TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

LẬP TRÌNH JAVA NÂNG CAO

CHƯƠNG 5: Thymeleaf

Giảng Viên Giảng Dạy:
ThS. Nguyễn Minh Tân
ThS. Đặng Đình Hòa
ThS. Trần Công Thanh

HỌC KỲ III – NĂM HỌC 2023-2024

KHÓA 27T-IT

Nội dung

1. Giới thiệu về Thymeleaf
2. Kiến trúc Thymeleaf
3. Cú pháp
4. Cách sử dụng
5. Q & A

THYMELEAF

Giới thiệu Thymeleaf

https://www.thymeleaf.org/documentation.html

- Thymeleaf là một template engine được sử dụng để hiển thị dữ liệu trên các trang web. Nó là một công cụ mã nguồn mở và được viết bằng Java, được phát triển bởi Daniel Fernández. Thymeleaf cho phép bạn kết hợp HTML hoặc XML với các biểu thức Thymeleaf để tạo ra các trang web động.
- Một trong những điểm mạnh của Thymeleaf là khả năng tích hợp với Spring Framework để xây dựng các ứng dụng web. Thymeleaf hỗ trợ các tính năng như việc hiển thị dữ liệu từ model, xử lý sự kiện, tạo các mẫu thư từ và các thông báo lỗi, v.v.

Các loại template của Thymeleaf

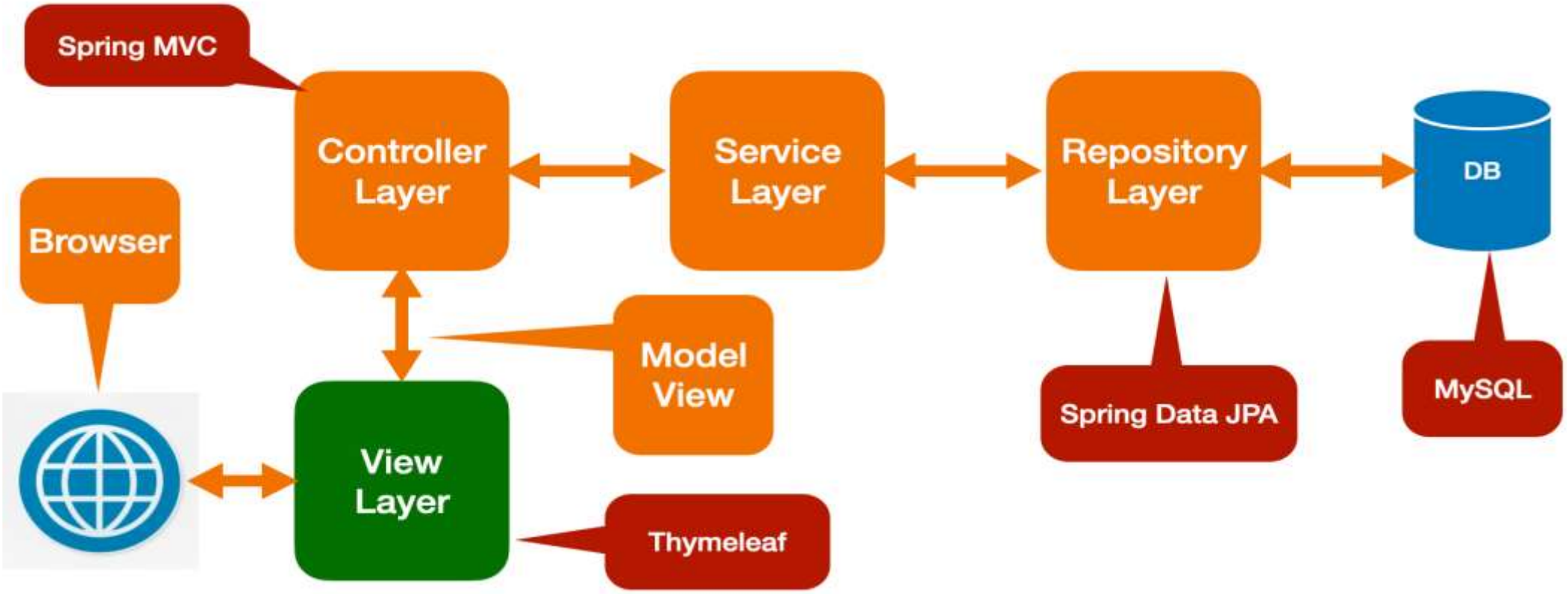
- Thymeleaf cho phép xử lý sáu loại template khác nhau, mỗi loại được gọi là một chế độ template:
  - HTML Template
  - XML Template
  - Text Template
  - Javascript Template
  - CSS Template
  - RAW Template

Ví dụ Thymeleaf HTML template

```
<table class="table table-striped table-hover table-bordered">
    <thead class="table-dark">
    <tr>
        <th>#</th>
        <th>Email</th>
        <th>Comment</th>
        <th>Created Date</th>
        <th>Actions </th>
    </tr>
    </thead>
    <tbody>
        <tr th:each = "comment, iterStat: ${comments}">
            <td th:text="${iterStat.count}"></td>
            <td th:text="${comment.email}"></td>
            <td th:text="${comment.content}"></td>
            <td th:text="${#temporals.format(comment.createdOn, 'dd MMM yyyy')}">
            <td>
                <a class="btn btn-danger" th:href="@{/admin/posts/comments/{comme
            </td>
        </tr>
    </tbody>
</table>
```

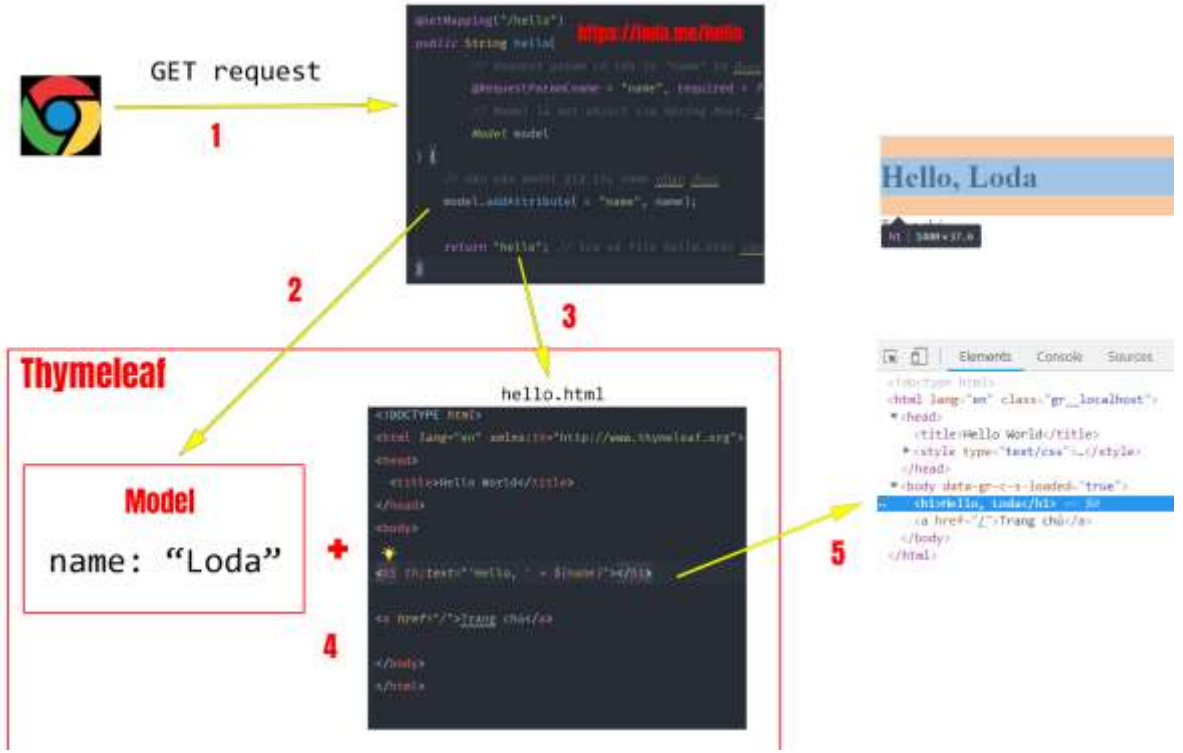
Mô hình Thymeleaf

Trong các ứng dụng web (Ví dụ: ứng dụng web Spring MVC), Thymeleaf được xử lý ở phía máy chủ và kết quả được đưa vào HTML và được trả về trình duyệt.



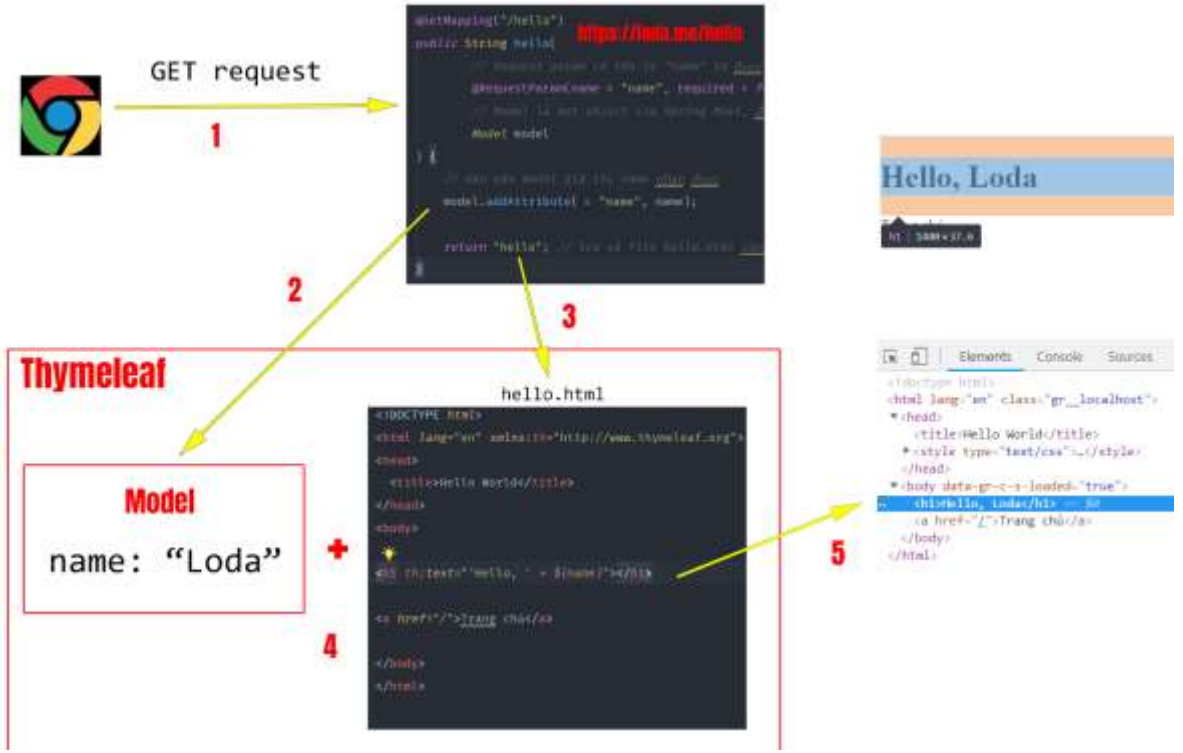
Cách hoạt động của Thymeleaf

Các file HMTL do Thymeleaf tạo ra là nhờ kết hợp dữ liệu và template + quy tắc để sinh ra một file HTML chứa đầy đủ thông tin.
Việc của bạn là cung cấp dữ liệu và quy định template như nào, còn việc dùng các thông tin đó để render ra HTML sẽ do Thymeleaf giải quyết.



Cách hoạt động của Thymeleaf

Các file HMTL do Thymeleaf tạo ra là nhờ kết hợp dữ liệu và template + quy tắc để sinh ra một file HTML chứa đầy đủ thông tin.
Việc của bạn là cung cấp dữ liệu và quy định template như nào, còn việc dùng các thông tin đó để render ra HTML sẽ do Thymeleaf giải quyết.



Cách hoạt động của Thymeleaf

Model

```java
public class StudentDto {
    private Long id;
    private String firstName;
    private String lastName;
    private String email;

    // getter/setter methods
}
```

Controller

```java
// handler method to handle list students request
@GetMapping({"/students"})
public String listStudents(Model model){
    List<StudentDto> students = studentService.getAllStudents();
    model.addAttribute( attributeName: "students", students);
    return "students";
}
```

Thymeleaf Template

```html
<!DOCTYPE html>
<html lang="en"
    xmlns:th="http://www.thymeleaf.org"
>
<head...>
<body>
    <table class="table table-striped table-hover table-bordered">
        <thead class="table-dark">
            <tr>
                <th>Student First Name</th>
                <th>Student Last Name</th>
                <th>Student Email</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="student : ${students}">
                <td th:text="${student.firstName}"></td>
                <td th:text="${student.lastName}"></td>
                <td th:text="${student.email}"></td>
            </tr>
        </tbody>
    </table>
</div>
</body>
</html>
```

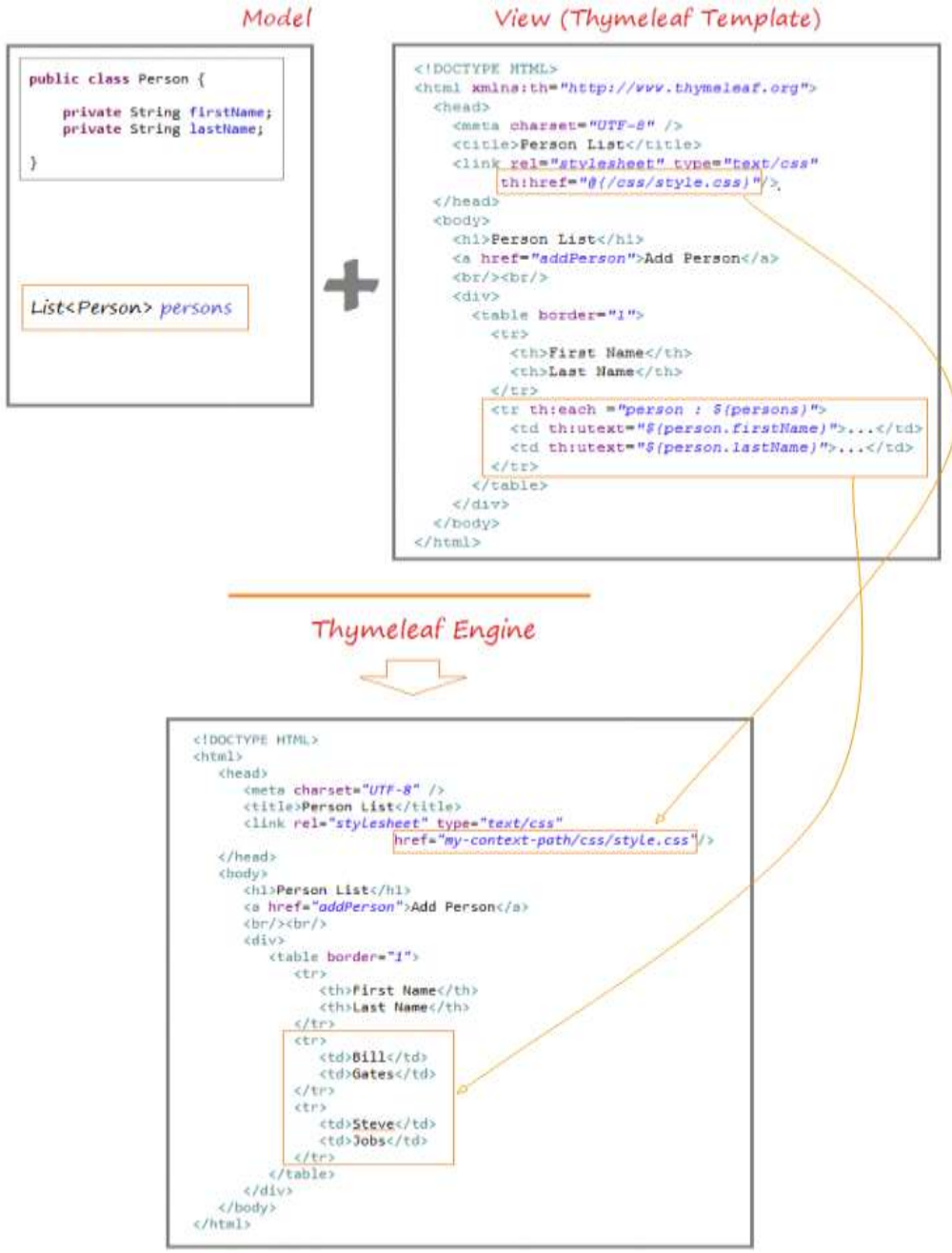
Output

localhost:8080/students

Student First Name	Student Last Name	Student Email
ram	jadhav	ram@gmail.com
vikram	jadhav	vikram@gmail.com
umesh	fadatare	umesh@gmail.com

Cách hoạt động của Thymeleaf

Thymeleaf Engine sẽ phân tích **Thymeleaf Template**, nó sử dụng các dữ liệu **Java** để thay thế các vị trí được đánh dấu trên **Thymeleaf Template** để tạo ra một văn bản mới.



Cài đặt thư viện Thymeleaf

Chúng ta sử dụng các dependency sau đây để sử dụng Thymeleaf:

- **spring-boot-starter-thymeleaf:** dependency cung cấp các thư viện cần thiết để sử dụng Thymeleaf trong project.
- **thymeleaf-extras-springsecurity5:** dependency cung cấp tích hợp Thymeleaf với Spring Security để thực hiện các chức năng bảo mật trong project.
- Thymeleaf template sử dụng ở **/resources/templates** folder

Cú pháp

Cú pháp của Thymeleaf sẽ là một attributes (Thuộc tính) của thẻ HTML và bắt đầu bằng chữ th:. Với cách tiếp cận này, bạn sẽ chỉ cần sử dụng các thẻ HTML cơ bản đã biết mà không cần bổ sung thêm syntax hay thẻ mới như JSP truyền thống.

Ví dụ

Để truyền dữ liệu từ biến *name* trong Java vào một thẻ *H1* của HTML.

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

Chúng ta viết thẻ H1 như bình thường, nhưng không chứa bất cứ text nào trong thẻ. Mà sử dụng cú pháp th:text="${name}" để **Thymeleaf** lấy thông tin từ biến *name* và đưa vào thẻ *H1*.

- Kết quả khi render ra:

// Giả sử String name = "loda"
<h1>Loda</h1>

Thuộc tính th:text biến mất và giá trị biến *name* được đưa vào trong thẻ *H1*. Đó là cách **Thymeleaf** hoạt động.

Định nghĩa Biến trong Thymeleaf

Trong Thymeleaf, bạn có thể định nghĩa biến bằng cách sử dụng cú pháp

- th:object
- th:field

Dưới đây là cách bạn có thể sử dụng chúng:

Biến đối tượng

Biến đối tượng cho phép bạn đặt một đối tượng Java vào phạm vi của Thymeleaf. Ví dụ, nếu bạn có một đối tượng *Person* với thuộc tính *name*, bạn có thể sử dụng *th:object* như sau:

```
<div th:object="${person}">
        <p>Name: <span th:text="*{name}"></span></p>
</div>
```

Ở đây, `${person}` đại diện cho đối tượng `Person`, và bạn có thể truy cập tính `name` của nó bằng cách sử dụng `*{name}`.

Biến trường

Biến trường làm cho việc xử lý các trường dữ liệu trong các biểu mẫu HTML trở nên dễ dàng hơn. Ví dụ, nếu bạn có một biểu mẫu đăng ký với trường *username*, bạn có thể sử dụng *th:field* như sau:

```
<form th:object="${user}" th:action="@{/register}" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" th:field="*{username}" />
</form>
```

Trong trường này, `${user}` đại diện cho đối tượng `User`, và `*{username}` là cách Thymeleaf liên kết trường `username` trong biểu mẫu HTML với thuộc tính `username` của đối tượng `User`.

Các loại biểu thức

- ${...}: biểu thức thay giá trị của biến vào template
- *{...}: biểu thức thay thuộc tính của biến vào template. Hay dùng với form post
- #{...}: message expression, biểu thức thay chuỗi đa ngôn ngữ từ file resource.
- @{...}: link expression, biểu thức liên kết
- ~{...}: fragement expression, biểu thức mảnh

Hiển thị giá trị của biến

Khi bạn đã định nghĩa biến trong Thymeleaf, bạn có thể hiển thị giá trị của chúng bằng cách sử dụng các thuộc tính Thymeleaf như `th:text`, `th:value`, `th:src`, và nhiều thuộc tính khác.

Dưới đây là một số ví dụ:

Hiển thị giá trị của biến

Hiển thị giá trị văn bản (th:text)

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

Trong ví dụ này, `th:text` sẽ hiển thị giá trị của thuộc tính `name` của đối tượng `user`.

Hiển thị giá trị của biến

Hiển thị giá trị vào trường (th:value)

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

`th:value` cho phép bạn gán giá trị của thuộc tính `email` vào trường nhập liệu.

Điều kiện và vòng lặp

Thymeleaf cung cấp cách mạnh mẽ để sử dụng biến trong các biểu thức điều kiện và vòng lặp. Bạn có thể kiểm tra điều kiện, duyệt qua danh sách và hiển thị các phần tử tùy thuộc vào giá trị của biến.

Ví dụ về Điều kiện

```
<p th:if="${user.isAdmin()}">This user is an admin.</p>
```

Trong ví dụ này, `th:if` sẽ kiểm tra xem đối tượng `user` có phải là admin hay không.

Vòng lặp

```
<ul> <li th:each="item : ${items}" th:text="${item}"></li> </ul>
```

- Với th:each, bạn có thể duyệt qua danh sách items và hiển thị mỗi phần tử trong danh sách.
- Biến trong Thymeleaf cho phép bạn tạo ra các trang web động và tương tác bằng cách truyền dữ liệu từ máy chủ Java của bạn vào giao diện người dùng. Bằng cách sử dụng cú pháp Thymeleaf và các thuộc tính như th:text, th:if, và th:each, bạn có thể hiển thị dữ liệu theo cách linh hoạt và dễ dàng. Điều này làm cho việc phát triển ứng dụng web trở nên dễ dàng và mạnh mẽ hơn bao giờ hết.

Form dữ liệu

Thymeleaf cung cấp 3 cú pháp để làm việc với Form:

1. th:action viết biểu thức đường dẫn mà Form sẽ gửi lên server. Nó khác thuộc tính action vốn có của Form ở chỗ, bạn có thể viết biểu thức, biến, so sánh, điều kiện...để động hoá đường dẫn.

2. th:object khai báo đối tượng chứa dữ liệu các trường để điền vào form.

3. th:field lấy dữ liệu trong từng thuộc tính của đối tượng đổ vào một trường text.

```
<form action="#" th:action="@{/bmi}" th:object="${bmiRequest}" method="post">
    <input type="text" placeholder="Your name" th:field="*{name}"/><br><br>
    <input type="text" placeholder="Your email" th:field="*{email}"/><br><br>
    <input type="text" placeholder="Your height" th:field="*{height}"/> (m)<br><br>
    <input type="text" placeholder="Your weight" th:field="*{weight}"/> (kg)<br><br>
    <button type="submit">Calculate BMI</button>
</form>
```

TRƯỜNG ĐẠI HỌC
VĂN LANG

KHOA CÔNG NGHỆ THÔNG TIN