Thymeleaf

Thymeleaf is a server-side Java template engine used for rendering HTML views in Spring web applications. It is used to create dynamic content by combining static HTML with Java data objects.

Core Concepts

Thymeleaf templates look like regular HTML but include special attributes and expressions that make them dynamic. These attributes and expressions allow us to render data, iterate over collections, and create links in a way that integrates with Spring.

Key Expression Symbols

1. **${} (Variable Expressions)**

   This type of expression is used to access and display values from the model attributes (i.e. data passed from the Java controller to the template).

   - Example:

     Default Name

   In this example, ${name} will be replaced with the value of the name attribute from the model. If name is "John," it will render as:

     - John

2. ***{} (Selection Variable Expressions)**

   The *{} syntax is used when you want to reference properties of an object selected with th:object, often in form handling.

   - Example:

     <form th:object="${user}">

     <input type="text" th:field="*{name}" />

     </form>

   Here, *{name} refers to the name property of the user object, which is defined by th:object="${user}".\

3. **@{} (URL Expressions)**

   @{} is used to generate URLs dynamically. It helps link to other pages or static resources (like images or stylesheets).

   - Example:

     <a th:href="@{/courses}">Courses

   This generates a link to the /courses endpoint, which could be handled by a controller in your Spring application.

Core Attributes in Thymeleaf

1. **th:text (Text Replacement)**

   Replaces the content of an HTML element with a variable from the model.

   - Example:

     <h1 th:text="${welcomeMessage}">Welcome</h1>

     If welcomeMessage contains "Hello, User!", the rendered HTML will be:

     <h1>Hello, User!</h1>

2. **th:each (Iteration)**

   Used to iterate over a collection of items (like a list or array).

   - Example:

      <li th:each="course : ${courses}" th:text="${course.title}">

   If **courses** is a list of course objects, this will generate a list (<ul>) where each <li> contains the title of a course. For example:

   -
   - Introduction to Java
   - Spring Boot Basics

- Advanced Web Development
-
-

3. **th:if and th:unless (Conditional Rendering)**

   th:if renders an element if a condition is true, while th:unless renders it if the condition is false.

   - Example:

     Welcome, Admin!

     Welcome, User!

   If user.admin is true, it shows "Welcome, Admin!" Otherwise, it shows "Welcome, User!"

4. **th:object, th:action, and th:field (Form Handling)**

   - These attributes are used for working with form inputs.
   - th:action defines the URL where a form should be submitted.
   - th:field binds an input field to a property of an object.
   - Example:

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

      <input type="text" th:field="*{name}" />

      <button type="submit">Submit</button>

   - </form>

   Here, the form submits data to the /submit endpoint, and the input field is bound to the name property of the user object.

Example: Putting It All Together

Given a Course object and a list of courses, and you want to display them on a webpage:

Java Controller:

```
@GetMapping("/courses")
public String getCourses(Model model) {
   List<Course> courses = List.of(
      new Course("CS101", "Intro to Computer Science", "A"),
      new Course("MATH201", "Calculus II", "B+")
   );
   model.addAttribute("courses", courses);
   return "courses";
}
```

Thymeleaf Template (courses.html):

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <title>Courses</title>
</head>
<body>
   <h1 th:text="${pageTitle}">Course List</h1>
   <ol>
      <li th:each="course : ${courses}">
         <span th:text="${course.code}"></span>:
         <span th:text="${course.title}"></span> -
         <span th:text="${course.grade}"></span>
      </li>
   </ol>
</body>
```

</html>

If pageTitle is set to "Available Courses" and courses contains the two courses above, this will generate:

<h1>Available Courses</h1>

  CS101: Intro to Computer Science - A

  MATH201: Calculus II - B+

Summary

Thymeleaf makes it easy to create dynamic web pages by allowing you to:

- Access and display data using ${} for variables.
- Iterate over collections using th:each.
- Conditionally render elements using th:if and th:unless.
- Construct dynamic links using @{}.
- Bind form inputs to Java objects with th:field and th:object.

With these tools, Thymeleaf integrates seamlessly with Spring MVC, allowing you to build user-friendly, data-driven websites in your Spring Boot applications.