

Nama : Hanifa Arrumaisha

NPM : 1606824332

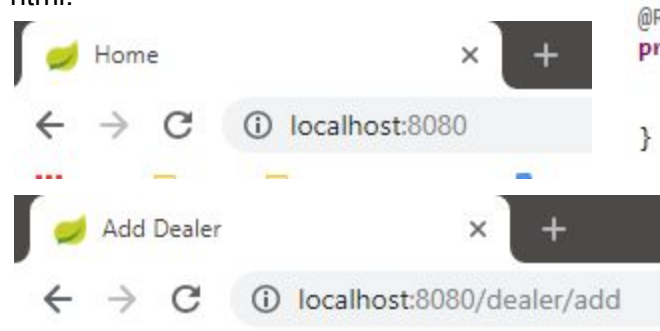
Kelas : APAP-A

Notes to kakak asdosku: saya tau kak saya telat hehe, saya akan menerima konsekuensinya, terimakasih masih mau memeriksa hasil pengerjaan saya, yang dapat saya jamin saya mengerjakan sendiri.

1. 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 kali ini, Anda diminta untuk membuat sebuah fragment header yang dinamis. Dimana, aspek dinamis tersebut terdapat pada title yang diletakkan pada fragment header, bukan setiap page HTML yang ada. Lebih jauh, title tersebut akan berubah-ubah sesuai dengan page yang ada. Silahkan implementasi dengan kebutuhan tersebut, dan jika sudah berhasil, pastikan header tersebut dimuat pada setiap view yang ada di project ini.

Jawab:

Dalam pengerjaan ini, saya mengubah file fragment.html, membuat fragment baru berisi tag title yang didalamnya terdapat attribute thymeleaf th:text dan diisi dengan variable bernama title yang dapat di pass melalui controller dengan nama atribut yang sama yaitu title. Jadi nanti setiap controller yang memanggil html yang memiliki fragment ini harus memberikan attribute title pada html.



```
@RequestMapping("/")
private String home(Model model) {
    model.addAttribute("title", "Home");
    return "home";
}

@RequestMapping(value = "/dealer/add", method = RequestMethod.POST)
private String addDealerSubmit(@ModelAttribute DealerModel dealer, Model model) {
    dealerService.addDealer(dealer);
    model.addAttribute("title", "Berhasil! Add Car");
    return "add";
}
```

Begitu juga dengan file-file yang lainnya.

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

- Pada halaman add-car terdapat tombol untuk menambah field untuk object mobil lainnya

APAP [Home](#)

Hello!

Tambah Mobil

No	Brand	Type	Price	Amount	
1	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<div>Tambah</div> <div>Hapus</div>

Save

- Ketika tombol tambah ditekan, maka akan muncul field untuk insert mobil lainnya, dan ketika tombol hapus ditekan, maka row tersebut akan terhapus.

APAP [Home](#)

Hello!

Tambah Mobil

No	Brand	Type	Price	Amount	
1	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<div>Hapus</div>
2	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<div>Hapus</div>

Save

- Ketika tombol simpan ditekan, maka semua data mobil yang diinput akan ter-insert ke database

Hello!

Tambah Mobil

No	Brand	Type	Price	Amount	Tambah
1	coba	satuu	1888	89	Hapus
2	coba lagi	duaa	899	89	Hapus

Save

Data Berhasil Ditambahkan

Dealer 5

Alamat: jalanan

Telepon: 879

Hapus Dealer

No.	Brand	Type	Price	Amount	Aksi
1	coba lagi	duaa	Dibawah 1M	89	<input checked="" type="checkbox"/> Hapus
2	coba	satuu	Dibawah 1M	89	<input checked="" type="checkbox"/> Hapus
3	Mitsubishi Xenia	Dibawah 1M	152		<input checked="" type="checkbox"/> Hapus

Hapus yang dipilih

Tambah Mobil

HTML yang digunakan seperti berikut:

```
<h3>Tambah Mobil</h3>
<form th:action="@{'/car/add/' + ${dealer.id}}" th:object="${dealer}" method="POST">
  <input type="hidden" th:field="*{id}"></input>
  <table>
    <thead>
      <tr>
        <th>No</th>
        <th>Brand</th>
        <th>Type</th>
        <th>Price</th>
        <th>Amount</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>
          <!-- *{listCar[__${iterStat.index}__].brand} itu untuk ngambil data ke listcar pada index tersebut, karena kalau pakai
          langsung kayak biasa dia gabisa, perlu ada predecessor [__] -->
          <input type="hidden" th:field="*{listCar[__${iterStat.index}__].dealer}" th:value="${dealer}" />
          <input type="text" th:field="*{listCar[__${iterStat.index}__].brand}" />
        </td>
        <td>
          <input type="text" th:field="*{listCar[__${iterStat.index}__].type}" />
        </td>
        <td>
          <input type="text" th:field="*{listCar[__${iterStat.index}__].price}" />
        </td>
        <td>
          <input type="text" th:field="*{listCar[__${iterStat.index}__].amount}" />
        </td>
        <td>
          <button type="submit" name="addRow" th:text="Tambah">Tambah</button>
        </td>
      </tr>
    </tbody>
  </table>
  <tr>
    <td>
      <button type="submit" name="removeRow"
        th:value="${iterStat.index}" th:text="Hapus">Remove row</button>
    </td>
  </tr>
</form>
```

- Terdapat th:object="\${dealer}" yang memberikan data ke controller berupa model yang telah dikirimkan sebelumnya oleh controller ke template, namun kini telah dilakukan perubahan melalui form dengan trigger dari button yang ada dengan masing-masing parameternya.
- Th:object terhubung dengan @ModelAttribute pada controller di bagian parameter fungsi untuk menerima model yang dikirimkan dari template ke controller.

Terdapat pembuatan 4 buah fungsi:

1. Fungsi biasa di-init awal-awal:

```
@RequestMapping(value = "/car/add/{dealerId}", method = RequestMethod.GET)
private String add(@PathVariable(value = "dealerId") Long dealerId, Model model) {
    DealerModel dealer = dealerService.getDealerDetailById(dealerId).get();
    ArrayList<CarModel> cars = new ArrayList<CarModel>();
    cars.add(new CarModel());
    dealer.setListCar(cars);

    model.addAttribute("dealer", dealer);
    model.addAttribute("title", "Add Car");
    return "addCar-dynamic";
}
```

dikarenakan kita ingin membuat data car yang terdapat dalam Dealer, jadi kita harus mengambil object dealer terlebih dahulu, lalu saat pembuatan

list car yang baru untuk dealer tersebut, kita buat dia listCar baru dan initial car untuk menambahkan car, lalu dikirim ke htmlnya.

2. Fungsi dengan parameter AddRow

```
@RequestMapping(value="/car/add/{dealerId}", params={"addRow"})
public String addRow(@ModelAttribute DealerModel dealer, final BindingResult bindingResult, Model model) {
    if (dealer.getListCar()==null) {
        dealer.setListCar(new ArrayList<CarModel>());
    }
    dealer.getListCar().add(new CarModel());
    for (int i=0;i<dealer.getListCar().size();i++) {
        System.out.println(dealer.getListCar().get(i));
    }
    model.addAttribute("dealer", dealer);
    return "addCar-dynamic";
}
```

Parameter ini addRow didapatkan dari name pada atribut button submit, lalu dilakukan penambahan CarModel didalam listCar.

3. Fungsi dengan parameter save

```
@RequestMapping(value="/car/add/{dealerId}", params={"save"})
public String saveRow(@PathVariable(value = "dealerId") Long dealerId,
    @ModelAttribute DealerModel dealer, final BindingResult bindingResult) {
    for (CarModel car : dealer.getListCar()) {
        System.out.println(dealer.getId());
        car.setDealer(dealer);
        carService.addCar(car);
    }
    return "add";
}
```

4. Fungsi dengan parameter removeRow

```
@RequestMapping(value="/car/add/{dealerId}", params={"removeRow"})
public String removeRow(@PathVariable(value = "dealerId") Long dealerId,
    @ModelAttribute DealerModel dealer, final BindingResult bindingResult,
    final HttpServletRequest req, Model model) {
    final Integer carId = Integer.valueOf(req.getParameter("removeRow"));
    System.out.println(carId);
    dealer.getListCar().remove(carId.intValue());
    model.addAttribute("dealer", dealer);
    return "addCar-dynamic";
}
```

3. Jelaskan perbedaan antara th:include dengan th:replace.

Saya tidak menemukan ada th:include, yang saya temukan adalah th:insert. Adapun th:insert memasukkan fragment ke dalam body dari tag yang memanggil th:insert, dan content didalam body dari tag yang memanggil akan terhapus.

Th:replace akan menggantikan tag yang memanggil fragment tersebut.