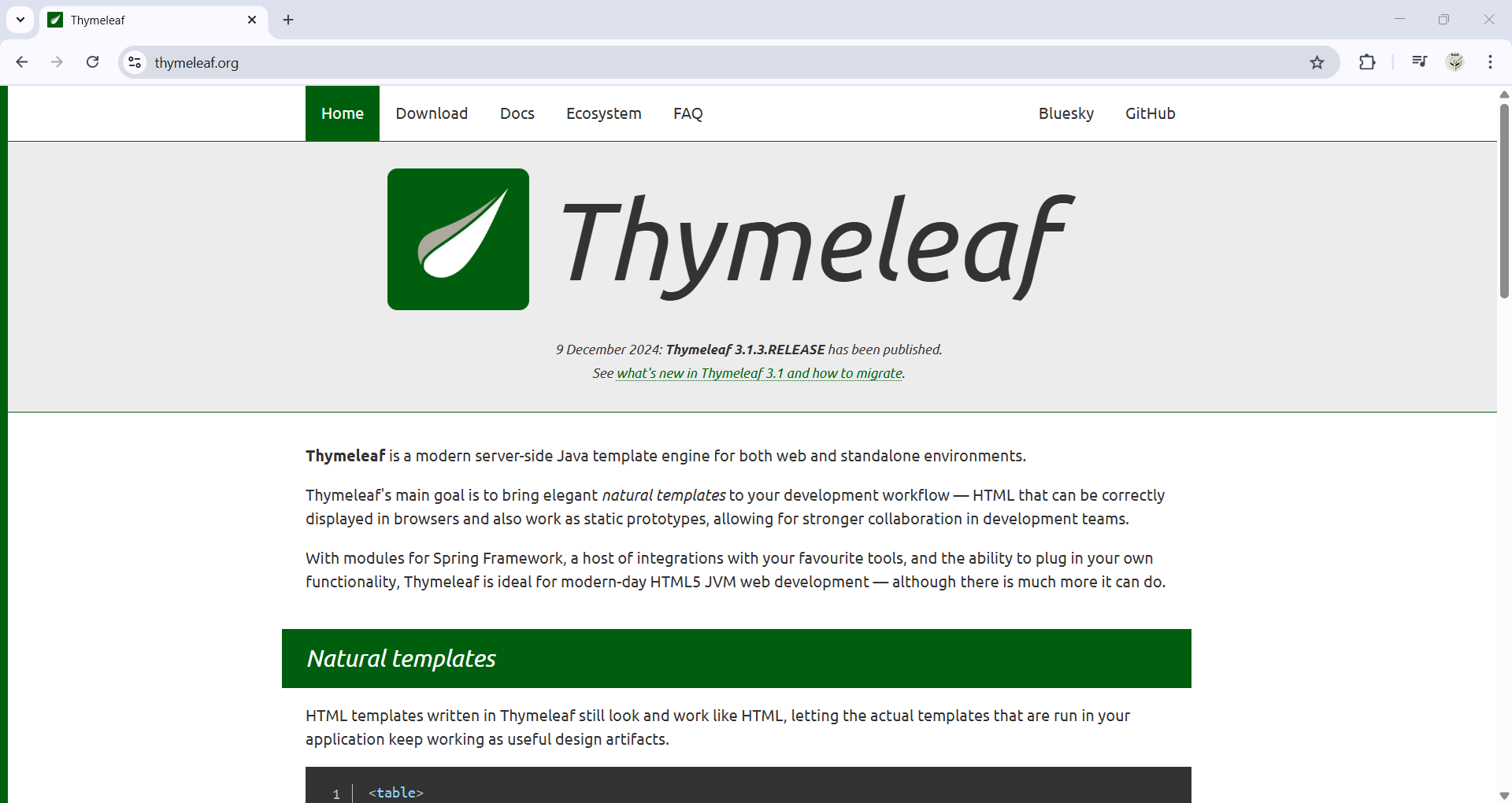
# Thymeleaf



## Thymeleaf là gì?

* Thymeleaf là một **modern server-side Java template engine** cho cả môi trường web và môi trường độc lập.
* Thymeleaf được sử dụng để xử lý và tạo ra các trang **HTML**, **XML**, **JavaScript**, **CSS**, **text**, và **raw content**.

## Expression (biểu thức) trong Thymeleaf

* **${…}**: Lấy giá trị của một biến.
* **\*{…}**: Lấy giá trị của một biến được chỉ định.
* **#{…}**: Lấy message.
* **@{…}**: Lấy đường dẫn URL dựa theo context của server.

### Biểu thức ${…} - Truy cập giá trị biến

Biểu thức **${…}** cho phép ta truy cập giá trị biến trong phạm vi của Context hoặc Model.

**Controller**:

model.addAttribute("today", "Monday");

**HTML**:

<p>Today is: <span th:text="${today}"></span>.</p>

**Tương đương với**:

<p>Today is: <span>Monday</span>.</p>

### Biểu thức \*{…} - Truy cập giá trị biến trên các phần tử

Biểu thức **\*{…}**, còn được gọi là **asterisk syntax**, có chức năng giống ${…}, tức là lấy giá trị của một biến. Tuy nhiên, nó sẽ lấy giá trị của biến trong ngữ cảnh của **th:object**.

<div th:object="${session.user}">

<!-- th:object tồn tại trong phạm vi của thẻ div này -->

<!-- Lấy tên từ đối tượng session.user -->

<p>Name: <span th:text="\*{firstName}"></span>.</p>

<!-- Lấy lastName từ đối tượng session.user -->

<p>Surname: <span th:text="\*{lastName}"></span>.</p>

</div>

**${…}** sẽ lấy giá trị cục bộ trong **Context** hoặc **Model**, trong khi **\*{…}** sẽ lấy giá trị từ đối tượng đã được chỉ định bởi **th:object**.

### Biểu thức #{…} - Lấy thông báo (Message)

Ví dụ, trong tệp cấu hình **.properties**:

home.welcome=Hello bạn

Thì cách lấy nó ra nhanh nhất là:

<p th:utext="#{home.welcome}">Xin chào các bạn!</p>

Đoạn văn bản tiếng Việt bên trong thẻ p sẽ bị **thay thế** bởi Thymeleaf khi render **#{home.welcome}**.

### Biểu thức @{…} - Lấy URL dựa trên ngữ cảnh máy chủ

**@{...}** xử lý và trả về giá trị URL dựa trên ngữ cảnh của máy chủ. Ta có thể sử dụng nó để tạo các liên kết URL động.

Ví dụ:

<!-- Tương đương 'http://localhost:8080/order/details?orderId=3' -->

<a

href="details.html"

th:href="@{http://localhost:8080/order/details(orderId=${o.id})}"

>view</a>

<!-- Tương đương '/order/details?orderId=3' -->

<a href="details.html" th:href="@{/order/details(orderId=${o.id})}">view</a>

Nếu ta bắt đầu bằng dấu /, nó sẽ là URL tương đối và sẽ được xác định dựa trên ngữ cảnh máy chủ của chúng ta.

## Tóm lược cú pháp thường gặp

|  |  |  |  |
| --- | --- | --- | --- |
| **Chủ đề/Mục đích** | **Cú pháp** | **Giải thích** | **Lưu ý / Mẹo** |
| Hiển thị nội dung | th:text="${...}" | Hiển thị giá trị bên trong văn bản | Thymeleaf tự escape, tránh XSS |
| Vòng lặp danh sách | th:each="item : ${list}" | Lặp qua tập hợp trong model | item không cần `${}` vì là biến mới |
| Hiển thị có điều kiện | th:if="${cond}" / th:unless="${cond}" | Hiển thị hoặc ẩn theo điều kiện | th:unless là phủ định th:if |
| Link động | th:href="@{/path/{id}(id=${val})}" | Gán URL động với tham số | Tự thêm context path |
| Ảnh động | th:src="@{/img/logo.png}" | Gán URL động cho ảnh | Tương tự th:href |
| Giá trị input | th:value="${val}" | Gán giá trị cho thuộc tính value | Dùng input readonly hoặc form |
| Form binding | th:object="${obj}" + th:field="\*{field}" | Binding form đến object trong model | \*{} sử dụng context của th:object |
| Thêm class điều kiện | th:classappend="${cond} ? 'c1' : 'c2'" | Append CSS class dựa trên cond | th:class thay cho class toàn phần |
| Biến tạm | th:with="var=${expr}" | Khai báo biến tạm trong template | Chỉ dùng trong scope chứa th:with |
| Tách layout | th:replace="~{layout/master :: frag}" | Chèn fragment từ layout | DRY, tái sử dụng layout |
| Include fragment | th:include="~{common/footer}" | Nhúng fragment | Không thay thế container |

## Ví dụ chi tiết từng cú pháp

### th:text - Hiển thị nội dung

**Controller**:

@GetMapping("/hello")

public String hello(Model model) {

model.addAttribute("ten", "Nguyễn Văn A");

return "hello";

}

**HTML**:

<span th:text="${ten}">Giá trị mặc định</span>

### th:each - Vòng lặp danh sách

**Controller**:

@GetMapping("/products")

public String showProducts(Model model) {

List<Product> list = List.of(...);

model.addAttribute("productList", list);

return "product-list";

}

**HTML**:

<tr th:each="p : ${productList}">...

<td th:text="${p.name}"></td>

</tr>

### th:if / th:unless - Hiển thị có điều kiện

**Controller**:

@GetMapping("/status")

public String status(Model model) {

model.addAttribute("user", u);

return "status";

}

**HTML**:

<p th:if="${user.active}">Active</p>

<p th:unless="${user.active}">Inactive</p>

### th:href - Link động

**Controller**:

@GetMapping("/profile/{id}")

public String prof(@PathVariable Long id, Model model) {

model.addAttribute("u", userService.findById(id));

return "profile";

}

**HTML**:

<a th:href="@{/user/{id}(id=${u.id})}">Profile</a>

Giải thích: `{id}` đóng vai trí placeholder, `(id=${u.id})` gán giá trị; @{} thêm context path.

### th:value - Gán giá trị input

**Controller**:

@GetMapping("/form")

public String form(Model model) {

model.addAttribute("user", new User());

return "form";

}

**HTML**:

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

### th:object + th:field - Form binding

**Controller**:

@GetMapping("/edit")

public String editForm(Model model) {

model.addAttribute("user", userService.get());

return "edit";

}

**HTML**:

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

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

</form>

Giải thích: `th:object` đặt context object; `\*{name}` tương đương `${user.name}`.

### th:classappend - Thêm class điều kiện

**Controller**:

@GetMapping("/dash")

public String dash(Model model){...}

**HTML**:

<div th:classappend="${user.admin} ? 'admin' : 'user'"></div>

### th:with - Biến tạm

**Controller**:

@GetMapping("/welcome")

…

**HTML**:

<div th:with="fullName=${user.firstName + ' ' + user.lastName}">...

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

</div>

### th:replace - Tách layout

**Controller**:

@GetMapping("/home")

public String home(){return "home";}

**HTML**:

<div th:replace="~{layout/master :: header}"></div>

Sample layout/master.html:

<html th:fragment="layout">

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

</html>

### th:include - Include fragment

**Controller**:

@GetMapping("/page")

public String page(){return "page";}

**HTML**:

<footer th:include="~{common/footer}"></footer>

Sample common/footer.html:

<footer th:fragment="footer">...</footer>