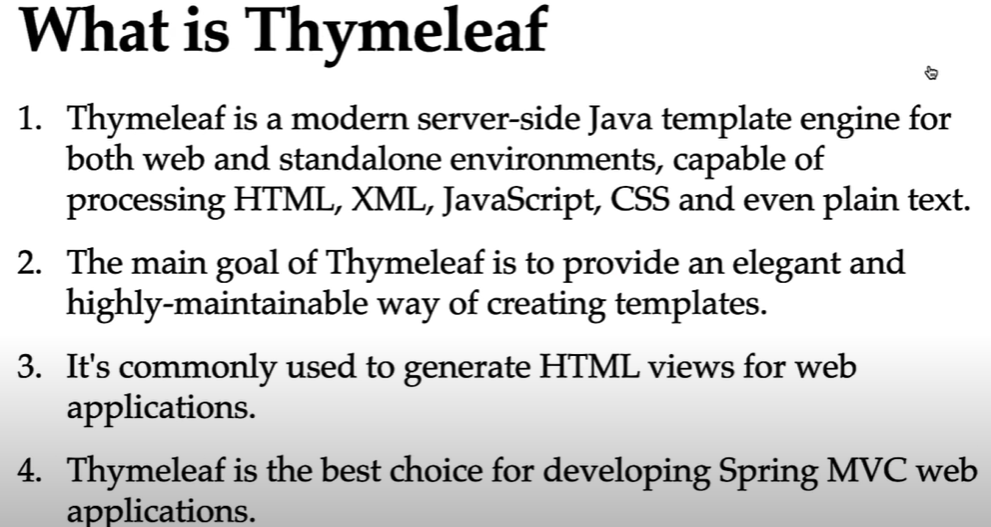
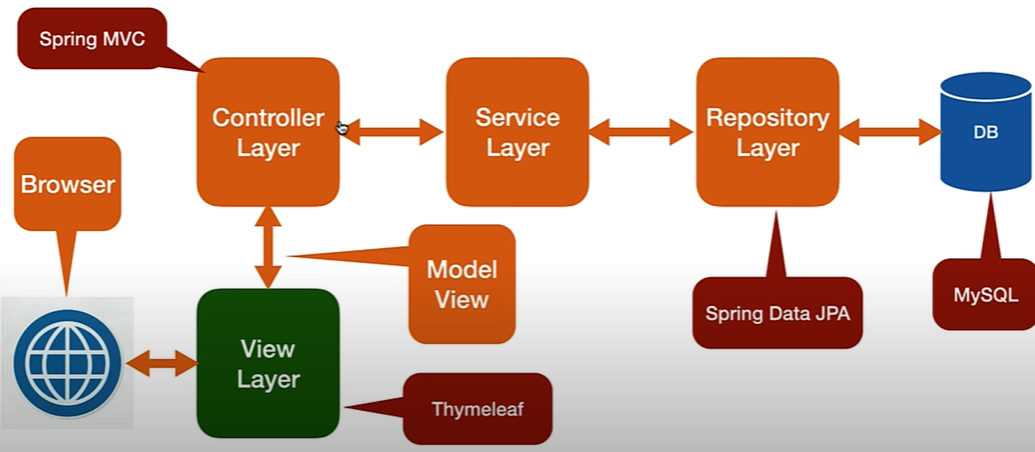
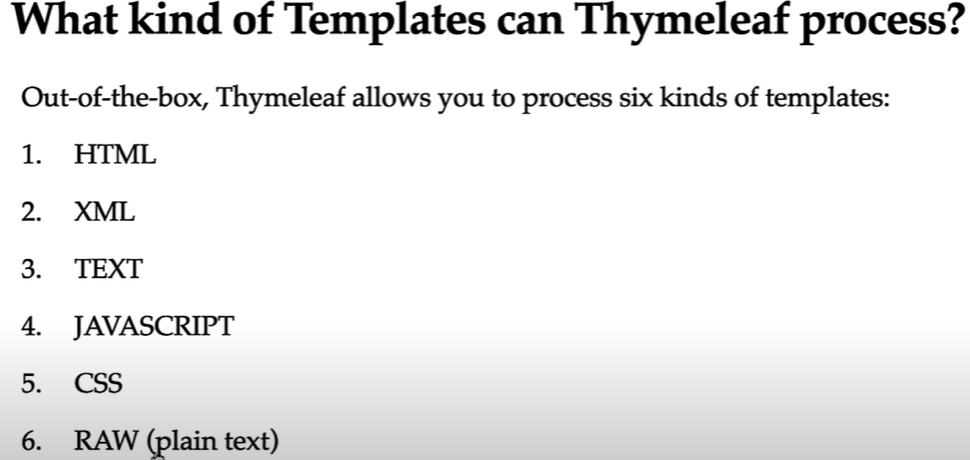
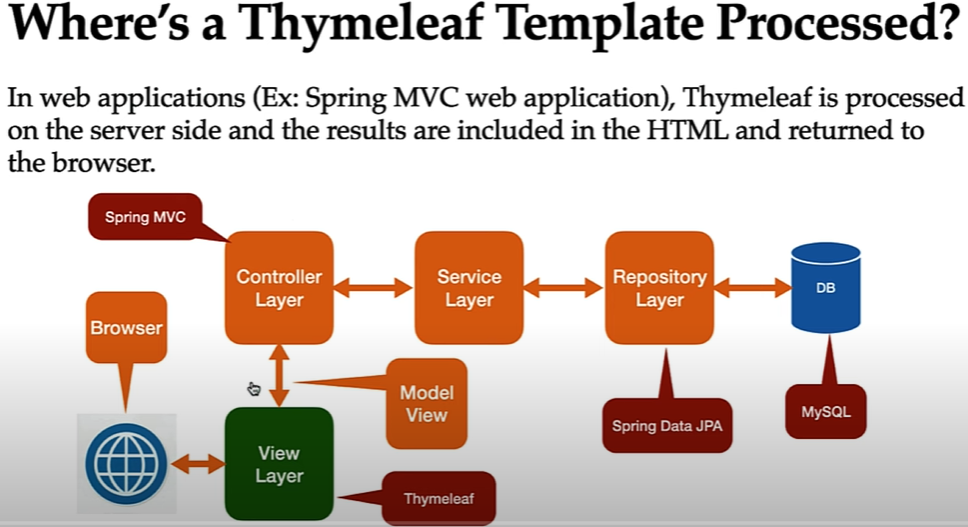
**Thymleaf**









### Step-by-Step Workflow of Thymeleaf Engine:

#### 1. User Request

* A user sends an HTTP request (e.g., via a browser) to a Spring Boot application.

#### 2. Controller Processes Request

* A Spring @Controller method is mapped to handle the request.
* It performs business logic and prepares data (model attributes).

#### 3. Controller Returns View Name + Model

* The controller returns:
* A **view name** (e.g., "home" or "greeting")
* A **Model** object (key-value pairs of data)

#### 4. View Resolver Looks Up Template

* Spring Boot uses **InternalResourceViewResolver** (or another view resolver) to map the view name to a **Thymeleaf template** file (e.g., resources/templates/greeting.html).

#### 5. Thymeleaf Engine Kicks In

* Thymeleaf loads the template and:
* Parses the HTML
* Finds special Thymeleaf attributes like th:text, th:each, etc.
* Replaces placeholders with actual model data

#### 6. HTML is Rendered

* Thymeleaf renders the complete HTML page with dynamic content.

#### 7. Response Sent Back

* The rendered HTML is returned to the browser as the HTTP response.

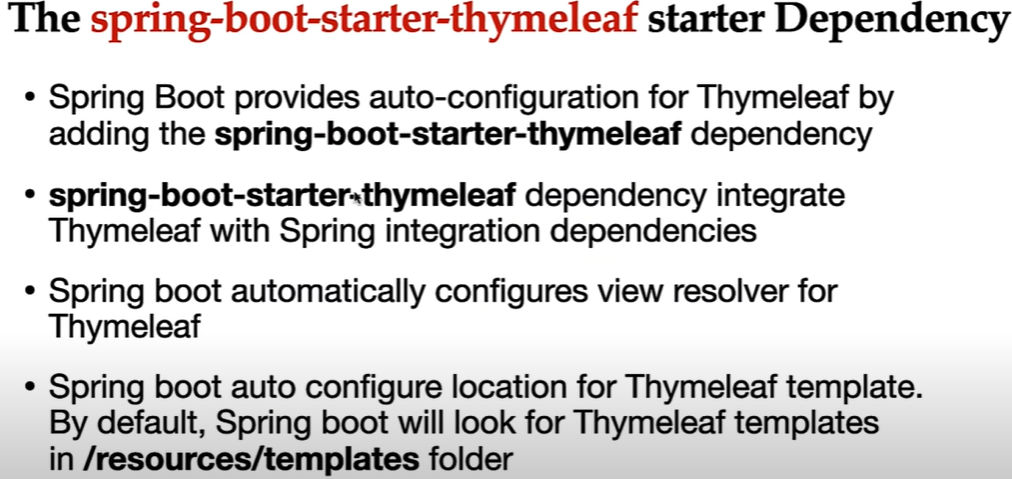
**<dependency>**

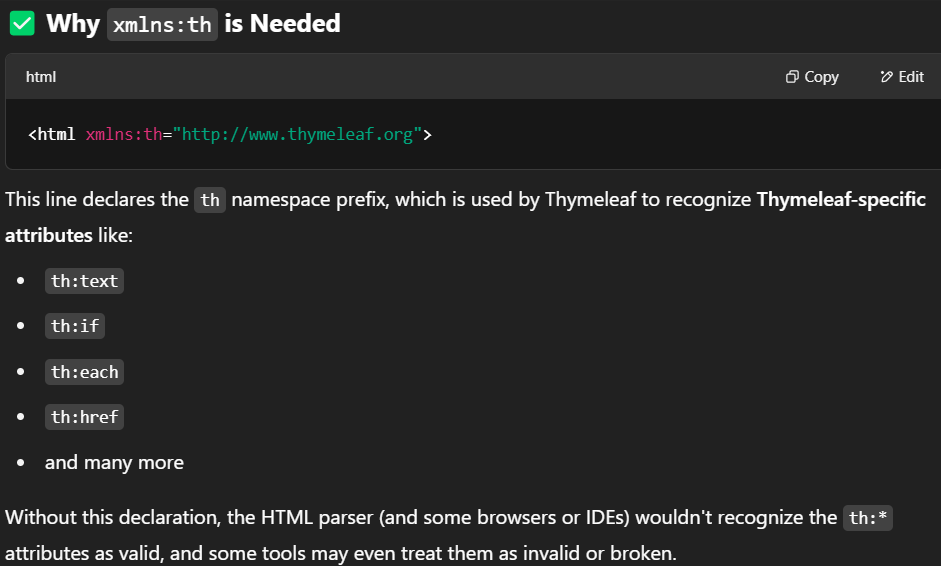
**<groupId>org.springframework.boot</groupId>**

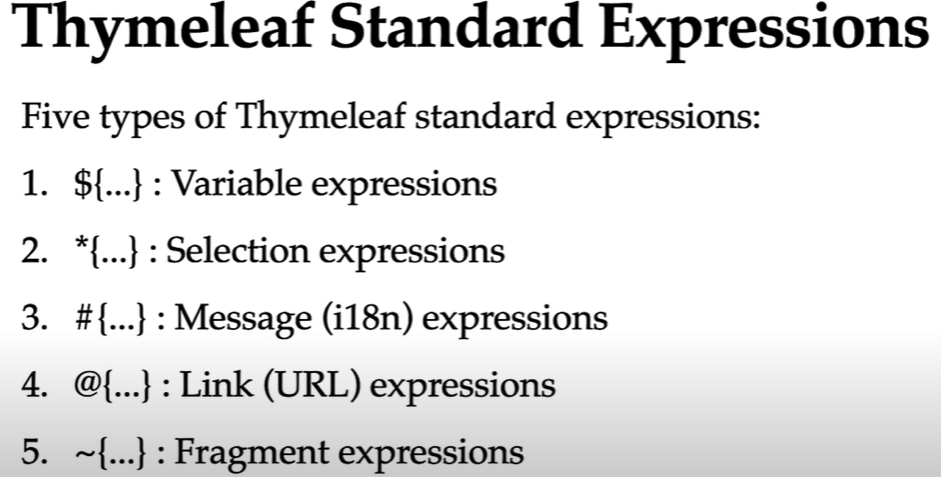
**<artifactId>spring-boot-starter-thymeleaf</artifactId>**

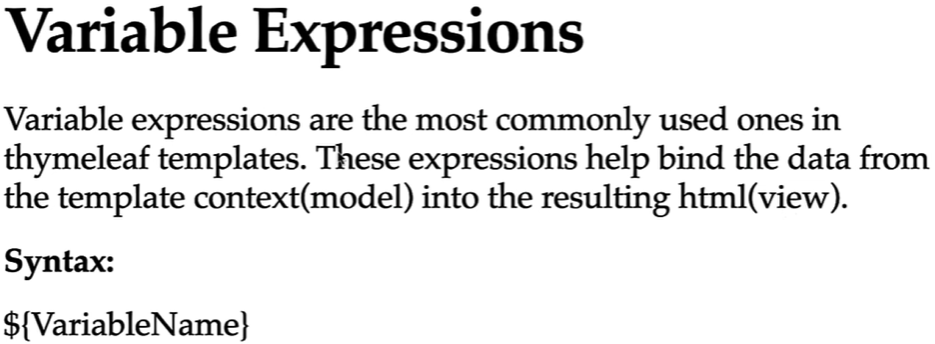
**<version>3.0.1</version>**

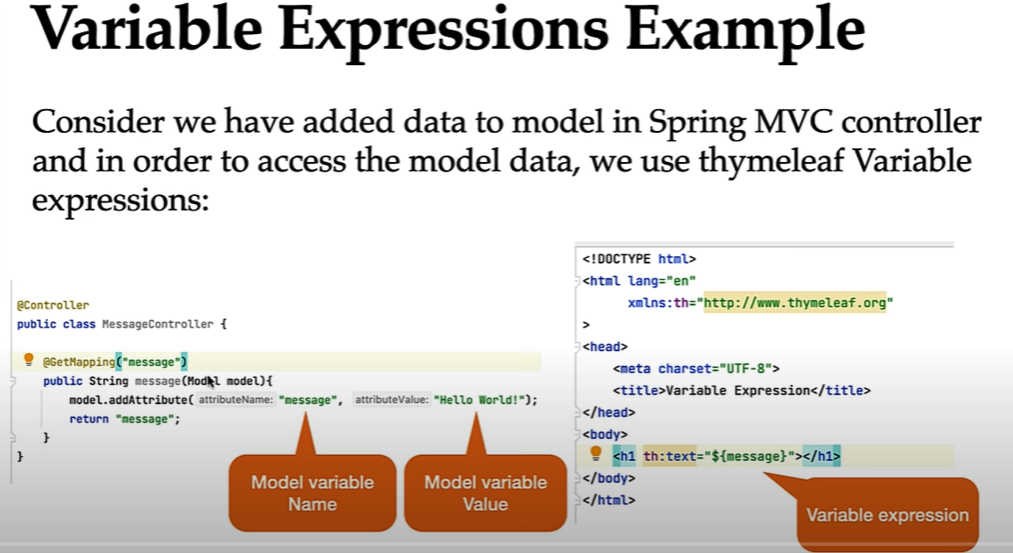
**</dependency>**  
  
Add above thymleaf dependency which will internally contains thymleaf integration dependency.

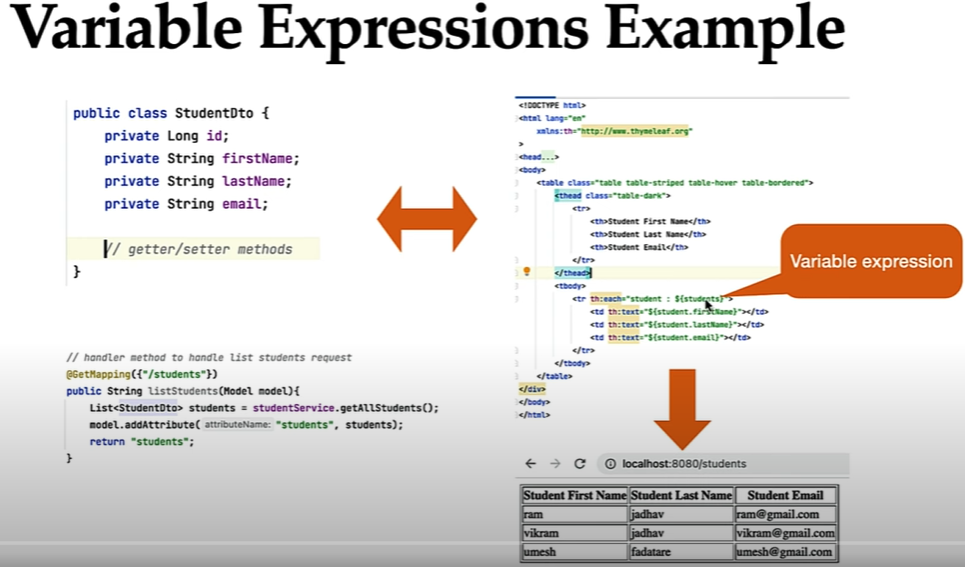




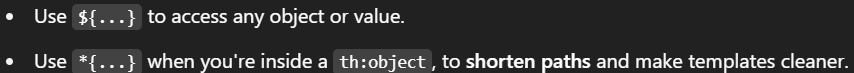


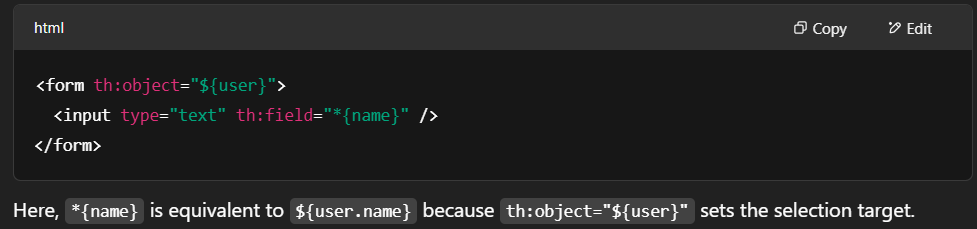


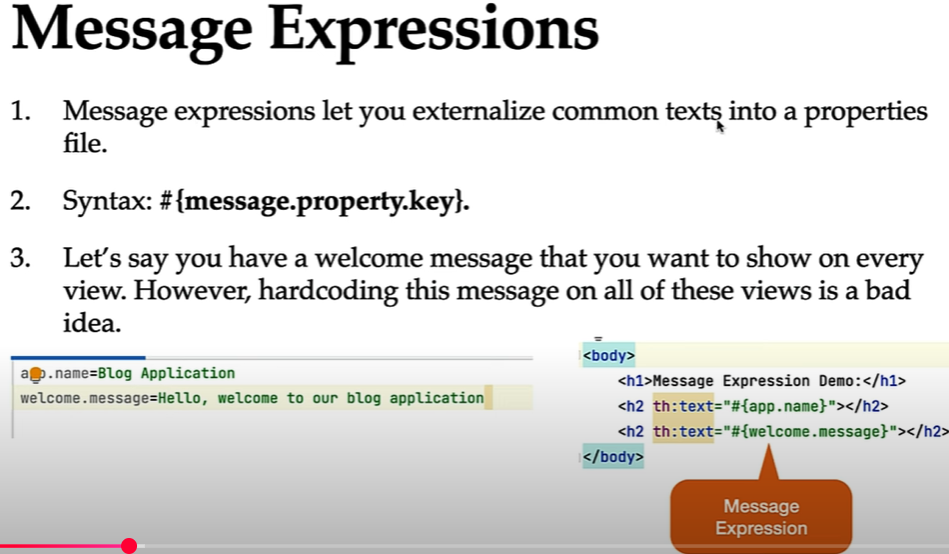








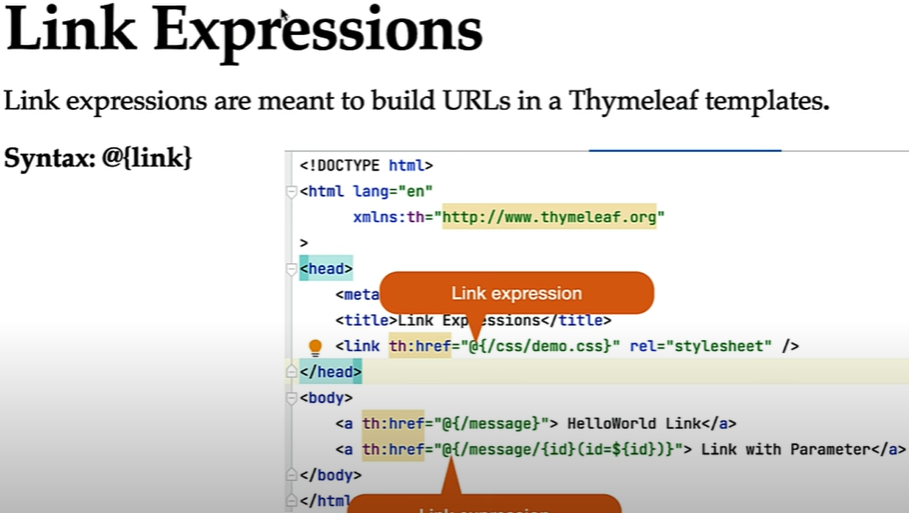


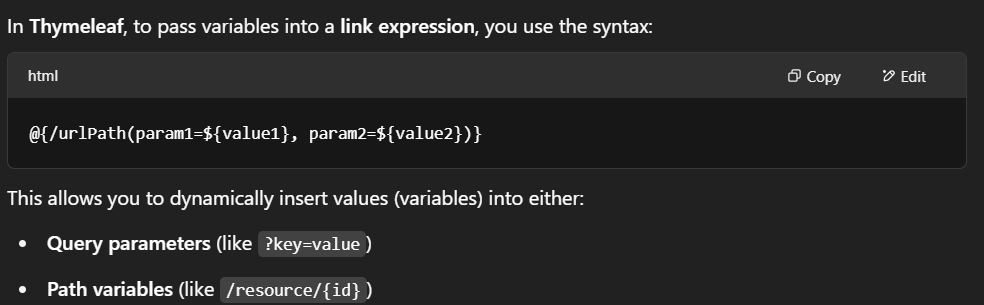


Here the default properties filename is **messages.properties**. You can change filename also by using below code in application properties

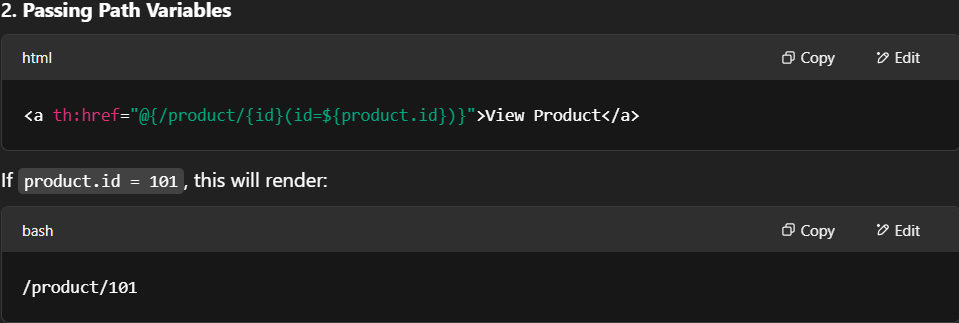
Let’s say you want to use a file named \*\*labels.properties\*\* instead of the default messages.properties.

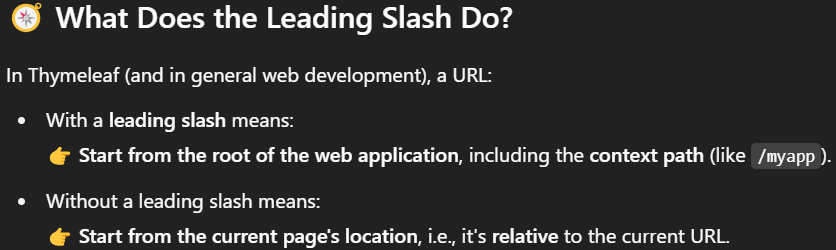
**spring.messages.basename=labels**

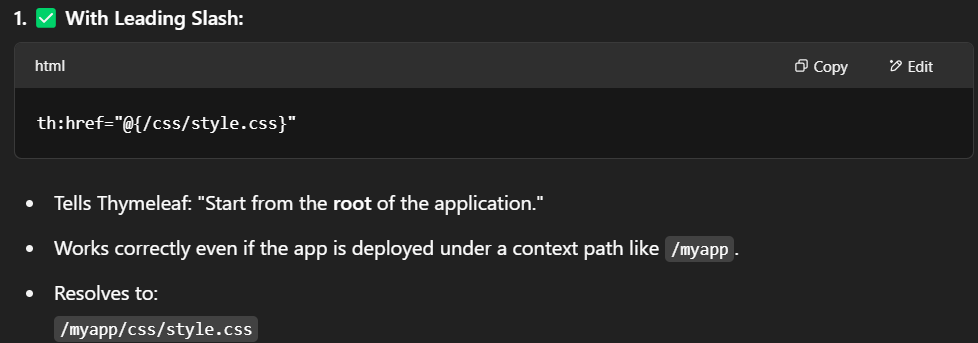


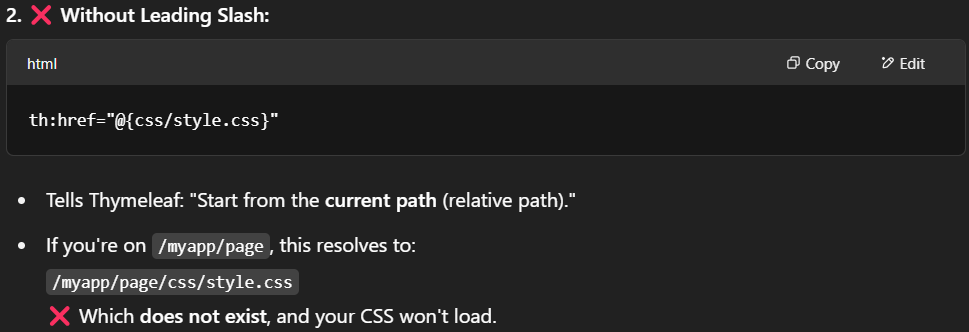


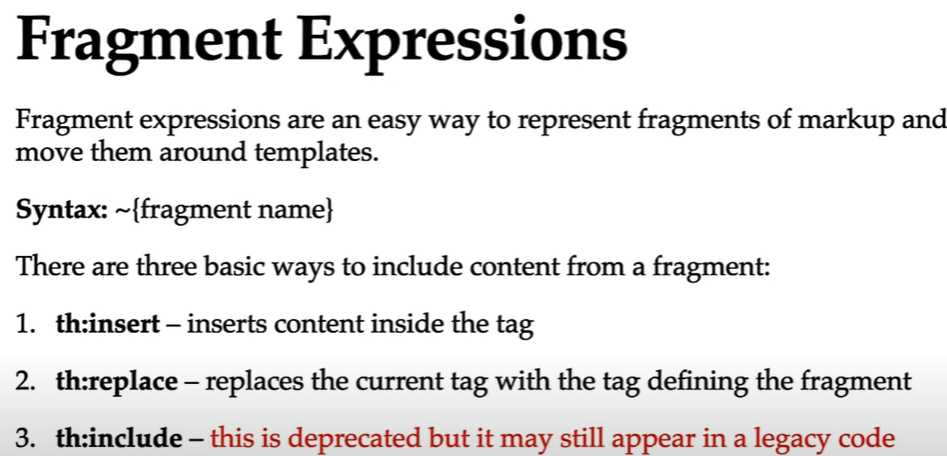


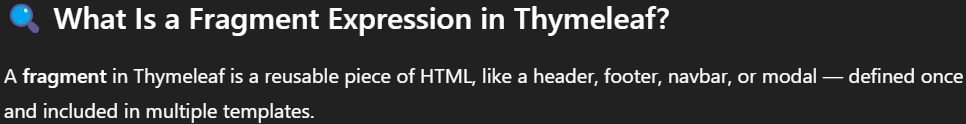


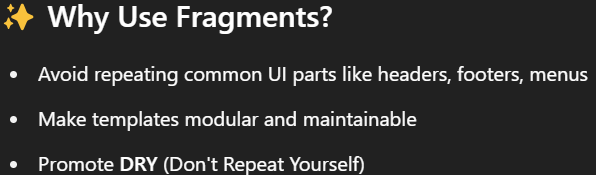




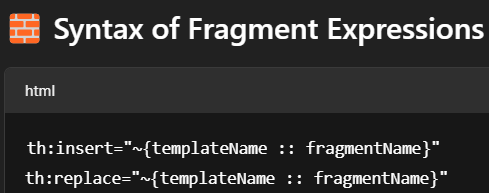






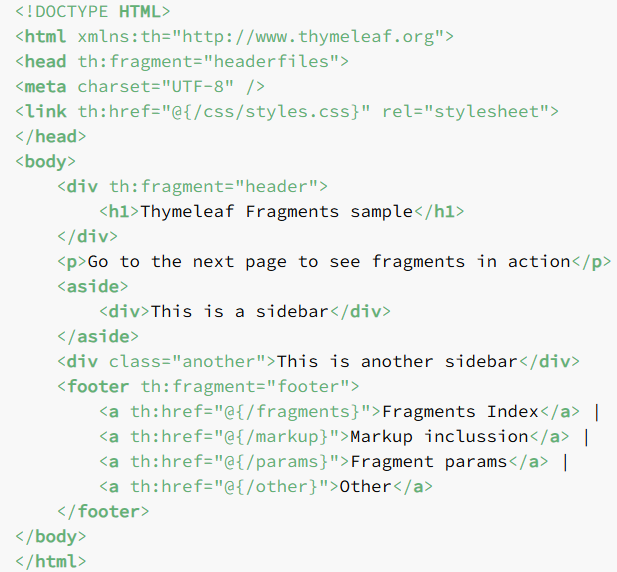






Let me explain you how insert and replace works

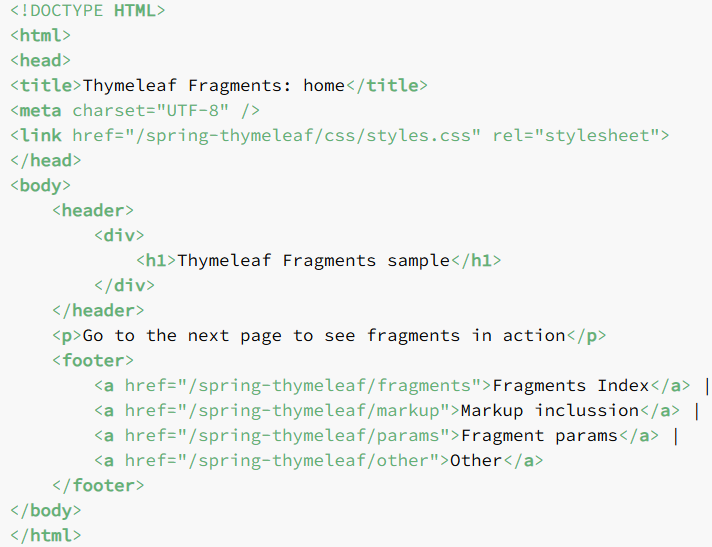
**general.html**



**fragment.html**



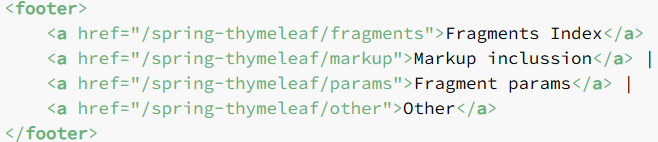
**After the rendering and fragments inclusion, the returned content is:**

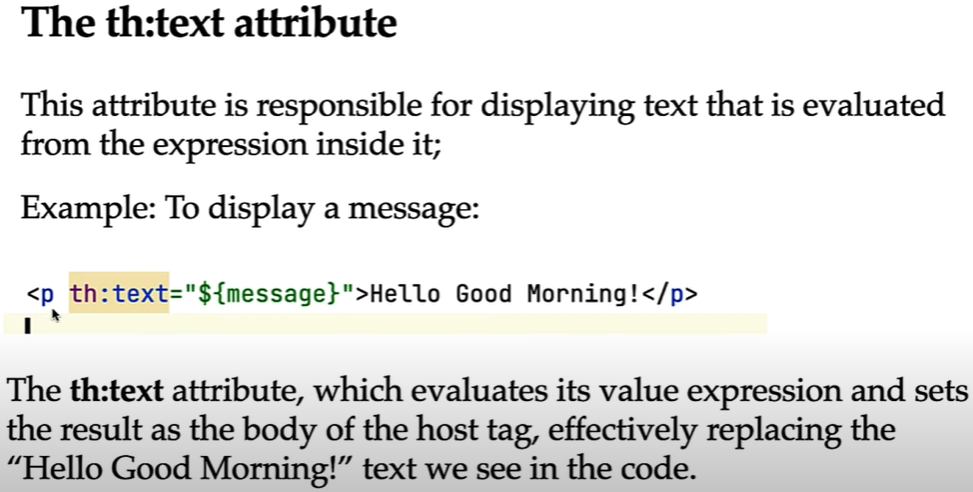


insert: It will put the fragment tag inside the current tag.

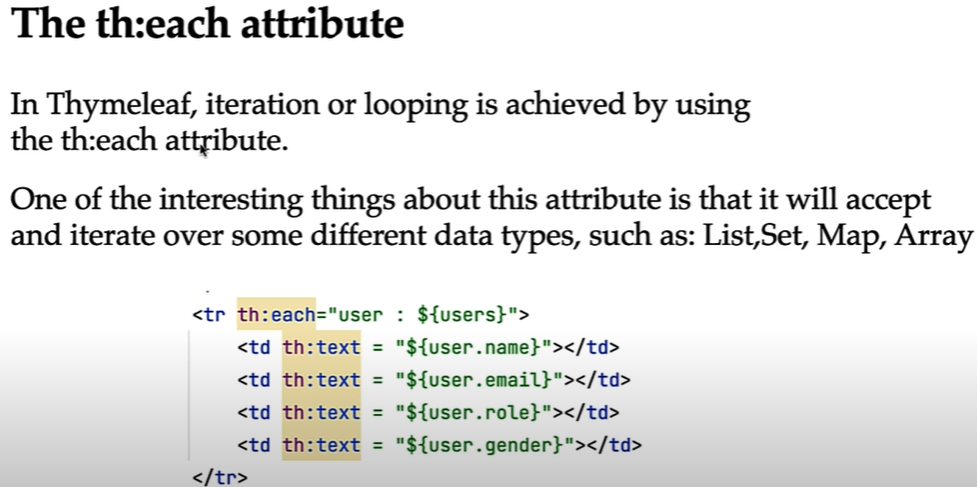


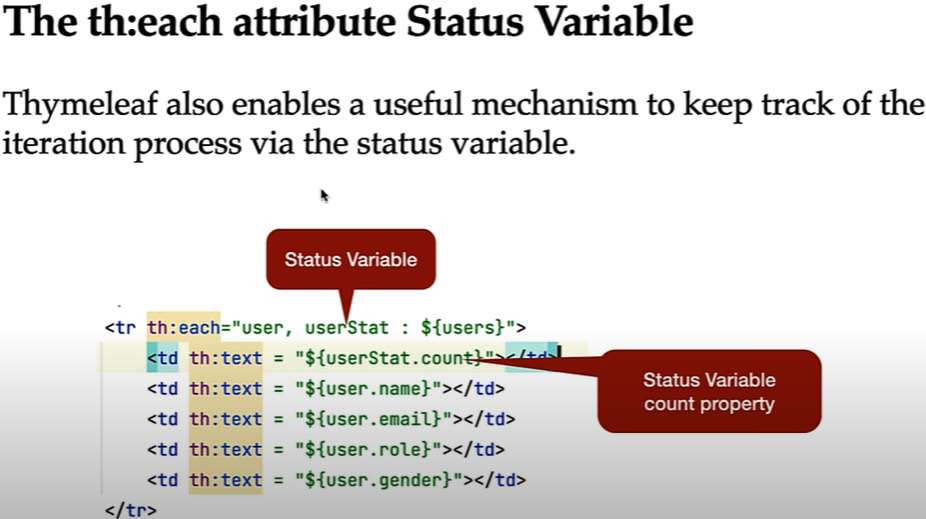
replace: It will the current tag from the fragment tag.

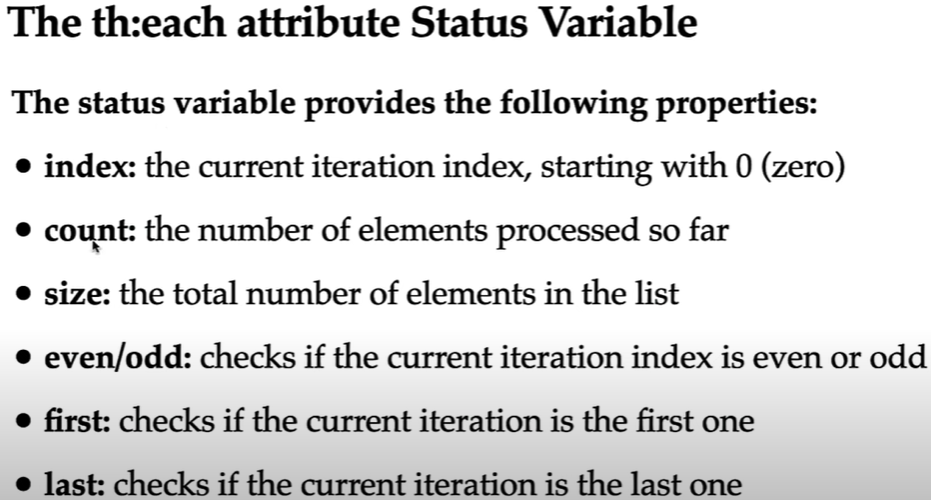


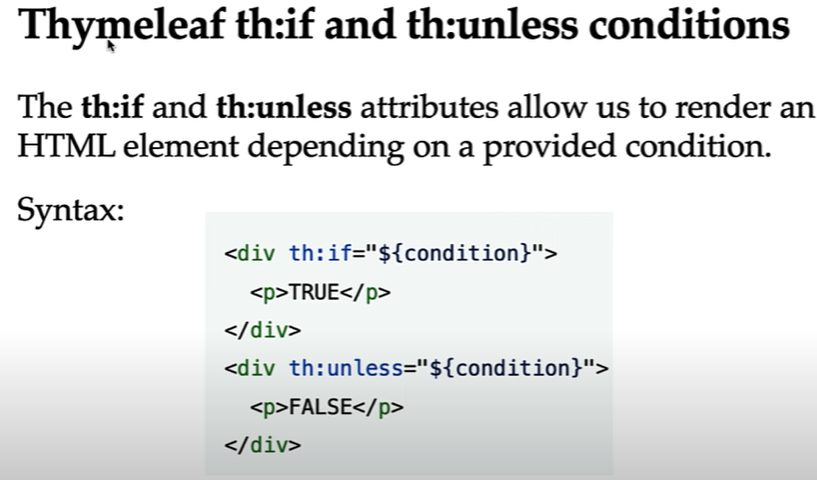


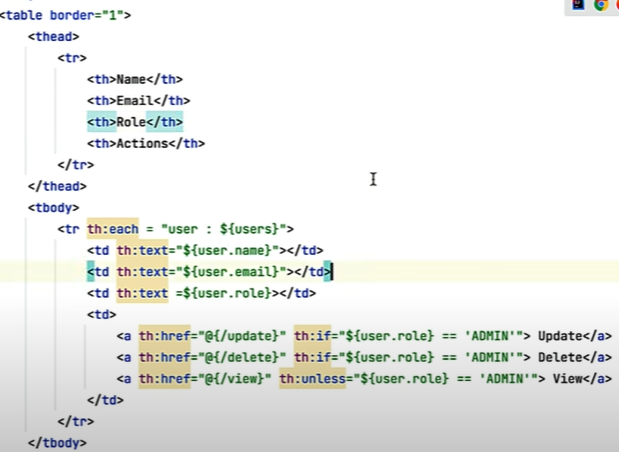
If “message” is not present then nothing will print, If present then it will replace “Hello Good Morning” with “message” key value



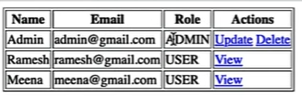








Result:



Here unless means true when it not satisfy the condition.

