Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

Andre Gema Syahputra
1506689282
Kelas B

Tutorial kali ini mengulas mengenai *Model, View, Controller (MVC)* secara lebih mendalam, dan menggunakannya dalam sebuah aplikasi dengan menggunakan Spring Boot.

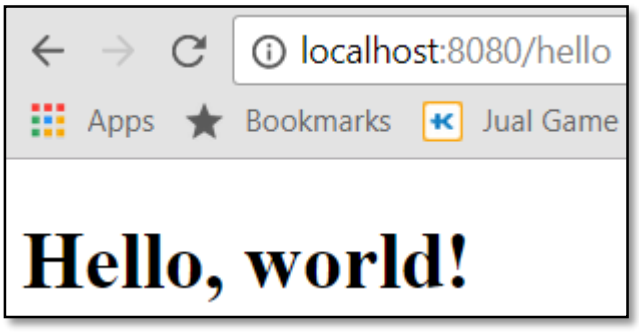
Pertama, tutorial ini mengajarkan untuk membuat sebuah *Controller* sederhana bernama ***PageController.java*** seperti berikut:

```
package com.example.demo.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class PageController
{
    @RequestMapping("/hello")
    public String index ()
    {
        return "hello";
    }
}
```

Selanjutnya, membuat view bernama ***hello.html*** seperti berikut:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>First Page</title>
    </head>
```

Aplikasi dapat dijalankan dengan mengklik kanan pada nama project di window Project Explorer > Run As…> Spring Boot App. Kemudian, buka **localhost:8080/hello** pada browser, akan muncul *page* sebagai berikut:



Disini, kita mempelajari bahwa **@Controller** pada **PageController.java** menandakan bahwa **PageController** merupakan sebuah *Controller* yang akan me-*respond HTTP Requests* dan mengembalikan *View*. Sedangkan **@RequestMapping("/hello")** menandakan bahwa *method* dibawahnya (index()) akan dijalankan apabila ada yang memanggil *path* /hello.

Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

Andre Gema Syahputra
1506689282
Kelas B

Latihan Project Hello World

1. Ganti baris tersebut menjadi @RequestMapping("/hello123")

   Pertanyaan:
   Apakah *compile error*?

Tidak

   Jika tidak, stop Spring Boot yang sedang berjalan, run kembali dan buka localhost: 8080/hello apa yang terjadi?

Terjadi sebuah *error* yang menampilkan sebuah *Whitelabel Error Page*. Hal ini terjadi dikarenakan tidak adanya @RequestMapping pada path /hello.

2. Ganti nama method index() dengan nama method hello()

   Pertanyaan:
   Apakah compile error?

Tidak

   Jika tidak, *Stop* Spring Boot yang sedang berjalan, run kembali dan buka localhost: 8080/hello apakah *page* hello sebelumnya masih muncul?

Masih muncul, karena perubahan nama *method* tidak berpengaruh terhadap jalannya aplikasi dalam kasus ini

3. Ganti string return type menjadi return "hello123";

   Pertanyaan:
   Apakah compile error?

Tidak

   Jika tidak, Stop Spring Boot yang sedang berjalan, run kembali dan buka localhost: 8080/hello apakah page hello sebelumnya masih muncul?

Terjadi *error* yang memunculkan *Whitelabel Error Page*. Hal ini dikarenakan tidak adanya *template* yang bernama “hello123”

   Menandakan apakah String yang di-return tersebut?

Menandakan nama *template* yang akan dikembalikan

Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

Andre Gema Syahputra
1506689282
Kelas B

Request Parameter (Query String)
Tutorial selanjutnya memplejari mengenai cara mengirimkan parameter menggunakan GET *request* pada Spring Boot.

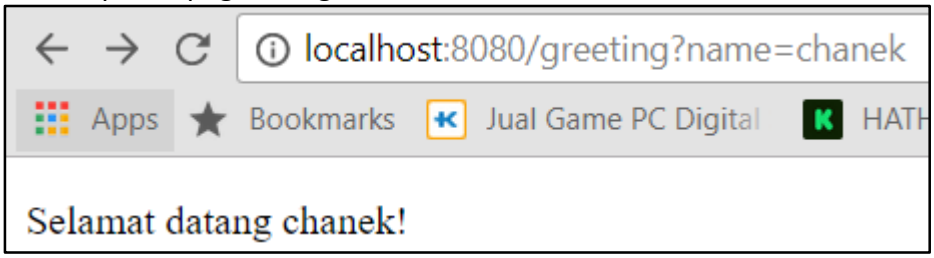
Pertama, buat *template* baru dengan nama **greeting.html** seperti gambar berikut:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Second Page</title>
    </head>
    <body>
       <p th:text="'Selamat datang ' + ${name} + '!'"> Sapaan
                    untuk user</p>
    </body>
</html>
```

Kemudian, tambahkan sebuah *method* baru pada **PageController.java** sebagai berikut:

```
@RequestMapping("/greeting")
public String greeting (@RequestParam(value = "name") String name, Model model)
{
   model.addAttribute ("name", name);
   return "greeting";
}
```

Jalankan aplikasi dan buka **localhost:8080/greeting?name=chanek**, hal ini akan menampilkan page sebagai berkut:



Parameter passing dilakukan dengan GET *request* yang mengambil *value* dari GET dan kemudian menambahkannya ke dalam sebuah objek *Model* dan mengirimkannya ke *View*.

Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

Andre Gema Syahputra
1506689282
Kelas B

Latihan Request Parameter

1. Ubah nilai anotasi RequestMapping dari "/greeting" menjadi "/hello/greeting"
   Buka **localhost: 8080/hello/greeting?name=chanek**
   Pertanyaan: apakah hasilnya?

   Aplikasi berjalan seperti sebelumnya, menampilkan *page* dengan tulisan "Selamat datang chanek!"

2. Akses **localhost: 8080/greeting**
   Pertanyaan: Apakah hasilnya?

   Terjadi *error* dikarenakan tidak adanya *parameter* 'name'. Hal ini dikarenakan adanya parameter 'name' bersifat *required*.

   Ubah header method greeting menjadi seperti berikut:

```
public String greeting (@RequestParam(value = "name", required = false) String
name, Model model)
```

   Stop Spring Boot yang sedang berjalan, run kembali, buka **localhost: 8080/greeting**
   Pertanyaan: Apakah hasilnya?

   Hasilnya memunculkan page dengan tulisan "Selamat datang null!" Hal ini dikarenakan 'name' tidak lagi bersifat *required*, tetapi tidak ada sehingga yang ditampilkan hanyalah *null* sebagai *defaultValue*.

3. Ubah header method greeting menjadi seperti berikut

```
public String greeting (@RequestParam(value = "name", required = false,
defaultValue = "dunia") String name, Model model)
```

   Stop Spring Boot yang sedang berjalan, run kembali, dan buka **localhost: 8080/greeting.**
   Pertanyaan: apakah hasilnya?

   Hasilnya memunculkan page dengan tulisan "Selamat datang dunia!" Hal ini dikarenakan dengan mengganti *defaultValue* dari 'name' menjadi "dunia", apabila 'name' tidak ada, yang ditampilkan adalah "dunia", bukan *null.*

Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

Andre Gema Syahputra
1506689282
Kelas B

4. Perhatikan bahwa pada berkas greeting.html, tag paragraf yang kita tambahkan adalah sebagai berikut:

```
<p th:text="'Selamat datang ' + ${name} + '!'">Sapaan untuk user</p>
```

Pertanyaan: Mengapa tulisan “Sapaan untuk user” tidak pernah muncul?

Atribut **th:text** meng-*override* tulisan dalam *tag* tersebut, sehingga teks “Sapaan untuk user” tidak pernah muncul.

Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

Andre Gema Syahputra
1506689282
Kelas B

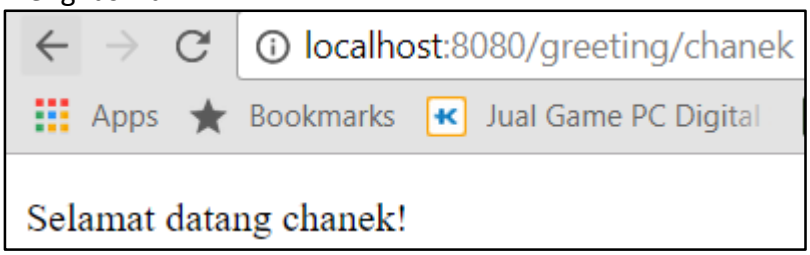
Path Variable

Selanjutnya, *tutorial* meng-eksplor cara lain untuk *data passing* dengan menggunaan *path variable*.

Pertama, tambahkan *method* "greetingPath" pada **PageController.java** sebagai berikut:

```
@RequestMapping("/greeting/{name}")
public String greetingPath (@PathVariable String name, Model model)
{
        model.addAttribute("name", name);
        return "greeting";
}
```

Jalankan aplikasi dan buka **localhost:8080/greeting/chanek** pada *browser* yang akan menghasilkan:



Berbeda dengan GET *request*, cara ini mengambil data dari *path* yang dituju.

Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

Andre Gema Syahputra
1506689282
Kelas B

Latihan Path Variable

1. Akses **localhost:8080/greeting/**

   Pertanyaan: Apa hasilnya?

   Hasilnya memunculkan page dengan tulisan “Selamat datang dunia!”. Hal ini dikarenakan tidak adanya “nama” yang di-*pass*.

   Ubah method greetingPath menjadi seperti berikut

```
@RequestMapping(value = {"/greeting", "greeting/{name}"})
public String greetingPath(@PathVariable Optional<String> name, Model model) {
        if (name.isPresent()) {
                model.addAttribute("name", name.get());
        } else {
                model.addAttribute("name", "dengklek");
        }
        return "greeting";
}
```

   Akses **localhost:8080/greeting/**

   Pertanyaan: Apa hasilnya?

   Terjadi *error* “*Ambiguous handler methods*” yang dikarenakan *RequestMapping* /greeting pada dua buah *method*.

   Akses localhost:8080/greeting/chanek

   Pertanyaan: Apa hasilnya?

   Hasilnya akan menampilkan “Selamat datang chanek!”. *Passing data* berhasil dilakukan.

Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

Andre Gema Syahputra
1506689282
Kelas B

Latihan Perkalian

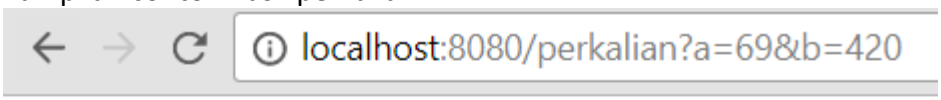
1. Berikut *template* yang saya buat dengan nama **peerkalian.html**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Perkalian</title>
    </head>
    <body>
        <h1>Perkalian</h1>
        <h2 th:text="${a} + ' x ' + ${b} + ' = ' + ${c}">Hasil perkalian</h2>
    </body>
</html>
```

2. Berikut tambahan *method* perkalian pada **PageController.** Dengan RequestMapping perkalian yang memanfaatkan cara GET *request* untuk *data passing*. *Method* menerima *parameter* a dan b yang bersifat tidak *required* dan dengan *defaultValue* 0. Kemudian keduanya dikalikan untuk menghasilkan variabel c. Ketiganya kemudian ditambahkan ke dalam model untuk ditampilkan ke *view* pada *template* perkalian

```
@RequestMapping("/perkalian")
public String perkalian(@RequestParam(value = "a", required = false, defaultValue = "0") int a,
        @RequestParam(value = "b", required = false, defaultValue = "0") int b, Model model)
{
    int c = a * b;
    model.addAttribute("a", a);
    model.addAttribute("b", b);
    model.addAttribute("c", c);
    return "perkalian";
}
```

3. Tampilan contoh hasil perkalian



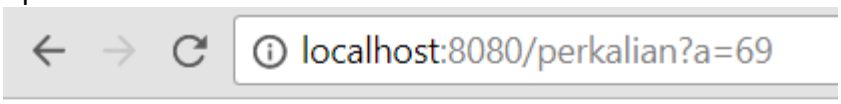
Perkalian

69 x 420 = 28980

Arsitektur dan Pemrograman Aplikasi Perusahaan
Tutorial 2 – Menggunakan Routing dan Controller dalam Project Spring Boot

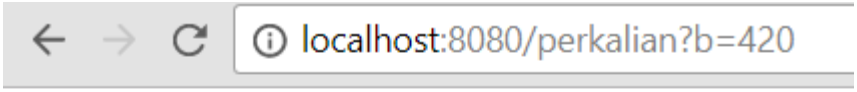
Andre Gema Syahputra
1506689282
Kelas B

4. Apabila a atau b tidak ada



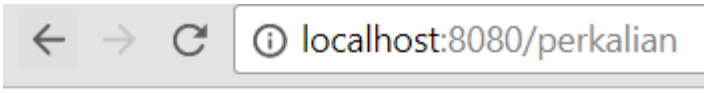
Perkalian

69 x 0 = 0



Perkalian

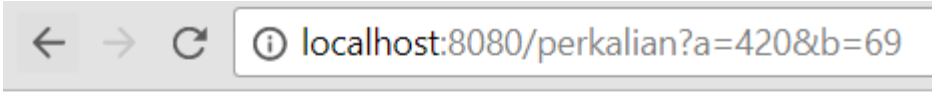
0 x 420 = 0



Perkalian

0 x 0 = 0

5. Contoh hasil tampilan



Perkalian

420 x 69 = 28980