

Daftar Isi

1. Pengertian Thymeleaf.....	2
2. Thymeleaf Dan Spring Boot.....	3
3. Atribut Dalam Thymeleaf	6
4. Expression Dalam Thymeleaf.....	7
1. Thymeleaf Variable Expression.....	7
2. Thymeleaf Link Expression	12
3. Thymeleaf Fragment Expression.....	13
4. Cara menambahkan CSS dan JS Kedalam Thymeleaf	15
5. Penggunaan If Else dan Switch Case	16
6. Perulangan Dalam Thymeleaf	21

Modul Thymeleaf

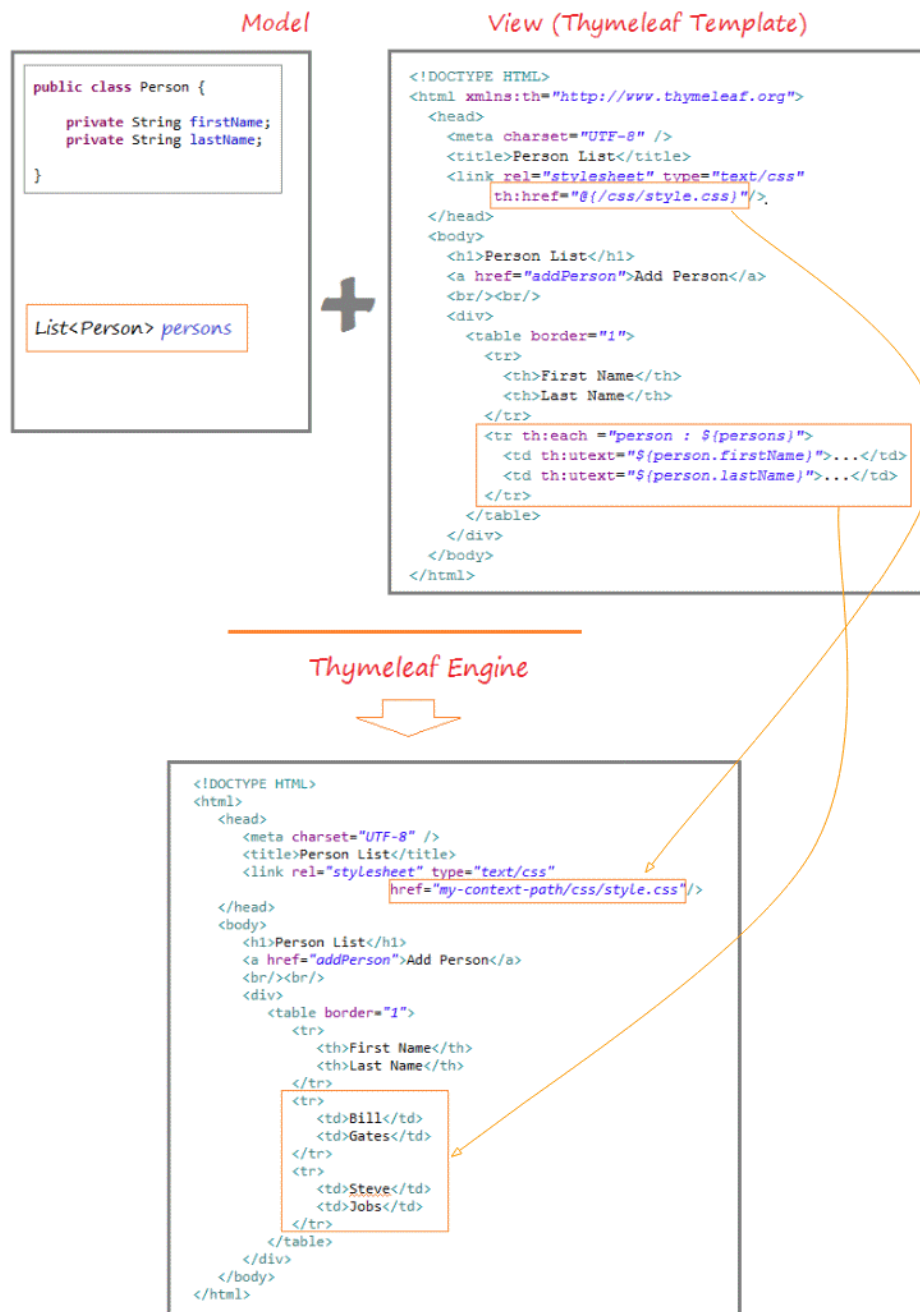
1. Pengertian Thymeleaf

Thymeleaf adalah mesin template Java XML/HTML/Javascript/CSS dan bahkan teks biasa yang dapat bekerja baik di lingkungan web maupun non-web. Namun lebih cocok untuk melayani XHTML/HTML5 pada lapisan tampilan aplikasi web berbasis MVC, tetapi dapat memproses file XML apa pun bahkan di lingkungan offline.

Berikut adalah halaman HTML dengan atribut dan ekspresi Thymeleaf:

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Home Page</title>
7     <link
8       href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css"
9       rel="stylesheet"
10      integrity="sha384-4bw+aeP/YC94hEpVNVgiZdgIC5+VKNBQNGCheKRQn+PtmoHDEXuppvndJzQIU9"
11      crossorigin="anonymous"
12    />
13   </head>
14   <body>
15     <div class="container my-2">
16       <h2>Cars List</h2>
17       <a th:href="@{/add}" class="btn btn-primary btn-sm mb-3">Add Car</a>
18       <table
19         class="table table-dark table-hover table-striped table-responsive-md rounded-3"
20       >
21         <thead class="text-center">
22           <tr>
23             <!-- <th>ID</th> -->
24             <th>Brand</th>
25             <th>Model</th>
26             <th>Description</th>
27             <th>Action</th>
28           </tr>
29         </thead>
30         <tbody>
31           <tr th:each="car : ${allCarList}" class="text-center">
32             <!-- <td th:text="${car.id}"></td> -->
33             <td th:text="${car.brand}"></td>
34             <td th:text="${car.model}"></td>
35             <td th:text="${car.description}"></td>
36             <td class="d-flex justify-content-evenly">
37               <a
38                 th:href="@{/updatecar/{id}(id=${car.id})}"
39                 class="btn btn-secondary"
40               >Update</a>
41               <a
42                 th:href="@{/delete/{id}(id=${car.id})}"
43                 class="btn btn-danger"
44               >Delete</a>
45             </td>
46           </tr>
47         </tbody>
48       </table>
49     </div>
50   </body>
51 </html>
```

Dibawah ini merupakan proses Thymeleaf Engine memproses Thymeleaf Template :



Dari gambar diatas dapat disimpulkan bahwa, dalam aplikasi web Spring MVC, Controller Spring MVC mengembalikan tampilan (Thymeleaf Template) sebagai hasilnya dan akan dirender lagi di browser.

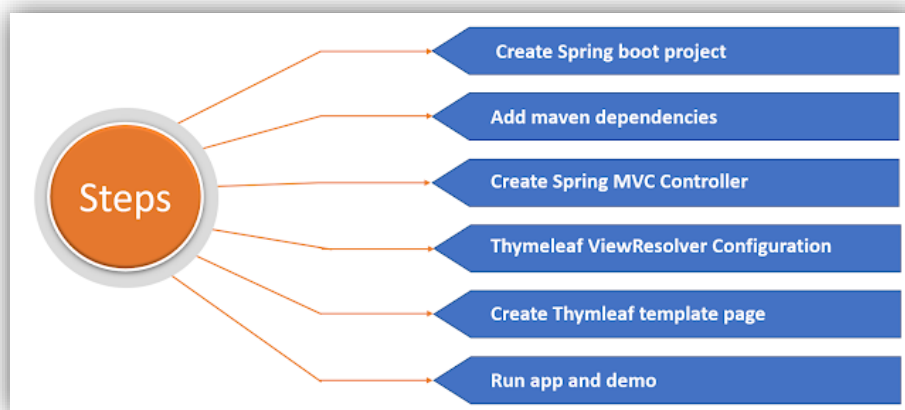
Selama kita belajar Thymeleaf, untuk semua program simpan dalam satu proyek sama agar kalian tidak perlu membuat proyek setiap pertemuannya. Jadi selama belajar Thymeleaf ini buatlah 1 proyek dengan nama domain `com.thymeleaf` dan nama proyek `getchstore`.

2. Thymeleaf Dan Spring Boot

Thymeleaf dan Spring Boot adalah dua teknologi yang dapat digunakan bersama-sama untuk membangun aplikasi web yang modern dan scalable. Thymeleaf adalah

template engine yang dapat digunakan untuk membuat halaman web statis atau dinamis. Thymeleaf mendukung HTML, XML, dan teks sebagai template. Thymeleaf juga mendukung berbagai macam fitur, seperti perulangan, perbandingan, dan ekspresi.

Spring Boot adalah framework Java yang dapat digunakan untuk membangun aplikasi web dan layanan web. Spring Boot mengabstraksi kerumitan konfigurasi Spring Framework, sehingga memudahkan pengembang untuk membangun aplikasi web yang kompleks dengan cepat dan mudah.



Ini adalah gambaran suatu proses dalam pengembangan aplikasi berbasis web dengan menggunakan Spring Boot dan Thymeleaf. Untuk menggunakan Thymeleaf dan Spring Boot bersama-sama, kita perlu menambahkan dependensi Thymeleaf ke `pom.xml` project kita.

Berikut adalah contoh `pom.xml` yang menambahkan dependensi Thymeleaf:

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

Untuk mencoba dalam pengaplikasian antara Spring Boot dan Thymeleaf, pertama terlebih dahulu buat package atau folder `model` dalam dalam proyek yang sudah dibuat sebelumnya. Kemudian di dalam package `controller` ini buat file `HelloWorldController.java` dan untuk isi file nya seperti dibawah ini :

```
package com.thymeleaf.getchstore.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
```

```

@Controller
public class HelloWorldController {
    @GetMapping("hello")
    public String hello(Model model) {
        model.addAttribute("message", "Hello World !!!");
        return "helloworld";
    }
}

```

Penjelasan dari program diatas :

HelloWorldController.java :

Annotation `@Controller` adalah annotation yang digunakan untuk menandai kelas sebagai controller Spring Boot. `@Controller` adalah kelas yang bertanggung jawab untuk menangani permintaan HTTP. `Model` digunakan untuk menyimpan data yang akan ditampilkan di halaman web. `@GetMapping` digunakan untuk mendefinisikan metode yang akan dijalankan ketika klien HTTP melakukan permintaan GET ke URL `/hello`.

Sedangkan untuk kode `model.addAttribute("message", "Hello World !!!");` artinya untuk menambahkan sebuah data kedalam model. Data yang ditambahkan adalah `message` dengan isi atau value `"Hello World !!!"`. Dan untuk `return "helloworld";` digunakan untuk mengembalikan nama file template Thymeleaf yang akan ditampilkan. File template Thymeleaf tersebut memiliki nama `helloworld.html`. File template Thymeleaf `helloworld.html` dapat berisi kode HTML biasa dan kode Thymeleaf. Kode Thymeleaf digunakan untuk menampilkan data yang telah ditambahkan ke `model`.

Kemudian buat file html di bagian package `resource.templates`, beri nama file html ini dengan string yang di return pada method `String hello` yang sudah di buat sebelumnya yaitu `helloworld.html`. Untuk isi file `helloworld.html` sebagai berikut :

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>Spring Boot With Thymeleaf</title>
  </head>
  <body>
    <h1 th:text="${message}"></h1>
  </body>
</html>

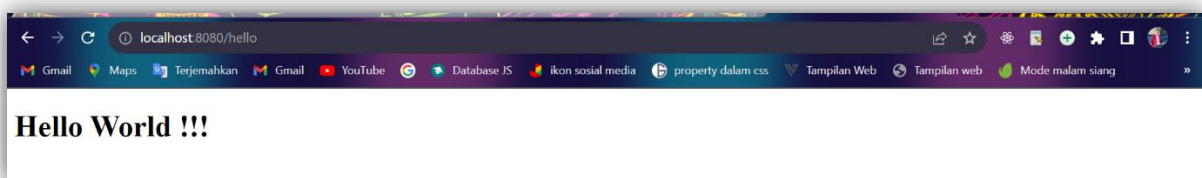
```

Penjelasan dari program diatas :

Atribut `th:text` digunakan untuk menampilkan nilai dari variabel `message`. Variabel `message` ini telah ditambahkan ke `model` oleh controller. Jadi, ketika HTTP melakukan permintaan GET ke URL `/hello`, Spring Boot akan menjalankan metode `hello()` dari controller `HelloWorldController`.

Metode `hello()` akan menambahkan data ke `model` dan mengembalikan nama file template Thymeleaf. Spring Boot akan merender file template Thymeleaf tersebut dan menampilkannya di browser. Pada halaman web yang ditampilkan, akan muncul pesan `"Hello World !!!"`.

Hasilnya:



3. Atribut Dalam Thymeleaf

Atribut dalam Thymeleaf adalah bagian dari tag HTML yang digunakan untuk mengontrol tampilan halaman web. Atribut ini dapat digunakan untuk menampilkan nilai dari variabel, menjalankan JavaScript, atau menentukan kondisi untuk menampilkan elemen HTML.

Thymeleaf memiliki berbagai macam atribut yang dapat digunakan. Berikut adalah beberapa atribut yang paling umum digunakan:

- Atribut ekspresi : Atribut ini digunakan untuk menampilkan nilai dari variabel atau ekspresi.
- Atribut kondisional : Atribut ini digunakan untuk menampilkan elemen HTML jika kondisi terpenuhi.
- Atribut iterasi : Atribut ini digunakan untuk mengulang elemen HTML berdasarkan data yang ada di variabel.
- Atribut aksi : Atribut ini digunakan untuk menjalankan JavaScript atau tindakan lainnya.

Adapun berikut ini atribut yang ada di dalam thymeleaf diantaranya `yaitu` `th:action`, `th:attr`, `th:block`, `th:case`, `th:classappend`, `th:each`, `th:field`, `th:fragment`,

`th:href`, `th:if`, `th:include`, `th:insert`, `th:object`, `th:remove`, `th:replace`, `th:src`, `th:style`, `th:swicth`, `th:text`, `th: unless`, `th:utext`, `th:value` dan `th:with`.

4. Expression Dalam Thymeleaf

Dalam thymeleaf terdapat beberapa atribut yang dapat digunakan diantaranya :

1. Thymeleaf Variable Expression

Atribut `th:text` berguna untuk menampilkan teks yang dihasilkan atau didapatkan dari ekspresi yang ada di dalamnya.

- Contoh 1 : Atribut Thymeleaf dengan Variable Expressions :

Fitur ini memungkinkan untuk membuat tampilan HTML yang dinamis dan sesuai dengan data yang berasal dari model Java atau dari lingkup tampilan Thymeleaf. Kita dapat menampilkan data, menghitung nilai, atau melakukan operasi lainnya dalam tampilan menggunakan variable expressions.

Untuk membuat contoh penggunaan variable expressions ini, pertama buat terlebih dahulu package `model` kemudian di dalam package `model` buat file dengan nama `Laptop.java` dengan isi file sebagai berikut :

```
package com.thymeleaf.getchstore.model;

public class Laptop {
    private String id;
    private String merk;
    private String tipe;
    private String harga;

    public Laptop() {
    }

    public Laptop(String id, String merk, String tipe, String harga) {
        this.id = id;
        this.merk = merk;
        this.tipe = tipe;
        this.harga = harga;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getMerk() {
```

```

        return merk;
    }

    public void setMerk(String merk) {
        this.merk = merk;
    }

    public String getTipe() {
        return tipe;
    }

    public void setTipe(String tipe) {
        this.tipe = tipe;
    }

    public String getHarga() {
        return harga;
    }

    public void setHarga(String harga) {
        this.harga = harga;
    }
}

```

Kemudian buat file `controller` dari `model` yang sudah dibuat diatas nama file `LaptopController.java` dan untuk isi file adalah sebagai berikut :

```

package com.thymeleaf.getchstore.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import com.thymeleaf.getchstore.model.Laptop;

@Controller
public class LaptopController {
    @GetMapping("data-laptop")
    public String dataLaptop(Model model) {
        Laptop laptop = new Laptop("1", "Asus", "ROG Zephyrus G14",
20000000);
        model.addAttribute("dataLaptop", laptop);
        return "laptop";
    }
}

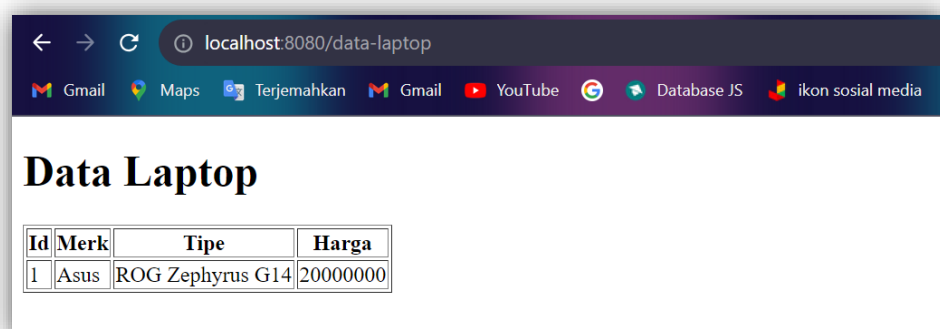
```


Yang terakhir buatlah file html untuk menampilkan data yang sudah dibuat diatas. Nama file html harus sama dengan string yang di return di LaptopController yaitu `laptop.html`. Untuk isi file adalah sebagai berikut :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>Data Laptop</title>
  </head>
  <body>
    <h1>Data Laptop</h1>
    <table border="1">
      <tr>
        <th>Id</th>
        <th>Merk</th>
        <th>Tipe</th>
        <th>Harga</th>
      </tr>
      <tr>
        <td th:text="${dataLaptop.id}"></td>
        <td th:text="${dataLaptop.merk}"></td>
        <td th:text="${dataLaptop.tipe}"></td>
        <td th:text="${dataLaptop.harga}"></td>
      </tr>
    </table>
  </body>
</html>
```

Atribut `th:text` akan mengisi content dari suatu tag atau mengganti content tag apabila sebelumnya sudah terdapat konten. Dalam contoh diatas, diartikan bahwa atribut `th:text` akan mengisi setiap tag `<td>` dengan data dari masing-masing data dari variable `dataLaptop`. Untuk mengetahui hasilnya bisa langsung mengakses link <http://localhost:8080/data-laptop>.

Hasilnya :



Id	Merk	Tipe	Harga
1	Asus	ROG Zephyrus G14	20000000

- Contoh 2 : Atribut Thymeleaf `th:text` dengan Selection Expressions :

Selection expressions adalah jenis expressions yang digunakan dalam Thymeleaf untuk menampilkan nilai dari atribut atau properti dari objek. Sintaks selection expressions adalah `*{attribute}`, di mana attribute adalah nama atribut atau properti. Selection expressions dapat digunakan untuk menampilkan nilai dari objek yang telah ditambahkan ke model oleh controller. Objek tersebut dapat berupa objek sederhana, seperti objek string, objek integer, atau objek boolean.

Untuk mengetahui bagaimana penggunaan selection expressions, tambahkan kode program dibawah ini kedalam file `LaptopController.java` dan `laptop.html`.

LaptopController.java :

```
Laptop laptop2 = new Laptop("2", "Asus", "Nitro 5", 1500000);  
model.addAttribute("dataLaptop2", laptop2);
```

laptop.html :

```
<h1>Selection Expressions Demo</h1>  
<table border="1">  
  <tr>  
    <th>Id</th>  
    <th>Merk</th>  
    <th>Tipe</th>  
    <th>Harga</th>  
  </tr>  
  <tr th:object="${dataLaptop2}">  
    <td th:text="*{id}"></td>  
    <td th:text="*{merk}"></td>  
    <td th:text="*{tipe}"></td>  
    <td th:text="*{harga}"></td>  
  </tr>  
</table>
```

Pada file `LaptopController.java` cukup menambahkan data laptop. Untuk di file `laptop.html`, pada tag `tr` terdapat `th:object="${dataLaptop2}"` digunakan untuk pemanggilan sebuah objek dari controller dengan nama variable `dataLaptop2`, dan `th:text="*{id}"` untuk menampilkan nilai properti `id` dari objek laptop, dan seterusnya.

Hasilnya :



- Contoh 3 : Atribut Thymeleaf `th:text` dengan Message Expressions

Message Expressions adalah jenis expressions yang digunakan dalam Thymeleaf untuk menampilkan pesan dari file properties. Sintaks Message Expressions adalah `#{key}`, di mana key adalah nama kunci dari pesan yang akan ditampilkan.

Pesan yang akan ditampilkan disimpan dalam file properties. File properties adalah file teks yang berisi pasangan key-value. Key adalah nama kunci dari pesan, sedangkan value adalah nilai dari pesan.

Untuk mengetahui bagaimana penggunaan message expressions, buat terlebih dahulu file `messages.properties`, di dalam folder `/resources` dan tambahkan kode dibawah ini :

```
app.name=Getch Store
welcome.message=Selamat Datang di Aplkasi Getch Store
```

Dalam file `LaptopController.java` tambahkan kode program seperti dibawah ini :

```
@GetMapping("message-expression")
public String messageExpression() {
    return "message-expression";
}
```

Yang terakhir buatlah file `message-expression.html` dan masukan kode program dibawah ini :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8" />
    <title>Message Expressions</title>
</head>
<body>
    <h2 th:text="#{app.name}"></h2>
```

```
<h2 th:text="#{welcome.message}"></h2>
</body>
</html>
```

Hasilnya :



Dalam contoh di atas, `th:text` digunakan dengan message expressions `#{app.name}` dan `#{welcome.message}` untuk menampilkan teks "Getch Store" dan "Selamat Datang di Aplikasi Getch Store" dengan pesan lokal yang sesuai dengan kunci `"app.name"` dan `"welcome.message"` yang telah didefinisikan dalam `messages.properties`.

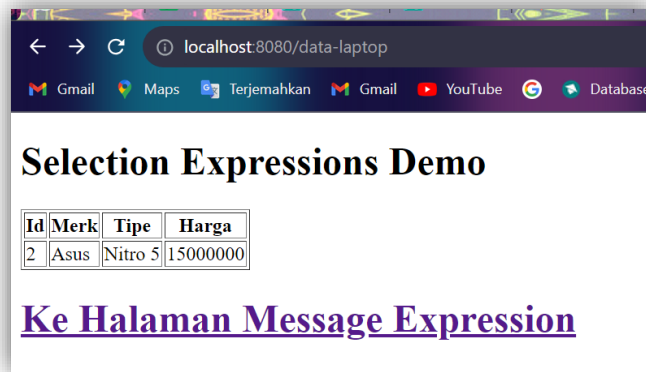
2. Thymeleaf Link Expression

Thymeleaf Link Expression adalah jenis expressions yang digunakan dalam Thymeleaf untuk membuat tautan ke view atau resource lainnya. Sintaks Thymeleaf Link Expression adalah `@{url}`, di mana `url` adalah URL dari view atau resource yang akan ditautkan. Thymeleaf Link Expression dapat digunakan untuk membuat tautan ke view atau resource lainnya.

Agar lebih jelas, maka langsung saja kepada contoh penerapan dari Thymeleaf Link Expression ini. Pertama, cukup tambahkan kode berikut di dalam file `laptop.html` :

```
<h1>
  <a th:href="@{message-expression}">Ke Halaman Message
  Expression</a>
</h1>
```

Nantinya tampilan file `laptop.html` akan seperti dibawah ini :



Jadi apabila kita klik `Ke Halaman Message Expression` maka akan otomatis ke link ke file `message-expression.html` yang isinya ada 2 pesan tadi yang sudah dibuat sebelumnya.

3. Thymeleaf Fragment Expression

Thymeleaf Fragment Expression adalah fitur dalam Thymeleaf yang memungkinkan kita untuk membagi tampilan HTML menjadi fragmen-fragmen kecil yang dapat digunakan ulang di berbagai bagian halaman web. Dengan kata lain, kita dapat membuat dan menggunakan potongan kode HTML yang sering digunakan di beberapa tempat tanpa harus mengulanginya.

Sintaks fragment expression adalah `th:fragment="fragment_name"`, di mana `fragment_name` adalah nama dari fragment yang akan dirujuk.

Contoh penggunaan Thymeleaf Fragment Expression adalah sebagai berikut :

Pertama buatlah sebuah folder `fragments` dalam folder `templates`, setelah itu dalam folder `fragments` buat 1 file html ialah `component.html` dengan isi kode program sebagai berikut :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Header Fragment</title>
  </head>
  <body>
    <header th:fragment="header">
      <h1>Ini adalah Header</h1>
    </header>
    <footer th:fragment="footer">
      <h1>Ini adalah Footer</h1>
    </footer>
  </body>
</html>
```

Kedua, buatlah file `fragment-expression.html` dalam folder `templates` dan masukan kode program dibawah ini :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8" />
    <title>Fragment Expressions</title>
  </head>
  <body>
    <header th:replace="~{fragments/component :: header}"></header>
    <hr />
    <div th:fragment="section">
      <h1>Ini adalah fragment section</h1>
    </div>
    <hr />
    <footer th:insert="~{fragments/component :: footer}"></footer>
  </body>
</html>
```

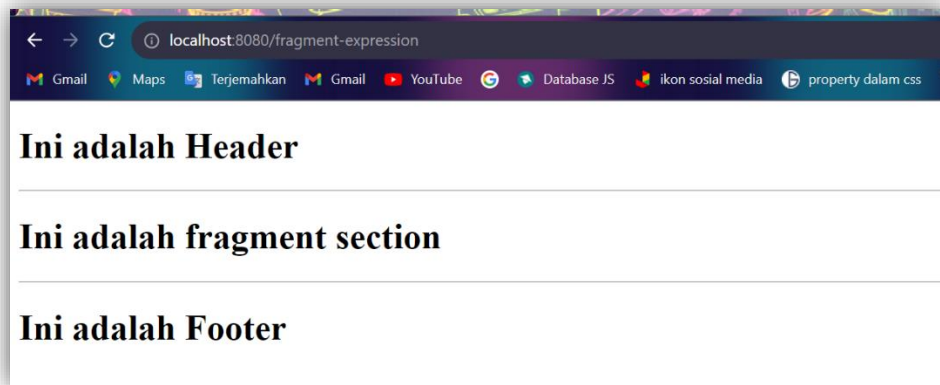
Yang terakhir sebelum menjalankan program fragment expression ini buatlah method yang nantinya merujuk pada file `fragment-expression.html` di dalam file `LaptopController.java`, cukup masukan kode berikut :

```
@GetMapping("fragment-expression")
public String fragmentExpression() {
    return "fragment-expression";
}
```

Untuk lebih jelas dalam penggunaan fragment expression ini, kalian boleh akses link dibawah ini :

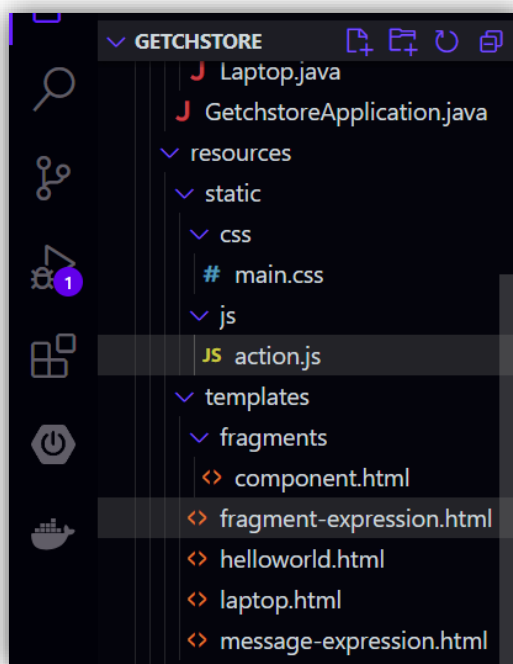
<https://attacomisian-com.translate.goog/blog/thymeleaf-fragments? x tr sl=en& x tr tl=id& x tr hl=id& x tr pto=tc#thinsert--threplace-attributes>

Hasilnya :



4. Cara menambahkan CSS dan JS Kedalam Thymeleaf

Pertama buatlah terlebih dahulu sebuah folder `css` dan `js` dalam folder `static`, kemudian dalam folder `css` buatlah file `main.css` dan di folder `js` buatlah file `actions.js`. Jadi untuk gambaran struktur folder nya seperti dibawah ini :



Setelah membuat file `main.css` dan `actions.js`, selanjutnya ialah mengisi `main.css` dengan css yang diinginkan, sedangkan untuk `actions.js` bisa diisi javascript sesuai selera kalian masing-masing. Kali ini file `laptop.html` yang akan diberi css dan js. Untuk isi css dan js nya sebagai berikut :

main.css :

```
h1 a {  
  color: red;  
}
```

```
table tr td {
  padding: 16px;
  color: blue;
}
```

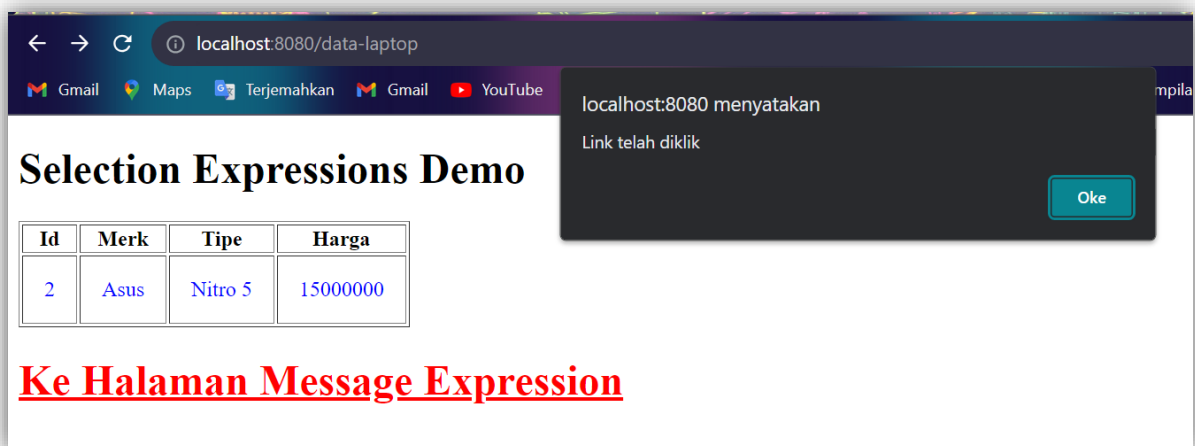
action.js :

```
function showAlert() {
  alert("Link telah diklik");
}
```

Kemudian dibagian head tambahkan kode berikut agar css dan js bisa dipakai di file `laptop.html` :

```
<link rel="stylesheet" th:href="@{/css/main.css}" />
<script th:src="@{/js/action.js}"></script>
```

Dan di file `laptop.html` di tag `a` yang isi konten nya `Ke Halaman Message Expression` tambahkan atribut `th:onclick="showAlert()"` dan hasilnya apabila dijalankan akan seperti dibawah ini :



5. Penggunaan If Else dan Switch Case

Thymeleaf mendukung penggunaan if else dan switch case untuk membuat logika kondisional dalam template. Untuk menggunakan if else dalam Thymeleaf, kita dapat menggunakan tag `th:if` dan `th:unless`. Tag `th:if` akan menampilkan kontennya jika

kondisinya benar, sedangkan tag `th:unless` akan menampilkan kontennya jika kondisinya salah.

Contoh dasar sintak penggunaan if else :

```
<div th:if="${condition}">
  <p>TRUE</p>
</div>
<div th:unless="${condition}">
  <p>FALSE</p>
</div>
```

Di mana `condition` adalah expression yang akan diperiksa. Jika kondisi tersebut benar, maka blok kode di dalam tag `th:if` akan dijalankan. Jika kondisi tersebut salah, maka blok kode di dalam tag `th:unless` akan dijalankan.

Untuk penerapannya dalam suatu program, pertama buat file `User.java` dalam folder `models` dan masukan kode program dibawah ini :

```
package com.thymeleaf.getchstore.models;

public class User {
    private int id;
    private String name;
    private int age;
    private String role;

    public User() {
    }

    public User(int id, String name, int age, String role) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.role = role;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
}
```

```

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }
}

```

Selanjutnya buatlah file `UserController.java` dalam folder `controllers` dan masukan kode program berikut ini :

```

package com.thymeleaf.getchstore.controllers;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import com.thymeleaf.getchstore.models.User;

@Controller
public class UserController {
    @GetMapping("users")
    public String listUsers(Model model) {
        List<User> users = new ArrayList<>();
        users.add(new User(1, "WAHYU", 20, "ADMIN"));
        users.add(new User(2, "JEBREDZ", 22, "USER"));
        users.add(new User(3, "WABREDZ", 21, "ADMIN"));
        model.addAttribute("listUsers", users);
        return "user";
    }
}

```

Dan yang terakhir ialah membuat file html untuk menampilkan data yang sudah dibuat sebelumnya. Jadi buatlah file `user.html` dan masukan kode program berikut ini :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>User</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEVDwwyk2MPK8M2HN"
      crossorigin="anonymous"
    />
  </head>
  <body>
    <table class="table table-striped table-hover w-50 m-4 caption-top">
      <caption>
        List User
      </caption>
      <thead class="table-dark">
        <tr>
          <th scope="col">ID</th>
          <th scope="col">Name</th>
          <th scope="col">Age</th>
          <th scope="col">Role</th>
          <th scope="col">Action</th>
        </tr>
      </thead>
      <tbody class="table-group-divider">
        <tr th:each="user : ${listUsers}">
          <td th:text="${user.id}"></td>
          <td th:text="${user.name}"></td>
          <td th:text="${user.age}"></td>
          <td th:text="${user.role}"></td>
          <td>
            <a class="btn btn-primary" th:if="${user.role} == 'ADMIN'"
              >Update</a>
            <a class="btn btn-danger" th:if="${user.role} ==
'ADMIN'">Delete</a>
            <a class="btn btn-primary" th:unless="${user.role} == 'ADMIN'"
              >View</a>
          </td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

```
</table>
</body>
</html>
```

Pada program diatas terdapat atribut `th:if="{user.role} == 'ADMIN'"` yang mana artinya ialah jika kondisi role dari suatu user "ADMIN" maka akan menampilkan button update dan button delete. Dan selain atribut `th:if="{user.role} == 'ADMIN'"`, adapun atribut `th:unless="{user.role} == 'ADMIN'"` yang artinya jika role dari user bukan "ADMIN" maka hanya akan menampilkan button view saja.

Nah sekarang coba masih menggunakan program diatas cuman penggunaannya kita ganti yang tadinya If Else menjadi Switch Case. Cukup ganti kode program yang ada di dalam tag table menjadi seperti dibawah ini :

```
<table class="table table-striped table-hover w-50 m-4 caption-top">
  <caption>
    List User
  </caption>
  <thead class="table-dark">
    <tr>
      <th scope="col">ID</th>
      <th scope="col">Name</th>
      <th scope="col">Age</th>
      <th scope="col">Role</th>
      <th scope="col">Greeting</th>
    </tr>
  </thead>
  <tbody class="table-group-divider">
    <tr th:each="user : {listUsers}">
      <td th:text="{user.id}"></td>
      <td th:text="{user.name}"></td>
      <td th:text="{user.age}"></td>
      <td th:text="{user.role}"></td>
      <td th:switch="{user.role}">
        <p th:case="ADMIN">Hallo Admin</p>
        <p th:case="USER">Hallo User</p>
        <p th:case="*">Hallo Guest</p>
      </td>
    </tr>
  </tbody>
</table>
```

Hasilnya :

ID	Name	Age	Role	Greeting
1	WAHYU	20	ADMIN	Hallo Admin
2	JEBREDZ	22	USER	Hallo User
3	WABREDZ	21	ADMIN	Hallo Admin

6. Perulangan Dalam Thymeleaf

Di Thymeleaf, perulangan menggunakan atribut `th:each`. Thymeleaf `th:each` memungkinkan untuk mengulang sebuah variabel. Perulangan ini memiliki dua parameter, yaitu:

- `item` : Variabel yang akan digunakan untuk menyimpan data dari setiap perulangan.
- `collection` : Variabel yang berisi data yang akan diulang.

Sintaks dasar perulangan `th:each` :

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

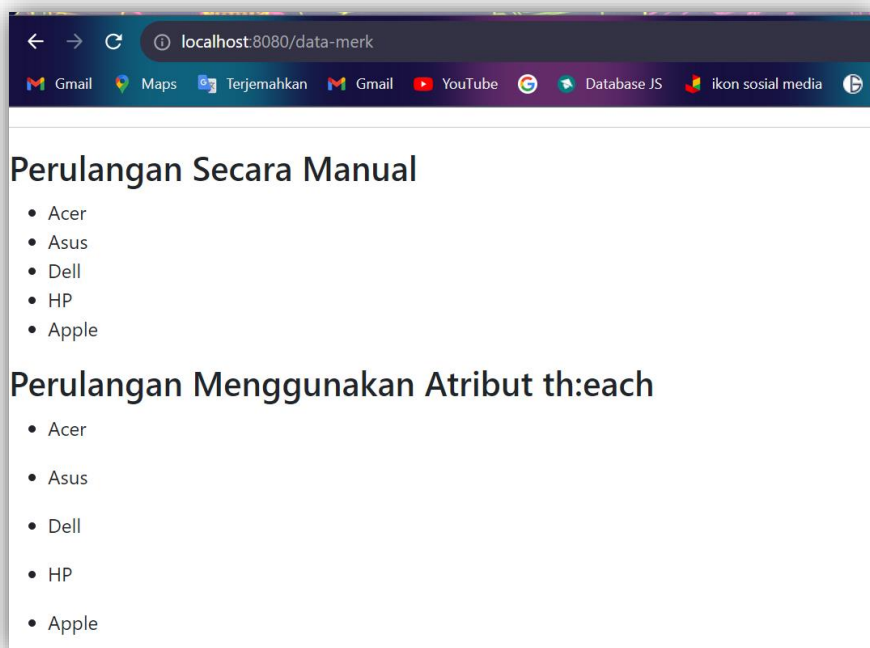
Contoh lengkap penggunaan `th:each` untuk melakukan perulangan sebuah variabel, dalam contoh ini kita menggunakan sebuah array biasa. Pertama buat terlebih dahulu sebuah method dalam file `LaptopController.java`, nama method `dataMerk` serta untuk endpoint nya gunakan `"data-merk"`. Untuk kode lengkap dari method `dataMerk` adalah sebagai berikut :

```
@GetMapping("data-merk")
public String dataMerk(Model model) {
    String[] merk = { "Acer", "Asus", "Dell", "HP", "Lenovo", "Apple", "MSI",
    "Razer", "Microsoft", "Toshiba" };
    model.addAttribute("merk", merk);
    return "laptop";
}
```

Kemudian dalam file `laptop.html` tambahkan kode dibawah ini :

```
<!-- Perulangan Manual -->
<h3>Perulangan Secara Manual</h3>
<ul>
  <li th:text="{merk[0]}"></li>
  <li th:text="{merk[1]}"></li>
  <li th:text="{merk[2]}"></li>
  <li th:text="{merk[3]}"></li>
  <li th:text="{merk[4]}"></li>
</ul>
<!-- Perulangan Menggunakan Atribut th:each -->
<h3>Perulangan Menggunakan Atribut th:each</h3>
<ul th:each="m : {merk}">
  <li th:text="{m}"></li>
</ul>
```

Hasilnya :



Sebelumnya sudah dijelaskan untuk mengulangi sebuah kumpulan objek menggunakan Thymeleaf melalui atribut `th:each`, dan untuk melakukan perulangan ini bukan hanya sebuah objek saja namun seperti list atau array tetap bisa.