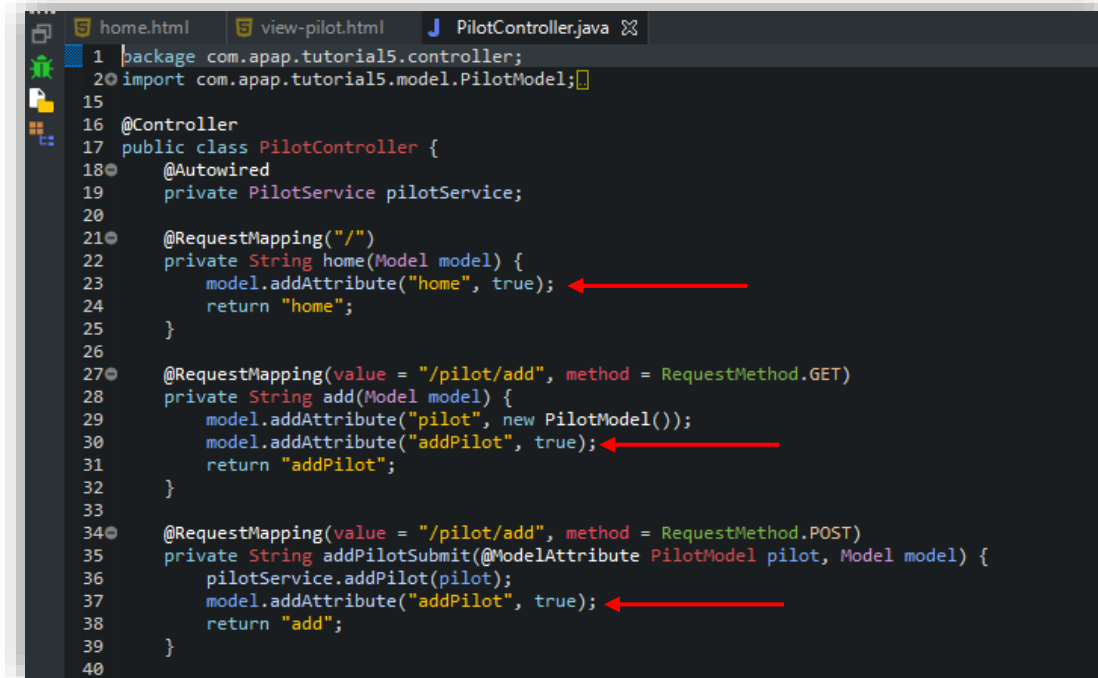


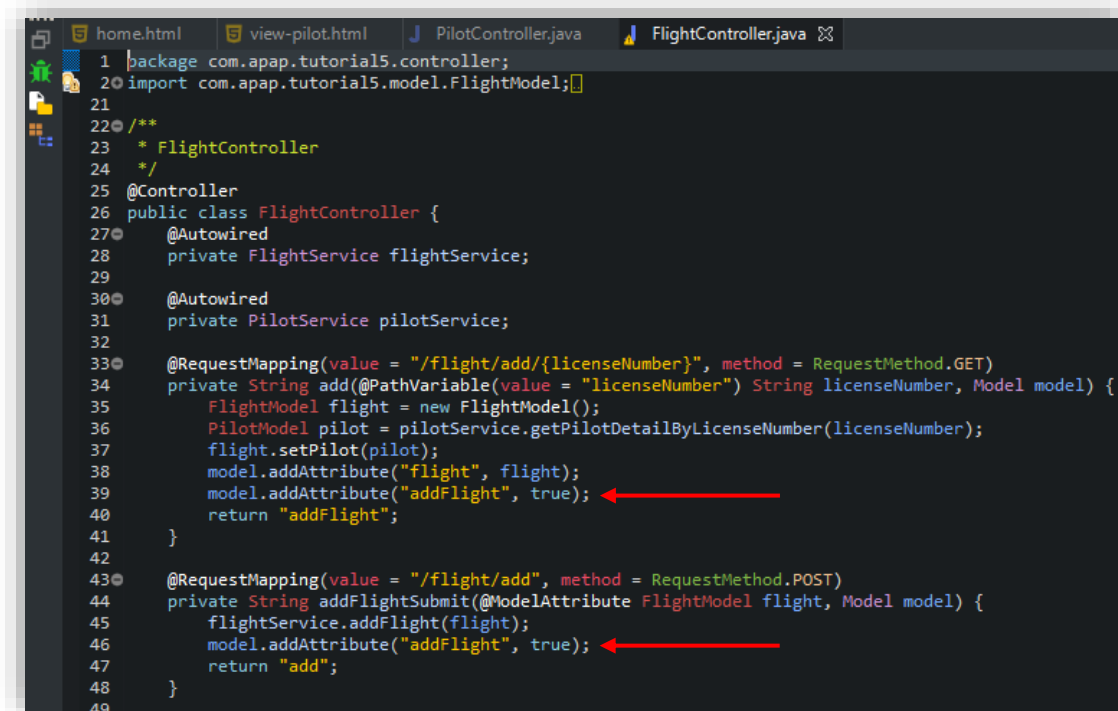
Latihan Tutorial 5

1. Untuk soal latihan no 1, awalnya saya sangat bingung bagaimana cara menyelesaikannya. Beberapa jam saya mencari jawaban di internet yaitu dari website stackoverflow.com, tutorialspoint.com, dan nixmash.com untuk mencari cara bagaimana menerapkan navbar brand yang dinamis. Akhirnya setelah beberapa jam mencari, meskipun tidak mendapatkan jawaban yang pas, tapi dari salah satu postingan di ketiga website tersebut saya mendapatkan ide untuk menyelesaikan masalah ini, yakni sebagai berikut.
 - a. Saya menambahkan atribut sesuai dengan nama fungsi dari @RequestMapping yang diminta user dengan value "true" yang menandakan bahwa halaman dari @RequestMapping tersebut sedang dibuka



```
1 package com.apap.tutorial5.controller;
2 import com.apap.tutorial5.model.PilotModel;
15
16 @Controller
17 public class PilotController {
18     @Autowired
19     private PilotService pilotService;
20
21     @RequestMapping("/")
22     private String home(Model model) {
23         model.addAttribute("home", true);
24         return "home";
25     }
26
27     @RequestMapping(value = "/pilot/add", method = RequestMethod.GET)
28     private String add(Model model) {
29         model.addAttribute("pilot", new PilotModel());
30         model.addAttribute("addPilot", true);
31         return "addPilot";
32     }
33
34     @RequestMapping(value = "/pilot/add", method = RequestMethod.POST)
35     private String addPilotSubmit(@ModelAttribute PilotModel pilot, Model model) {
36         pilotService.addPilot(pilot);
37         model.addAttribute("addPilot", true);
38         return "add";
39     }
40 }
```

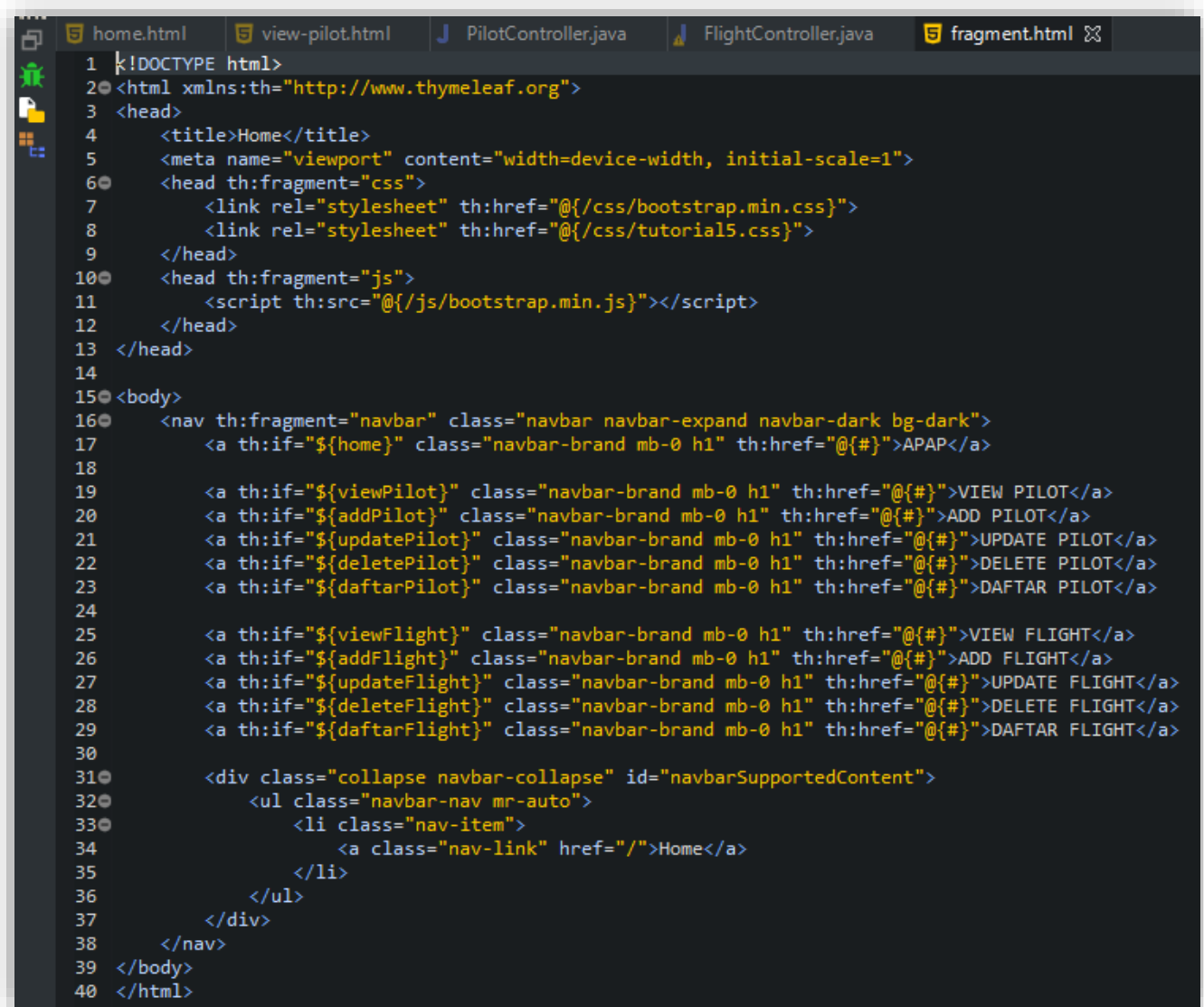
Gambar 1 : Tampilan Class PilotController



```
1 package com.apap.tutorial5.controller;
2 import com.apap.tutorial5.model.FlightModel;
21
22 /**
23  * FlightController
24  */
25 @Controller
26 public class FlightController {
27     @Autowired
28     private FlightService flightService;
29
30     @Autowired
31     private PilotService pilotService;
32
33     @RequestMapping(value = "/flight/add/{licenseNumber}", method = RequestMethod.GET)
34     private String add(@PathVariable(value = "licenseNumber") String licenseNumber, Model model) {
35         FlightModel flight = new FlightModel();
36         PilotModel pilot = pilotService.getPilotDetailByLicenseNumber(licenseNumber);
37         flight.setPilot(pilot);
38         model.addAttribute("flight", flight);
39         model.addAttribute("addFlight", true);
40         return "addFlight";
41     }
42
43     @RequestMapping(value = "/flight/add", method = RequestMethod.POST)
44     private String addFlightSubmit(@ModelAttribute FlightModel flight, Model model) {
45         flightService.addFlight(flight);
46         model.addAttribute("addFlight", true);
47         return "add";
48     }
49 }
```

Gambar 2 : Tampilan Class FlightController

- b. Lalu pada fragment.html, saya menambahkan semua opsi navbar brand yang berbeda sesuai dengan halaman yang sedang dibuka dan dibuat kondisi dengan menggunakan th:if sehingga tiap navbar brand hanya akan aktif jika menerima parameter yang sesuai dengan kondisi yang diminta. Berikut tampilan fragment.html



```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <title>Home</title>
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <head th:fragment="css">
7     <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
8     <link rel="stylesheet" th:href="@{/css/tutorial15.css}">
9   </head>
10  <head th:fragment="js">
11    <script th:src="@{/js/bootstrap.min.js}"></script>
12  </head>
13 </head>
14
15 <body>
16  <nav th:fragment="navbar" class="navbar navbar-expand navbar-dark bg-dark">
17    <a th:if="${home}" class="navbar-brand mb-0 h1" th:href="@{#}">APAP</a>
18
19    <a th:if="${viewPilot}" class="navbar-brand mb-0 h1" th:href="@{#}">VIEW PILOT</a>
20    <a th:if="${addPilot}" class="navbar-brand mb-0 h1" th:href="@{#}">ADD PILOT</a>
21    <a th:if="${updatePilot}" class="navbar-brand mb-0 h1" th:href="@{#}">UPDATE PILOT</a>
22    <a th:if="${deletePilot}" class="navbar-brand mb-0 h1" th:href="@{#}">DELETE PILOT</a>
23    <a th:if="${daftarPilot}" class="navbar-brand mb-0 h1" th:href="@{#}">DAFTAR PILOT</a>
24
25    <a th:if="${viewFlight}" class="navbar-brand mb-0 h1" th:href="@{#}">VIEW FLIGHT</a>
26    <a th:if="${addFlight}" class="navbar-brand mb-0 h1" th:href="@{#}">ADD FLIGHT</a>
27    <a th:if="${updateFlight}" class="navbar-brand mb-0 h1" th:href="@{#}">UPDATE FLIGHT</a>
28    <a th:if="${deleteFlight}" class="navbar-brand mb-0 h1" th:href="@{#}">DELETE FLIGHT</a>
29    <a th:if="${daftarFlight}" class="navbar-brand mb-0 h1" th:href="@{#}">DAFTAR FLIGHT</a>
30
31    <div class="collapse navbar-collapse" id="navbarSupportedContent">
32      <ul class="navbar-nav mr-auto">
33        <li class="nav-item">
34          <a class="nav-link" href="/">Home</a>
35        </li>
36      </ul>
37    </div>
38  </nav>
39 </body>
40 </html>
```

Gambar 3 : Tampilan fragment.html

- c. Oleh karena dua konfigurasi yang sudah saya lakukan sebelumnya maka navbar brand akan berubah secara dinamis sesuai dengan halaman yang sedang dibuka. Berikut contoh tampilan dari halaman web yang dibuka.

← → ↻ 🏠 ⓘ localhost:8080

APAP Home

Selamat datang di sistem penerbangan APAP Airlines

Tambah Pilot

Fitur untuk menambah pilot

Tambah

Cari Pilot Berdasarkan Nomor Lisensi

License Number:

Cari

← → ↻ 🏠 ⓘ localhost:8080/pilot/add

ADD PILOT Home

Tambah Pilot

License Number:

Name:

Flight Number:

Submit

[←](#) [→](#) [↻](#) [🏠](#) [localhost:8080/flight/view?flightNumber=333](#)

VIEW FLIGHT [Home](#)

Detail Flight

Flight Number:	333
Pilot Name:	wigoooooh
Pilot License Number:	111
Origin:	Bali
Destination:	Padang
Time:	11-11-2021

[←](#) [→](#) [↻](#) [🏠](#) [localhost:8080/pilot/update/111](#)

UPDATE PILOT [Home](#)

Update Pilot

License Number:

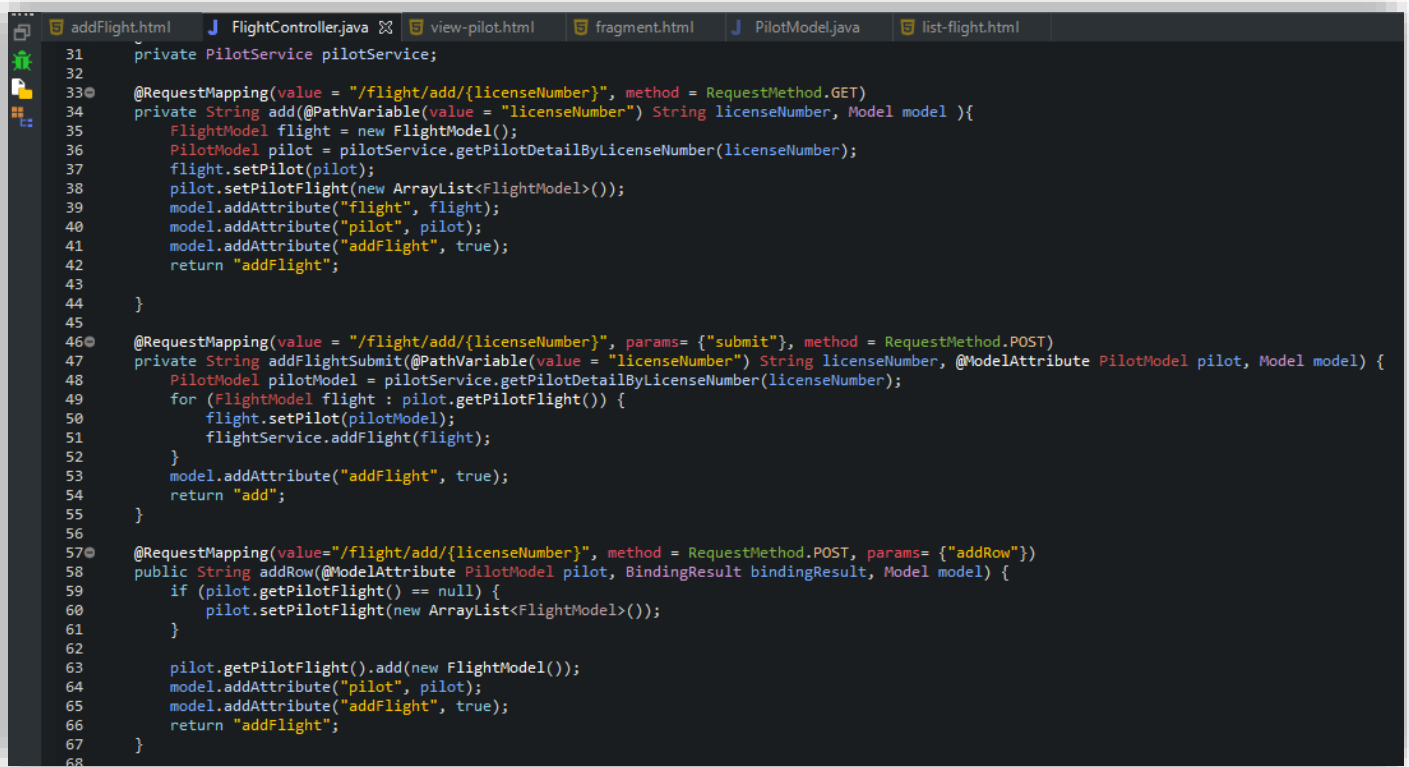
111

Name:

Fly Hour:

Submit

- Untuk soal nomor 2 saya melakukan perubahan dan penambahan Mapping baru pada Flight Controller untuk memetakan request dari user yang ingin menambah field input untuk Add Pilot (bukan Submit). Berikut perubahan dan penambahan Mapping yang saya lakukan.



```
31 private PilotService pilotService;
32
33 @RequestMapping(value = "/flight/add/{licenseNumber}", method = RequestMethod.GET)
34 private String add(@PathVariable(value = "licenseNumber") String licenseNumber, Model model ){
35     FlightModel flight = new FlightModel();
36     PilotModel pilot = pilotService.getPilotDetailByLicenseNumber(licenseNumber);
37     flight.setPilot(pilot);
38     pilot.setPilotFlight(new ArrayList<FlightModel>());
39     model.addAttribute("flight", flight);
40     model.addAttribute("pilot", pilot);
41     model.addAttribute("addFlight", true);
42     return "addFlight";
43 }
44
45
46 @RequestMapping(value = "/flight/add/{licenseNumber}", params= {"submit"}, method = RequestMethod.POST)
47 private String addFlightSubmit(@PathVariable(value = "licenseNumber") String licenseNumber, @ModelAttribute PilotModel pilot, Model model) {
48     PilotModel pilotModel = pilotService.getPilotDetailByLicenseNumber(licenseNumber);
49     for (FlightModel flight : pilot.getPilotFlight()) {
50         flight.setPilot(pilotModel);
51         flightService.addFlight(flight);
52     }
53     model.addAttribute("addFlight", true);
54     return "add";
55 }
56
57 @RequestMapping(value="/flight/add/{licenseNumber}", method = RequestMethod.POST, params= {"addRow"})
58 public String addRow(@ModelAttribute PilotModel pilot, BindingResult bindingResult, Model model) {
59     if (pilot.getPilotFlight() == null) {
60         pilot.setPilotFlight(new ArrayList<FlightModel>());
61     }
62
63     pilot.getPilotFlight().add(new FlightModel());
64     model.addAttribute("pilot", pilot);
65     model.addAttribute("addFlight", true);
66     return "addFlight";
67 }
68 }
```

Setelah itu saya juga melakukan perubahan pada file addFlight.html agar tiap kali tombol “+” (tambah pilot, bukan submit) ditekan, aplikasi mengenerate satu paket field baru untuk bisa dinput satu buah data Flight. Berikut tampilan dari file addFlight.html

```

addFlight.html FlightController.java view-pilot.html fragment.html PilotModel.java list-flight.html
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>Add Flight</title>
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <object th:include="fragments/fragment :: css" th:remove="tag"></object>
7 <object th:include="fragments/fragment :: js" th:remove="tag"></object>
8 </head>
9
10 <body>
11 <nav th:replace="fragments/fragment :: navbar"></nav>
12 <p>
13 <div class="container-fluid float-left" style="width: 950px;">
14 <h3>Tambah Pengebangan</h3>
15 <form th:action="@{/flight/add/} + ${pilot.licenseNumber}"
16 th:object="${pilot}" method="POST">
17 <table class="table">
18 <thead class="thead-light">
19 <tr>
20 <th style="text-align: center">Flight Number</th>
21 <th style="text-align: center">Origin</th>
22 <th style="text-align: center">Destination</th>
23 <th style="text-align: center">Time</th>
24 <th>
25 <button type="submit" name="addRow"
26 class="btn btn-secondary btn-sm">+</button>
27 </th>
28 </tr>
29 </thead>
30 <tbody>
31 <tr th:each="flight, iterator : ${pilotFlight}">
32 <th><input type="text" name="flightNumber"
33 class="form-control"
34 th:field="${pilotFlight[__${iterator.index}__].flightNumber}">
35 </th>
36 <th><input type="text" name="origin" class="form-control"
37 th:field="${pilotFlight[__${iterator.index}__].origin}">
38 </th>
39 <th><input type="text" name="destination"
40 class="form-control"
41 th:field="${pilotFlight[__${iterator.index}__].destination}">
42 </th>
43 <th><input type="date" name="time" class="form-control"
44 th:field="${pilotFlight[__${iterator.index}__].time}">
45 </th>
46 </tr>
47 </tbody>
48 </table>
49 <button class="btn btn-primary btn-block" type="submit" name="submit">Submit</button>
50 </form>
51 </div>
52 </body>
53 </html>

```