Project Hello World

Ringkasan: Pada project hello world kita belajar membuat tampilan sederhana “Hello World!”. Untuk membuat tampilan hello world langkah – langkah yang harus dilakukan adalah seperti dibawah ini.

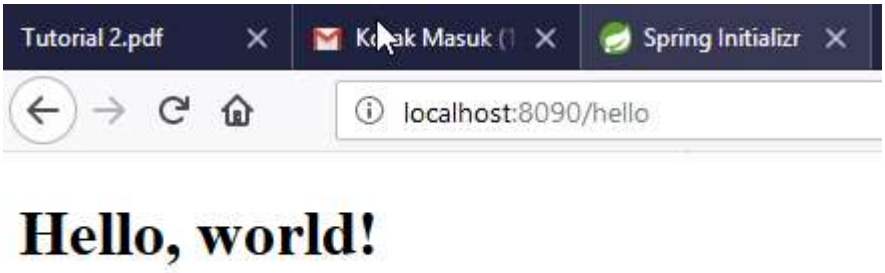
1. Buat PageController di controller. Controller dalam mvc berfungsi sebagai logic dari aplikasi. Isilah dengan kode program seperti dibawah ini.

```
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 hello()
    {
        return "hello";
    }
}
```

   @RequestMapping(“hello”) mengarahkan aplikasi agar dapat diakses dengan url **localhost:8080/hello.** Sehingga ketika ada request url /hello akan diarahkan ke method hello. Kode return “hello” mengarahkan nya ke file hello.html di resources/templates.
2. Buat file hello.html di resources/templates. Isilah dengan kode program seperti dibawah ini.

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

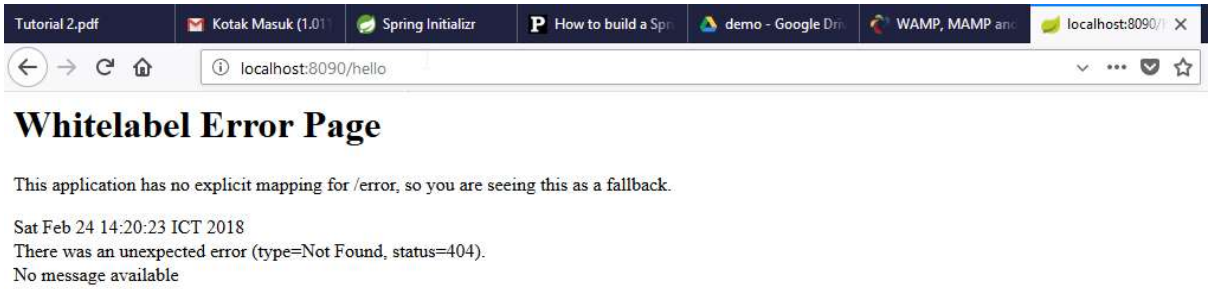
3. Jalankan aplikasi dan lihat di peramban web.



Hello, world!

Latihan Project Hello World

1. Ganti baris menjadi @RequestMapping("/hello123")
   Pertanyaan:
   Apakah compile error?
   Jawab: Terjadi error karena routing nya berubah. Url yang diakses di browser adalah /hello sementara url yang ditulis di controller adalah /hello123.



2. Ganti nama method index() dengan nama method hello()
   Pertanyaan:
   Apakah compile error?
   Jawab: Tidak terjadi error, nama *method* tidak berpengaruh terhadap eksekusi di nama *routing* nya.
3. Ganti string return type menjadi return “hello123”
   Pertanyaan:
   Apakah compile error?
   Jawab: Terjadi error, return “hello” menandakan bahwa “hello” menuju kepada file yang bertempat di resources/templates dengan nama “hello”. Apabila diganti menjadi “hello123” maka controller akan mencari file dengan nama tersebut jika tidak ditemukan maka akan dikembalikan dengan status error.

Request Parameter (Query String)

Ringkasan: pada tutorial ini kita belajar membuat method dengan menambahkan parameter. Selain itu kita belajar membuat template engine dengan thymeleaf. Untuk membuat method dengan parameter langkah-langkah yang harus dilakukan adalah sebagai dibawah ini.

1. Buat page baru dengan nama greeting.html pada folder templates dengan kode program sebagai berikut ini.

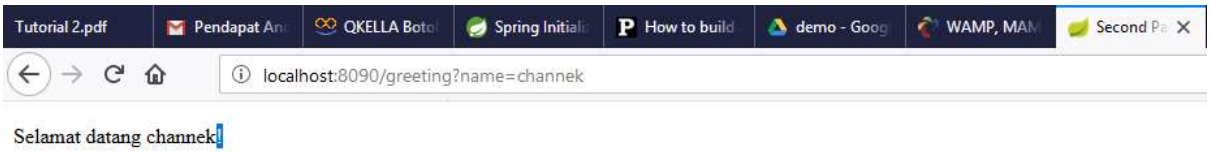
2. Selanjutnya pada PageController tambahkan import berikut:

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
```

Serta tambahkan methode berikut ini:

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

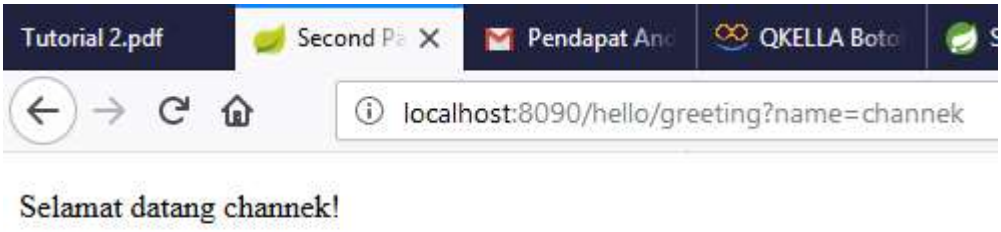
3. Buka browser lalu coba masukan **localhost:8090/greeting?name=chanek**



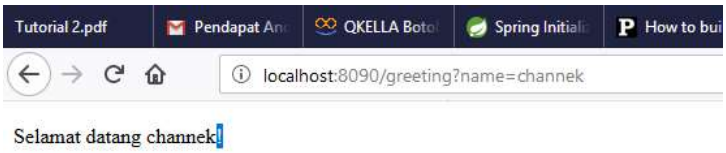
Selamat datang channek!

Latihan Request Parameter

1. Ubah nilai anotasi RequestMapping dari "/greeting" menjadi "/hello/greeting"
   Buka **localhost:8090/hello/greeting?name=channek**
   Pertanyaan: apakah hasilnya?



2. Akses localhost:8090/greeting
   Pertanyaan: apakah hasilnya?

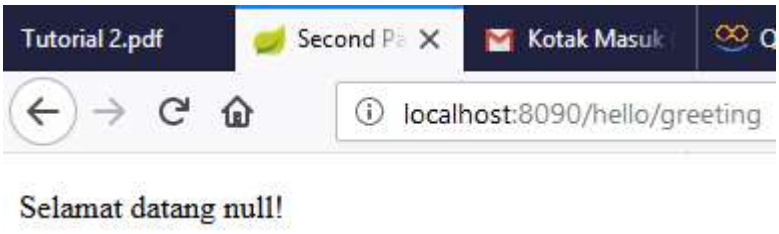


   Jawab: hasilnya sama dengan sebelumnya karena routing nya diganti ke seperti semula.

3. Tambahkan header method greeting menjadi seperti berikut ini.

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

   Apakah hasilnya.



   Hasil dari perubahan tersebut adalah menampilkan pesan selamat datang null!. Tambahan kode required=false artinya method tersebut tidak wajib melewatkan parameter.

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

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

   Pertanyaan: mengapa tulisan “sapaan untuk user” tidak pernah muncul?

karena kita menggunakan thymeleaf yang merupakan template engine untuk spring boot. Menurut situs resmi nya thymeleaf menggunakan xml tags dan atribut dalam eksekusinya dan tidak akan mengeksekusi kode yang ada diantara tag misal dalam kasus ini kode didalam tag sapaan untuk user tidak akan dieksekusi.

path variable

Ringkasan: path variable merupakan alternatif penggunaan url misalnya **/user/delete?id={id}.** Kita bisa menggunakan url **/user/delete/{id}**. Langkah yang harus dilakukan adalah seperti dibawah ini.

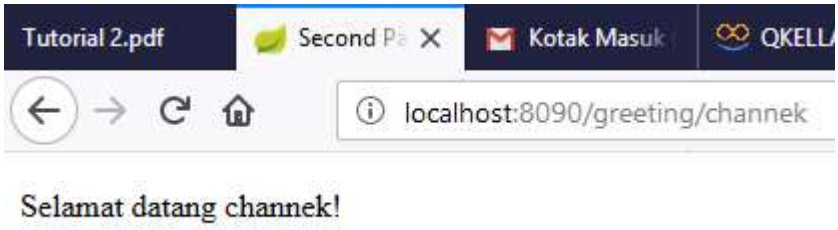
1. PageController tambahkan import berikut ini.

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

Dan tambahkan method greetingPath pada PageController.

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

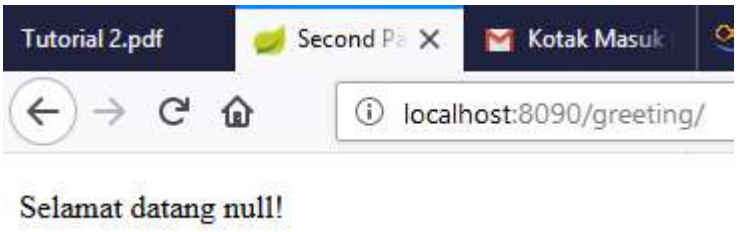
2. Buka **localhost:8090/greeting/chanek** apakah hasilnya?



Dalam spring boot, {name} merupakan path variable yang dapat diolah oleh controller.

Latihan Path Variable:

1. Akses locahost:8090/greeting/
   Pertanyaan: Apakah hasilnya?



Ubah method greetinPath menjadi seperti berikut.

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

Dan tambahkan import

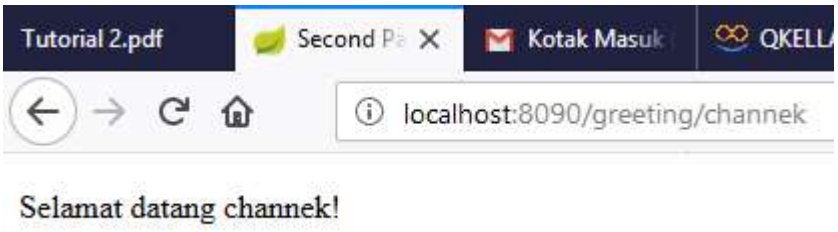
```
import java.util.Optional;
```

Akses localhost:8090/greeting/

Pertanyaan: Apa hasilnya?

hasilnya ambigous karena terdapat lebih dari satu requestmapping dengan nama /greeting. localhost:8080/greeting/chanek akan menampilkan Selamat datang chanek!

Buka **localhost:8090/greeting/chanek** apakah hasilnya?



Latihan

1. Tambahkan method di PageController dengan kode berikut ini.

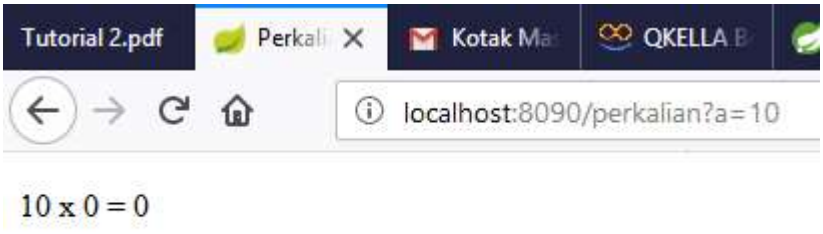
```
@RequestMapping("/perkalian")
public String perkalian(@RequestParam(value="a", required = false) Optional<String> a,
                        @RequestParam(value="b", required = false) Optional<String> b,
                        Model model)
{

    if(a.isPresent()) {
        model.addAttribute( s: "a", a.get());
    }else {
        model.addAttribute( s: "a",  o: 0);
    }
    if(b.isPresent()) {
        model.addAttribute( s: "b", b.get());
    }else {
        model.addAttribute( s: "b",  o: 0);
    }
    return "perkalian";
}
```

2. Tambahkan file di resources/templates

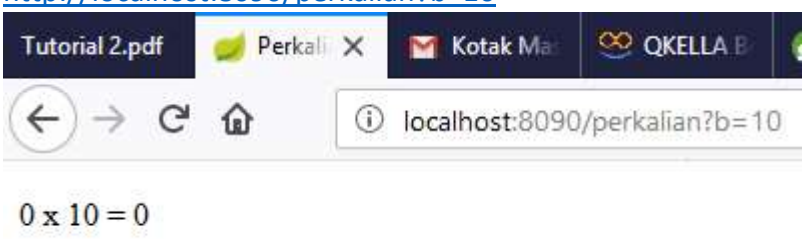
```
eController.java × perkalian.html × greeting.html × DemoApplication.java ×
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Perkalian</title>
    </head>
    <body>
        <p th:text="'' + ${a} + ' x ' + ${b} +' = ' + ${a}*${b} + ' ''></p>
    </body>
</html>
```

3. Jalankan method perkalian.

   http://localhost:8090/perkalian?a=10



   http://localhost:8090/perkalian?b=10



http://localhost:8090/perkalian?a=10&b=10