will simply not process templates at all. It is meant to be used for inserting untouched resources (files, URL responses, etc.) into the templates being processed.

What is a Thymeleaf template?

- Thymeleaf template can be an HTML page with some Thymeleaf expressions.
- It includes dynamic content to an HTML page with the help of thymeleaf expressions.
- It can access java code, objects, spring beans, and so on.



```
<input type="text" name="userName" value="James Carrot" th:value="${user.name}" />
```

HTML tags with Thymeleaf Expression

Thymeleaf Standard Expressions

ACE Inspiration

- There are five types of Thymeleaf standard expressions:

  1. ${...}: Variable expressions
  2. *{...} : Selection expressions
  3. #{...} : Message (i18n) expressions
  4. @{...} : Link (URL) expressions
  5. ~{...}: Fragment expressions

Variable expressions

- Variable expressions are the most commonly used ones in the Thymeleaf templates.
- These expressions help bind the data from the template context(model) into the resulting HTML(view).

Syntax:

${VariableName}

Thymeleaf attributes

1. Thymeleaf th:text attribute
2. Thymeleaf th:each attribute
3. Thymeleaf th:if and th:unless attribute
4. Thymeleaf th:switch and th:case attribute
5. Thymeleaf th:fragment attribute

Thymeleaf attributes

th:text attribute

```
<p> Name: <strong th:text="${user.name}"></strong></p>
```

th:each attribute

```
<tbody>
    <tr th:each="employee, state : ${employees}">
        <td th:text="${state.index}"></td>
        <td th:text="${state.count}"></td>
        <td th:text="${employee.firstName}"></td>
        <td th:text="${employee.firstName}"></td>
        <td th:text="${employee.lastName}"></td>
        <td th:text="${employee.email}"></td>
    </tr>
</tbody>
```

Thymeleaf attributes

th:if and th:unless attribute

```
<tbody>
        <tr th:each="user : ${users}">
            <td th:text="${user.userName}"></td>
            <td th:text="${user.email}"></td>
            <td><a class="btn btn-primary" th:if="${user.role} == 'ADMIN'">Update</a>
                  <a class="btn btn-danger" th:if="${user.role} == 'ADMIN'">Delete</a>
                  <a class="btn btn-primary" th:unless="${user.role} == 'ADMIN'">View</a>
            </td>
          </tr>
</tbody>
```

Thymeleaf attributes

th:switch and th:case attribute

```
<div th:switch="${user.role}">
        <p th:case="'ADMIN'">User is an administrator</p>
        <p th:case="'MANAGER'">User is a manager</p>
        <p th:case="'GUEST'">User is a guest</p>
        <!-- * for default case -->
        <p th:case="*">User is some other thing</p>
</div>
```

Thymeleaf attributes

th:fragment and th:replace attribute

Create **header.html** and **footer.html** inside “home” folder under **/resources/templates**

common/header.html

```
<div th:fragment="header">
    <h1> Header Part</h1>
    <hr />
</div>
```

common/footer.html

```
<div th:fragment="footer">
    <hr />
    <h1>Footer Part</h1>
</div>
```

common/fragment.html

```
<body>
<h1> Fragment Expressions :</h1>
<div th:replace="~{common/header :: header}"></div>
<div>
    <h1>Page Body</h1>
</div>
<div th:replace="~{common/footer :: footer}"></div>
</body>
```

Selection Expressions

- The selection expressions are best when used together with form elements.
- As form-inputs are mapped to a @ModelAttribute, it’s easier to bind them without having to worry about managing each input elements individually.
- To use these expressions you first need to define a th:object attribute.

<u>Syntax:</u> *{VariableName}

-
```
<form method="post" th:action="@{/}" th:object="${userInfo}">
            <input type="text" th:field="*{firstName}"/>
            <input type="text" th:field="*{lastName}"/>
            <input type="submit" value="Create User">
</form>
```

Message (i18n) Expressions

- Message expressions let you externalize common texts into a properties file.

Syntax: #{message.property.key}

- create a **messages.properties** file under /resources folder

```
app.name=Spring Boot Thymeleaf Application

welcome.message=Hello, welcome to Spring boot application
```

```
<body>
<h1>Message Expressions Demo:</h1>
<h2 th:text="#{app.name}"></h2>
<h2 th:text="#{welcome.message}"></h2>
</body>
```

Link (URL) Expressions

- Link expressions are meant to build URLs in Thymeleaf templates.

Syntax: @{link}

UserController.java

```
//http://localhost:8080/link-expression

@GetMapping("link-expression")
 public String linkExpression(Model model){
      model.addAttribute("id", 1);
      return "link-expression";
 }
```

Link (URL) Expressions

link.html

```
<head>
	<meta charset="UTF-8">
	<title>Link Expressions</title>
	<link th:href="@{/css/demo.css}" rel="stylesheet" />
</head>

<body>
	<h1>Link Expressions Demo:</h1>
	<a th:href="@{/variable-expression}"> variable-expression </a>
	<a th:href="@{/selection-expression}"> selection-expression </a>
	<p> <a th:href="@{link-expression/{id}(id=${id})}">link with parameter</a></p>
</body>
```

ACE Inspiration

Thank you!!

Q&As

References

- https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html
- https://www.javaguides.net/p/thymeleaf-tutorial.html