Thymeleaf Complete Guide

1. Introduction to Thymeleaf

Thymeleaf is a modern server-side Java template engine for web applications. It is designed to work with Spring Boot to dynamically generate HTML pages.

Thymeleaf allows the combination of logic and HTML to render dynamic content and provides excellent integration with Spring MVC.

2. Setting up Thymeleaf in Spring Boot

To set up Thymeleaf in a Spring Boot project, include the dependency in your pom.xml file:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

3. Basic Syntax and Structure

Thymeleaf templates use HTML5-compatible syntax. Add the xmlns:th attribute to the <html> tag: <html xmlns:th="http://www.thymeleaf.org">. Key features include 'th:text' for text replacement and 'th:utext' for unescaped HTML.

4. Dynamic Links and URLs

Thymeleaf provides 'th:href', 'th:src', and 'th:action' for dynamically setting URLs in links, images, and form actions.

Examples:

<a th:href="@{/home}">Home

5. Conditional Rendering

Thymeleaf supports 'th:if' and 'th:unless' for conditional rendering, and 'th:switch'/'th:case' for switch-like logic.

Example:

Welcome, User

6. Loops

Use 'th:each' to iterate over collections. Access loop index and status through 'Stat' objects:

<li th:each="item : ${items}" th:text="${item.name}">Item

7. Form Handling

Use 'th:field' to bind form inputs to model attributes in Spring MVC. This allows data submission from forms:

<form th:action="@{/register}" th:object="${user}" method="post">

<input th:field="*{username}" placeholder="Username"/>

8. Fragments and Layouts

Fragments allow reuse of components like headers and footers. Define reusable blocks with 'th:fragment':

<header th:fragment="headerFragment">...</header>

Include fragments with 'th:insert' or 'th:replace'.

9. Advanced Features

Inline JavaScript and CSS with 'th:inline', conditional classes and styles with 'th:classappend' and 'th:styleappend', and utility objects like #dates, #strings for manipulation.

10. Utility Objects

Thymeleaf includes utility objects for data manipulation such as #dates for formatting dates, #numbers for number formatting, and #strings for string operations.

11. Internationalization (i18n)

Thymeleaf supports message resolution for i18n using messages.properties files. Use '#{key}' syntax to load messages, e.g., Welcome!

12. Expressions and Precedence

Use complex expressions with precedence, like '${(user.age > 18) ? 'Adult' : 'Minor'}'. The 'th:with' attribute also allows setting default values for variables.