

# Presentation Layer using Thymeleaf

v.1.3

## PENJELASAN

Hello anak-anak Papa APAP! Sebelumnya kalian telah mempelajari MVC (Model-View-Controller) dengan menggunakan proyek Emsidi. Pada tutorial kali ini, kalian akan mempelajari lebih dalam role view menggunakan Thymeleaf. Thymeleaf merupakan Java Library berupa XML/XHTML/HTML template engine. Thymeleaf digunakan untuk melakukan transformasi dari file XML/XHTML/HTML template menjadi sebuah file XML/XHTML/HTML yang utuh untuk ditampilkan. Jika dibandingkan dengan template engine pada framework lain, Thymeleaf layaknya Blade di Laravel, Jade di Node.js, Jinja2 di Python, Django Template Engine di Python, dll. Tujuan atau fungsi utama dari Thymeleaf yaitu untuk memberikan fasilitas bagi developer untuk membuat template dengan elega dan rapi sehingga developer dapat menggunakan template berulang kali secara dinamis, reusable.

Ciri utama jika Anda menggunakan Thymeleaf adalah adanya definisi XML Namespace ke link <http://www.thymeleaf.org> pada tag HTML.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
```

Anda WAJIB menambahkan definisi XML Namespace diatas jika ingin menggunakan Thymeleaf. Kemudian baru Anda bisa menggunakan attributes seperti th:text. Anda dapat merujuk pada dokumentasi Thymeleaf pada tautan [Documentation - Thymeleaf 3.0 + Spring](#) untuk dokumentasi penggunaan yang lebih lengkap.

## PRASYARAT

- DDP2
  - Interface
  - Overriding
- PPW
  - Request Method
  - MVC Framework
- Basis Data
  - Relationship

## LINGKUP PEMBAHASAN

- Controller
  - RequestMapping
- Templates
  - Thymeleaf

## EKSPEKTASI

- Memahami konsep view pada Spring Boot
- Memahami apa itu Thymeleaf dan penggunaannya

## PENGUMPULAN

- Tanggal Waktu Pengumpulan (*Due Date*) :  
**Kamis, 30 September 2021, sebelum 23.55 WIB untuk semua kelas. Setiap keterlambatan selama 24 jam akan mengakibatkan -20% dari nilai sebenarnya.**
- Pengumpulan tutorial dalam bentuk **Pull Request (PR)** dari **branch feat/tutorial-4-emsidi ke master**. Waktu keterlambatan dilihat dari **aktivitas perubahan** di Pull Request.
- Jawablah **semua** pertanyaan dan kerjakan **semua** latihan yang ada pada tutorial.
- Jawablah pertanyaan-pertanyaan yang ada pada file **README.md** yang **sama** seperti tutorial sebelumnya.
- Tutorial 4 di-*push* ke dalam *repository* yang sama dengan tutorial sebelumnya sehingga pada *repository* akan terdapat tiga folder yaitu *isPalindrome*, *kebunsafari* dan *emsidi*.
- Gunakan teknik *branching* dan PR yang sama seperti pada tutorial sebelumnya.
- Tutorial 4 **ada demo**, silahkan isi slot demo di SCELE.

## PENILAIAN

$\text{Nilai akhir} = \text{Nilai Tutorial} + \text{Latihan} + \text{Pertanyaan} + \text{Demo}$
---

**Plagiarisme akan dikenakan sanksi nilai 0! ☹**

- Indikator Penilaian Tutorial
  - Kesesuaian Model, Controller, Repository, dan View Templates.
  - Keberhasilan Proyek Emsidi.
  - Kelengkapan dan Ketepatan Jawaban pada README.md
- Demo

## PRA-TUTORIAL

1. Pastikan kamu sudah melakukan **merge** dari branch **tutorial sebelumnya ke master**

2. Sebelum mulai mengerjakan tutorial ini, buat **issue "Pengerjaan Tutorial 4"** pada Repo GitHub kamu
3. Pastikan pada lokal kamu sudah pindah ke **branch master**, kemudian **pull master**
4. Buat **branch baru** dari **master**, dengan nama **'feat/tutorial-4-emsidi'**

## PASCA-TUTORIAL

1. Pull Request dengan meminta *review* dari asisten melalui GitHub sebelum ***Deadline***
2. Pastikan pada saat Pull Request, **tambahkan issue** yang telah dibuat pada **Linked Issue** atau dapat mengikuti tutorial pada link [berikut](#)
3. Isi slot demo sesuai dengan pembagian asisten minggu ini. Slot akan dibuka jam **20.00 WIB**.
4. Asisten akan melakukan *approve* Pull Request setelah demo dilaksanakan

## TUTORIAL

Pada tutorial 4, Anda sudah menggunakan expression iterasi pada salah satu halaman html. Lihatlah file view-cabang.html, terdapat baris-baris kode seperti berikut:

\*implementasi bisa jadi berbeda, tetapi akan tetap menggunakan expression iterasi

### 1. Ubah tampilan pada view-cabang untuk bagian daftar pegawai menjadi seperti berikut

```
<table class="table">
  <thead>
    <tr>
      <th>No</th>
      <th>No Pegawai</th>
      <th>Nama</th>
      <th>Jenis Kelamin</th>
      <th></th>
      <th>Hapus</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="pegawai, iterationStatus : ${listPegawai}"
      th:style="${iterationStatus.even} ? 'font-weight:bold;'">
      <td th:text="${iterationStatus.count}"></td>
      <td th:text="${pegawai.noPegawai}"></td>
      <td th:text="${pegawai.namaPegawai}"></td>
      <td th:if="${pegawai.jenisKelamin} == 0">Laki-Laki</td>
      <td th:if="${pegawai.jenisKelamin} == 1">Perempuan</td>
      <td>
        <a class="btn btn-sm btn-primary" th:href="@{/pegawai/update/} + ${pegawai.noPegawai}">Update</a>
      </td>
      <td>
        <a class="btn btn-sm btn-primary" th:href="@{/pegawai/delete/} + ${pegawai.noPegawai}">Delete</a>
      </td>
    </tr>
  </tbody>
</table>
```

Expression `iterationStatus` berada pada atribut `th:each` berfungsi untuk melakukan iterasi dari kumpulan data pada suatu objek yang berada di salah satu atribut yang dimiliki suatu kelas Model. Dari contoh di atas, melakukan iterasi pada `*{menuList}`. Setelah atribut `th:each`, terdapat menu yang merupakan property berisi data pada index ke-n sesuai dengan iterasi saat ini. Penamaan property tersebut bebas, tidak bergantung pada atribut apapun. Selain menyediakan index, `iterationStatus` juga dapat membedakan index yang ganjil dan genap. Berikut beberapa attributes yang ada pada

`iterationStatus` dan contoh implementasi ganjil genap untuk implementasi suatu style:

- index: index iterasi, dimulai dari 0 (nol)
- count: banyaknya element yang telah diproses
- size: jumlah total element yang ada pada list
- even/odd: cek index, apakah genap atau ganjil

- first: cek apakah element yang sedang diproses merupakan element pertama pada list
- last: cek apakah element yang sedang diproses merupakan element terakhir pada list

## Penggunaan Conditional Expression pada Thymeleaf

Ketika ingin menampilkan sesuatu hal dengan kondisi tertentu, Thymeleaf menyediakan conditional expression, salah satunya berupa `th:if=${...}` dan `th:unless=${...}`. Selain dengan `th:if`, operasi conditional juga dapat dilakukan untuk hampir seluruh expression thymeleaf, misalnya seperti `th:style` pada contoh sebelumnya.

Dengan `th:if`, kita dapat dengan mudah menampilkan suatu tampilan dengan kondisi tertentu. Berikut ini contoh penggunaan `th:if` dan `th:unless` yang telah kita implementasikan pada `view-cabang.html`

```
<div th:if="*{listPegawai.size() != 0}">
  <form th:action="@{/pegawai/delete}" th:object="${cabang}" method="POST">
    <table class="table">
      <thead>
        <tr>
          <th>No</th>
          <th>No Pegawai</th>
          <th>Nama</th>
          <th>Jenis Kelamin</th>
          <th></th>
          <th>Hapus</th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="pegawai, iterationStatus : ${listPegawai}"
            th:style="${iterationStatus.even} ? 'font-weight:bold;'">
          <td th:text="${iterationStatus.count}"></td>
          <td th:text="${pegawai.noPegawai}"></td>
          <td th:text="${pegawai.namaPegawai}"></td>
          <td th:if="${pegawai.jenisKelamin} == 0">Perempuan</td>
          <td th:if="${pegawai.jenisKelamin} == 1">Laki-Laki</td>
          <td>
            <a class="btn btn-sm btn-primary" th:href="@{/pegawai/update/} + ${pegawai.noPegawai}">Update</a>
          </td>
          <td>
            <a class="btn btn-sm btn-primary" th:href="@{/pegawai/update/} + ${pegawai.noPegawai}">Update</a>
          </td>
        </tr>
      </tbody>
    </table>
  </form>
</div>
<div th:unless="*{listPegawai.size() != 0}">
  <h2 th:text="'Tidak ada pegawai'"></h2>
</div>
```

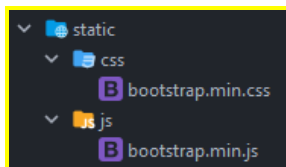
## Penggunaan File Static

Spring Boot dan Thymeleaf memberikan fasilitas untuk menggunakan static file seperti file css, javascript, image, dan sebagainya. Static file tersebut disimpan dan diakses melalui folder static yang sudah tersedia secara default dari pembuatan project berbasis Thymeleaf. Pada bagian ini, kita akan

mempelajari implementasi framework css bootstrap sebagai static file yang dapat digunakan pada Thymeleaf.

Berikut adalah langkah-langkah penggunaan Bootstrap pada Springboot:

1. Download CSS dan JS Bootstrap pada [halaman ini](#).
2. Ekstrak file zip dan hanya pindahkan 'bootstrap.min.css' dan 'bootstrap.min.js' ke dalam masing-masing folder 'css' dan 'js' pada direktori **src/main/resources/static**.



3. Bootstrap juga memerlukan jQuery dan Popper agar semua fitur dapat bekerja dengan baik. Lakukan langkah yang sama seperti pada bootstrap atau dapat memasukkan **cdn JQuery** ke dalam HTML. Untuk mengimport file static dan menambahkan cdn, kita dapat langsung menggunakan syntax pada HTML. Tambahkan kode berikut pada home.html untuk menambahkan bootstrap melalui file static dan JQuery melalui cdn.

```
<head>
  <meta charset="UTF-8">
  <title> Home </title>
  <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}"/>
  <script
    src="https://code.jquery.com/jquery-3.4.1.min.js"
    integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFIBw8HfCJo="
    crossorigin="anonymous"
  ></script>
  <script src="https://getbootstrap.com/docs/4.1/assets/js/vendor/popper.min.js"></script>
  <script th:src="@{/js/bootstrap.min.js}"></script>
</head>
```

```
<head>
  <meta charset="UTF-8">
  <title> Home </title>
  <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}"/>
  <script
    src="https://code.jquery.com/jquery-3.4.1.min.js"
    integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFIBw8HfCJo="
    crossorigin="anonymous"
  ></script>
  <script
    src="https://getbootstrap.com/docs/4.1/assets/js/vendor/popper.min.js"
  ></script>
  <script th:src="@{/js/bootstrap.min.js}"></script>
</head>
```

4. Lalu, ubahlah body home.html menjadi seperti

```
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-danger">
  <a class="navbar-brand font-weight-bold text-light" href="/">Emsidi</a>
  <button
    class="navbar-toggler"
    type="button"
    data-toggle="collapse"
    data-target="#navbarNavAltMarkup"
    aria-controls="navbarNavAltMarkup"
    aria-expanded="false"
    aria-label="Toggle navigation"
  >
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item">
        <a class="nav-link" href="/">Home</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0" th:action="@{/cabang/view}" method="GET">
      <input class="form-control mr-sm-2" type="search"
        name="noCabang" placeholder="No Cabang" aria-label="Search">
      <button class="btn btn-outline-light my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>

<div class="d-flex justify-content-center">
  <div class="m-4">
    <h2>Selamat datang di Emsidi</h2>
    <div class="d-flex justify-content-center">
      <a th:href="@{/cabang/viewall}" class="mx-2 btn btn-dark">Lihat semua cabang</a>
      <a th:href="@{/cabang/add}" class="mx-2 btn btn-outline-dark">Tambah cabang</a>
    </div>
    <div class="my-3 d-flex justify-content-center">
      <a th:href="@{/menu/viewall}" class="mx-2 btn btn-dark">Lihat semua menu</a>
      <a th:href="@{/menu/add}" class="mx-2 btn btn-outline-dark">Tambah menu</a>
    </div>
  </div>
</div>

</body>
```

5. Akses kembali halaman home, tampilan akan menjadi seperti ini.

Selamat datang di Emsidi

Lihat semua cabang

Tambah cabang

Lihat semua menu

Tambah menu

## Penggunaan Fragment

Fragment merupakan sebuah fitur dalam Thymeleaf yang memungkinkan developer untuk me-reuse source code dari halaman HTML-nya. Fitur ini memungkinkan kamu untuk memecah halaman HTML menjadi bagian-bagian kecil dan bagian tersebut dapat digunakan oleh halaman HTML lain. Code reuse ini berguna ketika developer menggunakan komponen-komponen yang sering dipakai di berbagai bagian dalam aplikasi yang dibuatnya. Contohnya adalah pembuatan navigation bar. Thymeleaf juga sudah menyediakan dokumentasi mengenai fragment yang dapat diakses pada <http://www.thymeleaf.org/doc/articles/layouts.html>

Berikut adalah langkah-langkah penggunaan fragment pada Springboot.

1. Buat sebuah folder dalam folder templates dengan nama fragments
2. Buat sebuah halaman HTML bernama fragment.html di folder yang telah dibuat sebelumnya yang berisi source code seperti dibawah ini.



```

<!DOCTYPE html>|
<html lang="en" xmlns:th="https://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Home</title>
  <head th:fragment="css">
    <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}"/>
  </head>
  <head th:fragment="js">
    <script
      src="https://code.jquery.com/jquery-3.4.1.min.js"
      integrity="sha256-CSXorXvZcTkaix6Yv6oHppcZGetbYMGWsfIBw8HfCJo="
      crossorigin="anonymous"
    ></script>
    <script src="https://getbootstrap.com/docs/4.1/assets/js/vendor/popper.min.js"></script>
    <script th:src="@{/js/bootstrap.min.js}"></script>
  </head>
</head>
<body>

```

```

<nav th:fragment="navbar(page)" class="navbar navbar-expand-lg navbar-light bg-danger">
  <a class="navbar-brand font-weight-bold text-light" href="/">Emsidi</a>
  <button
    class="navbar-toggler"
    type="button"
    data-toggle="collapse"
    data-target="#navbarNavAltMarkup"
    aria-controls="navbarNavAltMarkup"
    aria-expanded="false"
    aria-label="Toggle navigation"
  >
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse">
    <ul class="navbar-nav mr-auto">
      <li th:classappend="{page == 'Home'} ? active : ''" class="nav-item">
        <a class="nav-link" href="/">Home</a>
      </li>
      <li th:classappend="{page == 'Cabang'} ? active : ''" class="nav-item">
        <a class="nav-link" href="/cabang/viewall">Cabang</a>
      </li>
      <li th:classappend="{page == 'Menu'} ? active : ''" class="nav-item">
        <a class="nav-link" href="/menu/viewall">Menu</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0" th:action="@{/cabang/view}" method="GET">
      <input class="form-control mr-sm-2" type="search"
        name="noCabang" placeholder="No Cabang" aria-label="Search">
      <button class="btn btn-outline-light my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>
</body>
</html>

```

3. Pada source code diatas, `th:fragment` merepresentasikan nama fragment dari bagian yang kamu ingin jadikan sebuah fragment.
4. Lalu, fragment dapat digunakan dengan syntax seperti berikut ini. Cobalah untuk mengubah `home.html`.



```
<head>
  <title>Emsidi</title>
  <meta charset="UTF-8">
  <title> Home </title>
  <object th:include="fragments/fragment :: css" th:remove="tag"></object>
  <object th:include="fragments/fragment :: js" th:remove="tag"></object>
</head>

<body>
  <nav th:replace="fragments/fragment :: navbar('Home')"></nav>
```

The image shows a code editor with Thymeleaf source code. Three red arrows point to specific parts of the code: one points to the `th:include` attribute in the CSS object tag, another points to the `th:include` attribute in the JS object tag, and a third points to the `th:replace` attribute in the navbar tag. All three attributes use the `fragments/fragment` as a reference.

5. Implementasikan hal serupa untuk **semua page** lainnya. Pastikan semua page memiliki navbar.

## Error Handling 404 Not Found

Dalam membuat aplikasi berbasis web, terkadang halaman web yang dituju oleh pengguna tidak dapat ditemukan. Pengguna dapat kebingungan jika diberikan halaman error yang tidak umum atau tidak jelas. Untuk mengatasi hal tersebut, kita dapat membuat handler melalui Thymeleaf. Pembuatan handler ini berguna agar pengguna mengetahui mengapa suatu error terjadi. Misalnya jika terjadi error 404 not found, kita dapat mengembalikan halaman HTML yang mengatakan bahwa halaman yang pengguna akses tersebut tidak dapat ditemukan. Hal ini akan meningkatkan UX value yang diberikan oleh aplikasi yang kita kembangkan kepada pengguna.

Berikut adalah cara membuat halaman error pada Springboot.

1. Buat sebuah folder dalam folder **templates** dengan nama **error**

2. Buat sebuah halaman HTML bernama **404.html** di folder yang baru saja dibuat yang berisi source code seperti dibawah ini.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>404 not found</title>
  <object th:include="fragments/fragment :: css" th:remove="tag"></object>
  <object th:include="fragments/fragment :: js" th:remove="tag"></object>
</head>
<body>
  <nav th:replace="fragments/fragment :: navbar('404')"></nav>
  <div class="container-fluid">
    <h2>Halaman tidak ditemukan</h2>
    <a th:href="@{/}">Kembali ke home</a>
  </div>
</body>
</html>
```

3. Coba akses url yang tidak kamu definisikan. Misal <http://localhost:8080/belum-ada>
4. Tambahkan untuk beberapa error-error yang lain (ex: 500, 400).

## Form Handler

Thymeleaf menyediakan fitur integrasi penuh untuk Spring Boot Framework. Salah satu fitur utama yang disediakan Thymeleaf untuk mendukung Spring Boot adalah hal-hal yang berkaitan dengan form. Pada tutorial sebelumnya, kamu telah menggunakan expression seperti `th:each`, `th:text`, `th:value`, maupun `th:field`. Pada bagian tutorial ini, kamu akan mempelajari bagaimana Thymeleaf meng-handle checkbox yang dapat multivalue. Pada `PegawaiController`, tambahkan sebuah routing baru dengan isi sebagai berikut.

```

@PostMapping("/pegawai/delete")
public String deletePegawaiSubmit(
    @ModelAttribute CabangModel cabang,
    Model model
){
    LocalDateTime now = LocalDateTime.now();
    if (now.isBefore(cabang.getWaktuBuka()) || now.isAfter(cabang.getWaktuTutup())){
        for (PegawaiModel pegawai: cabang.getListPegawai()) {
            pegawaiService.removePegawai(pegawai);
        }
        model.addAttribute("noCabang", cabang.getNoCabang());
        return "delete-pegawai";
    }
    return "error-cannot-delete";
}

```

Ubah delete pegawai untuk menghapus pegawai yang dipilih pada file view-cabang.html agar dapat menerima multiple checkbox menjadi seperti berikut ini.

```

<td>
    <a class="btn btn-sm btn-primary" th:href="@{/pegawai/update/} + ${pegawai.noPegawai}">Update</a>
</td>
<td>
    <input class="form-check-input" type="checkbox"
        th:field="*{listPegawai}" th:value="${pegawai.noPegawai}"
        id="flexCheckDefault">
</td>

```

Di bawah form delete pegawai, tambahkan ketiga hidden field berikut

```

<form th:action="@{/pegawai/delete}" th:object="${cabang}" method="POST">
    <input type="hidden" th:field="*{noCabang}" />
    <input type="hidden" th:field="*{waktuBuka}" />
    <input type="hidden" th:field="*{waktuTutup}" />
    <table class="table">

```

Masih pada file view-cabang.html, tambahkan tombol seperti berikut :

```

</tbody>
</table>
<input type="submit" value="Delete yang dipilih" class="btn btn-danger">
</form>

```

Sesuaikan juga file delete-pegawai.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title> Delete Pegawai </title>
  <object th:include="fragments/fragment :: css" th:remove="tag"></object>
  <object th:include="fragments/fragment :: js" th:remove="tag"></object>
</head>
<body>
<nav th:replace="fragments/fragment :: navbar('')"></nav>
<div class="container">
  <div class="card m-4 p-4">
    <div class="card-body">
      <div class="justify-content-center">
        <br>
        <h2>
          th:text="'Pegawai terkait telah berhasil dihapus'"
        </h2>
        <a class="btn btn-primary" th:href="@{/cabang/view(noCabang=${noCabang})}">Kembali</a>
      </div>
    </div>
  </div>
</div>
</body>
</html>

```

Ketika tombol “Hapus yang dipilih” di klik, aplikasi akan melakukan POST request ke url ‘/delete/pegawai’ untuk menghapus list pegawai yang telah dicentang pada checkbox sebelumnya. Setelah melakukan semua perubahan tersebut, kamu akan dapat melakukan penghapusan banyak pegawai hanya dengan 1x proses. Hal ini dimungkinkan karena pada file view cabang, expression pada input checkbox dan label diproses menjadi HTML seperti ini.

No	No Pegawai	Nama	Jenis Kelamin		Hapus
1	1	Ronaldo	Laki-Laki	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	2	Messi	Laki-Laki	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	3	Markonah	Perempuan	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Delete yang dipilih

## Pelengkap

Pada latihan tutorial ini, akan dilibatkan `object Menu`, oleh karena itu Anda bisa menambahkan hal berikut pada **MenuModel.java**

```

@NotNull
@Size(max = 50)
@Column(nullable = false)
private String deskripsiMenu;

```

Kemudian, jangan lupa meng-copy dan paste code dari pastebin berikut ya :)

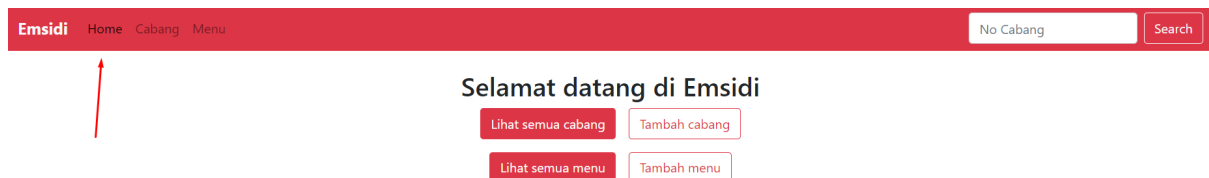
1. MenuDB.java (<https://pastebin.com/VwbBBW18>)
2. MenuService.java (<https://pastebin.com/qVvmSymA>)
3. MenuServiceImpl.java (<https://pastebin.com/dGEkZpt4>)
4. MenuController.java (<https://pastebin.com/5kAha8C1>)
5. form-add-menu.html (<https://pastebin.com/M629RbXn>)
6. add-menu.html (<https://pastebin.com/6wsH3zu3>)
7. viewall-menu.html (<https://pastebin.com/uRZxNaxP>)

## LATIHAN

1. Ubahlah View All Cabang dan View All Menu menjadi berbentuk tabel. Khusus View All Cabang, sediakan tombol lihat, ubah, dan hapus untuk tiap row nya. Ubahlah halaman tersebut menjadi lebih rapi.
2. Pada tutorial, Anda sudah mencoba membuat fragment navigation bar. Tetapi, navigation bar pada tutorial tersebut tidak dinamis. Data HTML pada header akan sama di setiap view yang memakai fragment tersebut. Pada latihan ini, kamu diminta untuk menambah satu menu pada navbar yang dapat berubah-ubah sesuai dengan judul halaman sekarang. Tambahkan menu ini setidaknya pada halaman home, view all cabang, (add, view, update) cabang, (add, view, update) menu.

Contoh (perhatikan kiri navbar):

- a. Ketika berada di halaman Home



- b. Ketika berada di halaman View Cabang



### Hint:

- Padu padankan `thymeleaf th:classappend` dan `class active` dari `bootstrap`.

3. Pada bagian form handler, anda telah mempelajari cara meng-handle input multiple checkbox untuk dapat melakukan delete banyak menu sekaligus. Pada latihan ini, anda diminta untuk dapat melakukan insert banyak menu sekaligus. Salah satu contoh implementasinya sebagai berikut:


- a. Ubah atau tambah html form untuk add cabang (url : '/cabang/add') menjadi muncul form yang isinya seperti ini.


### Tambah Cabang Emsidi

Nama Cabang :

Alamat :

Nomor Telepon :

Waktu Buka :  

Waktu Tutup :  

Nama Menu	Tambah Row
<input type="text" value="Ayam"/>	Hapus

- b. Apabila add row diklik, dropdown untuk menambah menu bertambah 1 baris, lalu jika dropdown-nya dipilih akan seperti ini. Pilihan pada dropdown merupakan daftar seluruh menu yang ada pada database.

Nama Menu	Tambah Row
<input type="text" value="Ayam"/>	Hapus
<input type="text" value="Burger"/>	Hapus

Nama Menu	Tambah Row
<input type="text" value="Ayam"/>	Hapus
<div><input type="text" value="Burger"/><div><div>Ayam</div><div>Burger</div></div></div>	Hapus

- c. Apabila hapus row diklik, row yang bersangkutan menjadi hilang seperti ini. Contoh row pertama yang dihapus.

Nama Menu

Tambah Row

Burger

Hapus

Submit

Cancel

- d. Ketika berhasil disimpan maka akan muncul tampilan berhasil seperti ini.

Cabang Emsidi dengan nomor 3 berhasil ditambahkan

Kembali ke Home

### Hint:

- Buat method-method pada controller dengan request method POST untuk menambah row, menghapus row, dan men-submit form.
- Setiap menekan add row/hapus row halaman akan ter-load ulang.
- Gunakan list *temporary*/sementara berisi objek menu untuk menyimpan row yang ditampilkan.

4. Pada halaman view-cabang, tambahkan tabel yang menampilkan seluruh menu yang ada pada cabang tersebut. Berikut adalah tampilan final untuk view-cabang.

Detail Cabang

Nomor Cabang : 3

Nama Cabang : MAARRRGONDA

Alamat Cabang : Jalan Rindu

Nomor Telepon Cabang : 091223

Waktu Buka : 09:00

Waktu Tutup : 21:00

No	No Pegawai	Nama	Jenis Kelamin		Hapus
1	4	Nam Do san	Perempuan	Update	<input checked="" type="checkbox"/>

Delete yang dipilih

No	No Menu	Nama	Deskripsi	Sedang Tersedia
1	2	Burger	Burger KFC	

Tambah Pegawai

Kembali



## PERTANYAAN

1. Jelaskan perbedaan `th:include` dan `th:replace`!
2. Jelaskan apa fungsi dari `th:object`!
3. Jelaskan perbedaan dari `*` dan `$` pada saat penggunaan `th:object`! Kapan harus dipakai?



## “Cukupkanlah”

- *Kunto Aji*