WRITE UP TUTORIAL 2 ADPAP

Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

YANG DIPELAJARI PADA TUTORIAL 2

Pada tutorial 2 mempelajari tentang penggunaan *Routing* dan *Controller* dalam *Project Spring Boot. Spring Boot* sendiri merupakan *web framework* menggunakan Java. Hal pertama yang dilakukan pada tutorial 2 mirip dengan tutorial sebelumnya yaitu membuat *Project Hello World.* Di *Project Hello World* ini terdapat halaman *controller* yang berformat Java yaitu PageController.java. Berikut ini adalah isi dari PageController.java untuk *Project Hello World* :

```java
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";
        }
}
```

Keterangan :

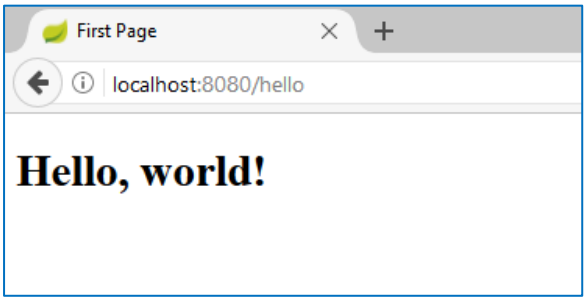
Annotation ***@Controller*** menunjukan bahwa *class* PageController.java adalah sebuah *Controller.*

Annotation ***@RequestMapping("/hello")*** menunjukkan bahwa ketika ada *request* HTTP dengan format *path/hello* maka akan dipanggil method index().

Setelah *Controller* disiapkan selanjutnya membuat file HTML yaitu hello.html yang berisi konten yang akan ditampilkan pada *localhost* dengan mengakses *path* berdasarkan anotasi *@RequestMapping* pada *Controller.* Berikut ini adalah isi dari hello.html untuk *Project Hello World* :

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
       <head>
              <title>First Page</title>
       </head>
       <body>
              <h1>Hello, world!</h1>
       </body>
</html>
```

Kemudian *Project* dijalankan dengan klik kanan pada nama project di window *Project Explorer > Run As… > Spring Boot App*. Lalu pada *browser* buka **localhost:8080/hello**, maka akan muncul tampilan seperti berikut :



WRITE UP TUTORIAL 2 ADPAP

Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

Setelah mempelajari mengenai *Controller* dan *RequestMapping* pada *Project Hello World,* pada tutorial 2 mempelajari tentang ***Request Parameter (Query String)***. Jika pada tutorial sebelumnya menggunakan *query string* untuk melakukan pengiriman data ke *server PHP,* maka di tutorial 2 pengiriman data dilakukan dengan memakai Spring Boot.

Untuk melakukan *request parameter* pada tutorial kali ini buat terlebih dahulu file html untuk menampilkan konten yaitu greeting.html yang isinya seperti berikut :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
	<head>
		<meta charset="UTF-8"></meta>
		<title>Second Page</title>
	</head>
	<body>
		<p th:text="'Selamat datang ' + ${name} + '!'">Sapaan untuk user</p>
	</body>
</html>
```

Setelah itu, pada PageController.java ditambahkan *method* seperti berikut :

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

Keterangan:

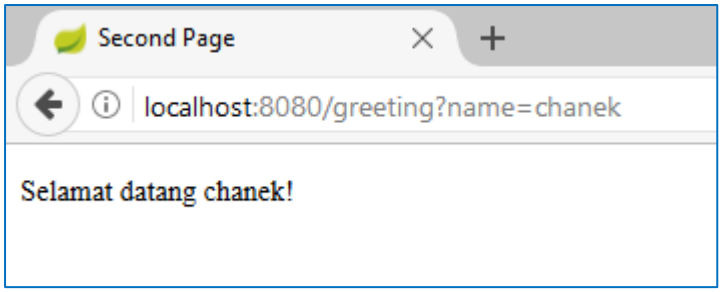
@RequestParam(value = "name") menangkap (GET) parameter *name* dari *query string.*

Object Model untuk melakukan *passing* nilai ke *view.*

Serta tambahkan *import :*

- `import org.springframework.ui.Model;`
- `import org.springframework.web.bind.annotation.RequestParam;`

Kemudian *run* kembali *project,* dan pada *browser* buka **localhost:8080/greeting?name=chanek** dan tampilan yang akan muncul adalah seperti berikut :



WRITE UP TUTORIAL 2 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

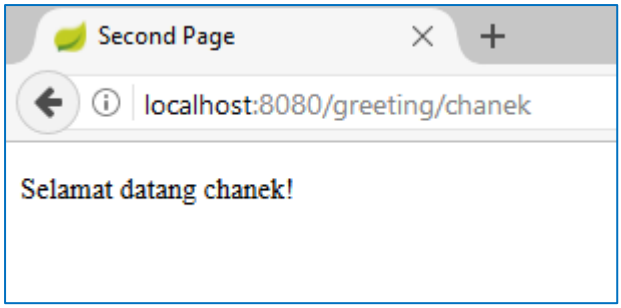
Selain *Request Parameter,* terdapat cara lain untuk melakukan *passing* data dari URL yaitu dengan menggunakan ***Path Variable***. Dengan menggunakan *path variable,* ada perbedaan penulisan URL *localhost* pada browser. Kita tidak perlu lagi membuka **localhost:8080/greeting?name=chanek** , tetapi dengan mengetikkan **localhost:8080/greeting/chanek.** Cara menggunakan *Path Variable* adalah dengan menambahkan *method* pada PageController.java seperti berikut ini :

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

Serta tambahkan *import :*

- `import org.springframework.web.bind.annotation.PathVariable;`

Pada anotasi *@RequestMapping* kita dapat langsung memasukan *path variable* {name} sehingga ketika *run* dan pada *browser* membuka **localhost:8080/greeting/chanek** tampilan yang akan muncul adalah sebagai berikut :



WRITE UP TUTORIAL 2 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

LATIHAN DAN MENJAWAB PERTANYAAN

- **LATIHAN *PROJECT HELLO WORLD***

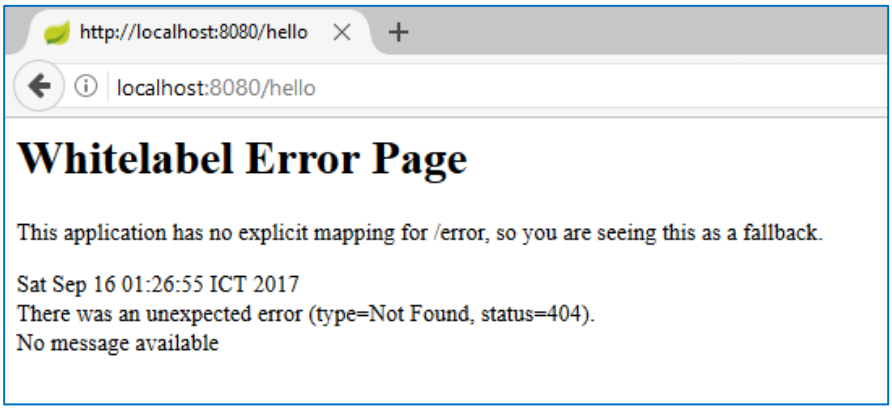
1. Ganti baris tersebut menjadi @RequestMapping("/hello123")

```
@RequestMapping("/hello123")
public String index() {
    return "hello";
}
```

Pertanyaan :
Apakah *compile error*?
⇨ Tidak terjadi *compile error.*

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

⇨ Pada *browser* muncul tampilan bertuliskan ***Whitelabel Error Page.*** Terdapat pesan yang menunjukan bahwa *mapping* yang dituju tidak berhasil ditemukan sehingga memunculkan pesan *error.*



Kembalikan RequestMapping menjadi /hello

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

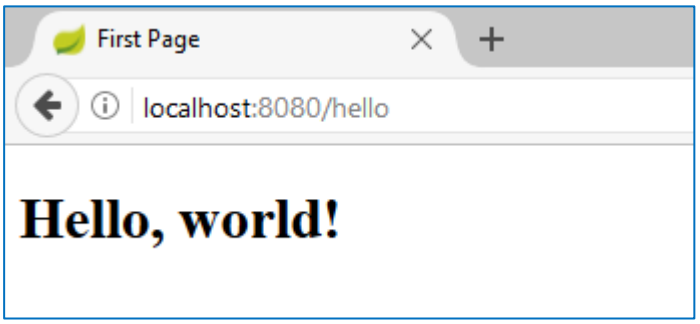
```
@RequestMapping("/hello")
public String hello() {
    return "hello";
}
```

Pertanyaan :
Apakah compile error?

⇨ Tidak terjadi *compile error*

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

⇨ Pada *browser* tidak memunculkan pesan *error.* Tampilan muncul seperti seharusnya atau memunculkan *output* yang benar seperti sebelum diubah nama *method*. Pada kasus ini, perubahan nama *method* tidak mempengaruhi program yang dijalankan.



Kembalikan nama method menjadi index()

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

```
@RequestMapping("/hello")
public String index() {
    return "hello123";
}
```

Pertanyaan :
Apakah compile error?
⇨ Tidak terjadi *compile error.*

Jika tidak, Stop Spring Boot yang sedang berjalan, run kembali dan buka localhost: 8080/hello apakah page hello sebelumnya masih muncul?
⇨ Pada *browser* muncul tampilan bertuliskan ***Whitelabel Error Page.*** Terdapat pesan yang menunjukan bahwa kemungkinan *template* tidak ada atau tidak bisa diakses.

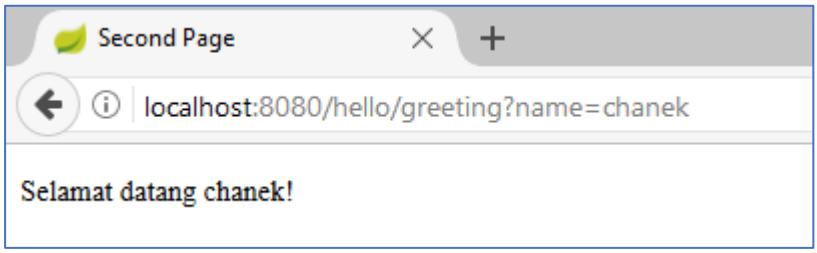
Kembalikan return type menjadi “hello”

Pertanyaan:
Menandakan apakah String yang di-return tersebut?
⇨ String yang di-*return* tersebut menandakan *file template* yang akan diakses. Pada kasus ini, kita mempunyai file hello.html sehingga seharusnya String yang di-*return* adalah ”hello” bukan “hello123” .

WRITE UP TUTORIAL 2 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

- **LATIHAN *REQUEST PARAMETER***
    1. Ubah nilai anotasi RequestMapping dari "/greeting" menjadi "/hello/greeting" Buka **localhost: 8080/hello/greeting?name=chanek**

       Pertanyaan :
       apakah hasilnya?
       ⇨ Hasilnya normal seperti *output* yang seharusnya muncul.

       Kembalikan request map menjadi @RequestMapping("/greeting")

    2. Akses **localhost: 8080/greeting**
       Pertanyaan :
       Apakah hasilnya?
       ⇨ Hasilnya memunculkan pesan *error* yang menyatakan bahwa tidak ada parameter *name* yang dimasukkan.

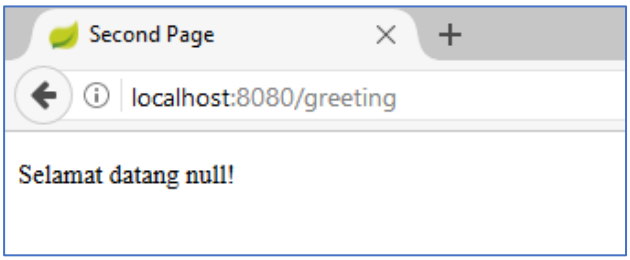
       

Ubah header method greeting menjadi seperti berikut

```
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 adalah seperti gambar dibawah ini, yaitu memunculkan *text* "Selamat datang null!". Hal ini disebabkan karena parameter *required* yang bernilai false tidak mewajibkan kita perlu mengisi *name* seperti sebelumnya. Namun nilai *default* yang dimunculkan saat tidak memasukkan *name* adalah ***null***.

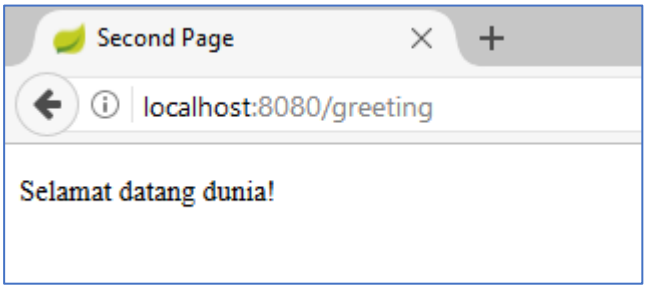


3. Ubah header method greeting menjadi seperti berikut

```
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 teks "Selamat datang dunia!". Hal ini disebabkan karena defaultValue yang diisi dengan String "dunia". Jika pada nomor sebelumnya ketika kita buka **localhost:8080/greeting** defaultValue adalah *null* karena tidak kita isi dengan String, maka pada kasus kali ini defaultValue diisi dengan String "dunia".



WRITE UP TUTORIAL 2 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

4. Perhatikan bahwa pada berkas greeting.html, tag paragraph 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?

⇨ Tulisan “Sapaan untuk user” tidak pernah muncul karena value **attribute th:text** di *tag* p mensubstitusi / menggantikan tulisan yang terdapat dalam *tag* (antara *tag* pembuka dan *tag* penutup). Itulah yang menyebabkan tulisan “Sapaan untuk user” tidak pernah muncul.

Sumber : http://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html

WRITE UP TUTORIAL 2 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

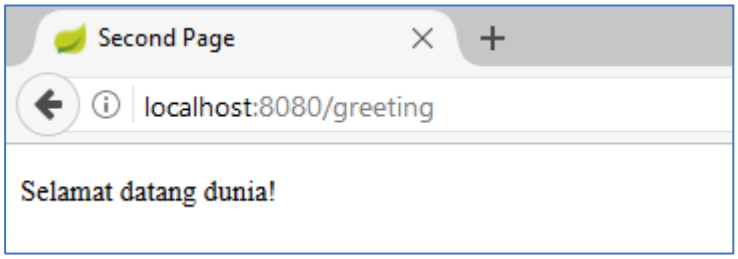
- **LATIHAN *PATH VARIABLE***

1. Akses localhost:8080/greeting/

   Pertanyaan:

   Apa hasilnya?

   ⇨ Hasilnya sama seperti pada nomor 3 latihan *Request Parameter* yaitu memunculkan *text* "Selamat datang dunia!". Karena tidak ada pengubahan lagi pada *method* yang mempunyai ***@RequestMapping(/greeting)*** .



   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?

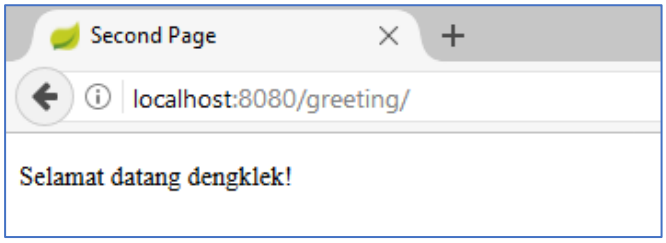
   ⇨ Hasilnya memunculkan pesan *error* pada *browser.* Hal ini terjadi karena terjadi keambiguan karena *value request mapping* yang sama di dua buah *method* yang berbeda yaitu *method* ***greeting*** dan *method* ***greetingPath.*** Untuk mendapatkan *output* yang tidak memunculkan pesan *error* kita bisa menjadikan *method* ***greeting*** sebagai *comment* terlebih dahulu.

*) Menjadikan *method greeting* sebagai *comment,* kemudian di *run* kembali dan buka **localhost:8080/greeting/**

```
    /*@RequestMapping("/greeting")
    public String greeting (@RequestParam(value = "name" , required = false, defaultV
        model.addAttribute("name",name);
        return "greeting";
    }*/

    @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";
    }
```

Maka hasilnya akan menampilkan teks “Selamat datang dengklek!” seperti pada *screenshot* berikut :



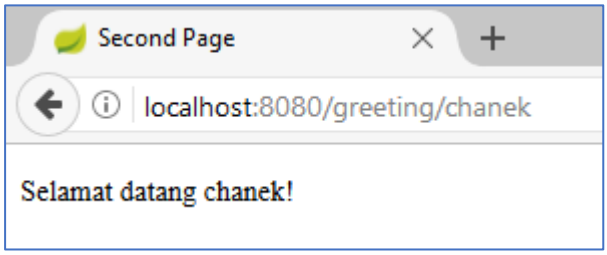
Hal ini disebabkan karena kondisi masuk ke bagian ***else*** yaitu ketika tidak ada *input name.* Sehingga *name* diisi oleh “dengklek”.

Akses localhost:8080/greeting/chanek

Pertanyaan :

Apa hasilnya?

⇨ Hasil yang ditampilkan pada *browser* adalah teks “Selamat datang chanek!” seperti pada *screenshot* berikut :



Hal ini disebabkan karena kondisi masuk ke bagian ***if(name.isPresent())*** yaitu ketika terdapat *input name.* Sehingga *name* diisi oleh nama yang ada pada *path* yaitu “chanek”.

WRITE UP TUTORIAL 2 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

LATIHAN PERKALIAN

1. **Buatlah satu halaman baru untuk melakukan operasi perkalian.**

   Halaman perkalian.html akan menampilkan a, b, dan c dari PageController.java.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8"></meta>
        <title>Perkalian</title>
    </head>
    <body>
        <h1>Perkalian</h1>
        <h2 th:text="${a} + ' x ' + ${b} + ' = ' + ${c}" >Hasil Perkalian</h2>
    </body>
</html>
```

   Pada PageController.java tambahkan *method 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 ) {
        model.addAttribute("a",a);
        model.addAttribute("b",b);
        int c = a * b;
        model.addAttribute("c",c);
        return "perkalian";
    }
```

2. **Halaman ini menerima 2 buah parameter dari query string yaitu a dan b.**

```
    @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 ) {
```

   Parameter a dan b diterima oleh :
   @RequestParam(value = “a”, required = false, defaultValue = “0”) int a (untuk parameter a)
   @RequestParam(value = “b”, required = false, defaultValue = “0”) int b (untuk parameter b)

3. **Pada view tampilkan hasil perkalian a dan b.**
   Hasil perkalian ditampilkan lewat perkalian.html adalah ${c} yand didapat dari *method* perkalian di PageController.java.

```
        int c = a * b;
        model.addAttribute("c",c);
        return "perkalian";
    }
```

   Perhitungan perkalian pada PageController.java di *method* perkalian.

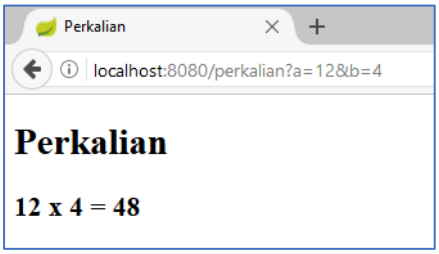
```
<h2 th:text="${a} + ' x ' + ${b} + ' = ' + ${c}" >Hasil Perkalian</h2>
```

   ${a} adalah input untuk parameter a
   ${b} adalah input untuk parameter b
   ${c} adalah hasil perkalian

WRITE UP TUTORIAL 2 ADPAP

Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

Contoh Perkalian:



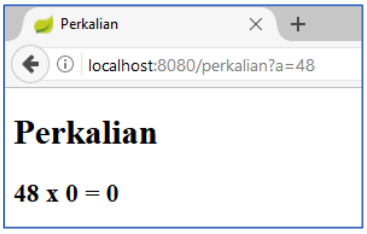
4. **Jika a atau b tidak ada, nilai defaultnya 0.**

Untuk *handle* masalah ini, pada *method* perkalian ditambahkan parameter ***required = false*** dan ***defaultValue = "0"*** untuk a dan b.

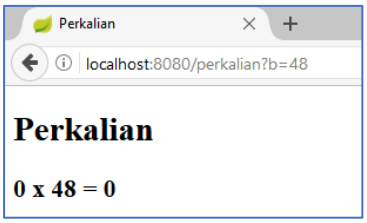
```
35    @RequestMapping("/perkalian")
36    public String perkalian (@RequestParam(value = "a", required = false, defaultValue = "0") int a,
37                             @RequestParam(value = "b", required = false, defaultValue = "0") int b,
38                             Model model ) {
```

Contoh perkalian :

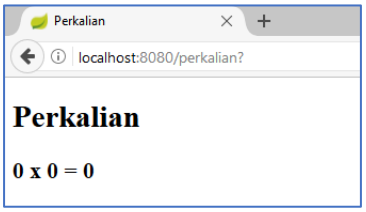
a ada, b tidak ada



b ada, a tidak ada



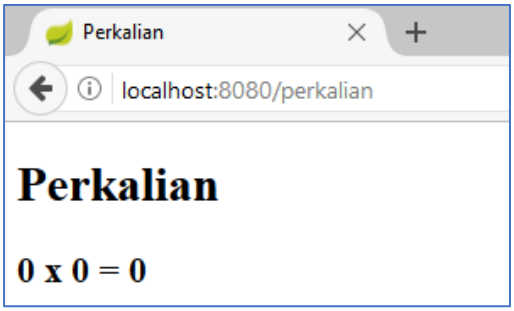
a dan b tidak ada



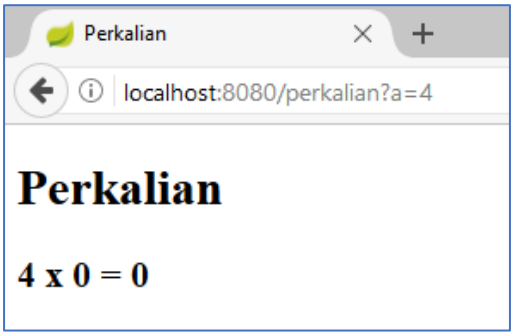
WRITE UP TUTORIAL 2 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

5. Contoh Hasil Tampilan

Test case : **localhost:8080/perkalian**



Test case : **localhost:8080/perkalian?a=4**



Test case : **localhost:8080/perkalian?a=4&b=5**

WRITE UP TUTORIAL 2 ADPAP

Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

KESIMPULAN

Kesimpulan dari tutorial 2 mata kuliah ADPAP ini adalah mempelajari lebih lanjut mengenai *Spring Boot* , mempelajari penggunaan anotasi yang tersedia dan penggunaannya, mempelajari Object yang ada dan penggunaannya, serta menyusun program sederhana menghitung perkalian dua bilangan.