



Menggunakan Routing dan Controller dalam Project Spring Boot

Versi: 1 (20 Februari 2018)

Outline:

- Membuat dan menggunakan Controller pada Spring Boot
- Pengenalan dasar routing pada Spring Boot

Requirements:

Sebelum memulai tutorial ini pastikan komputer Anda sudah terinstall JDK, Eclipse, Maven, dan Spring Tools Suite.

Penjelasan

Pengenalan Singkat Spring Boot

Pada tutorial kali ini Anda akan mulai menggunakan Spring Boot untuk mengerjakan tutorial. Spring Boot akan terus digunakan seterusnya pada setiap tutorial, tugas, UTS, dan tugas akhir.

Spring Boot merupakan sebuah web framework menggunakan Java. Framework menggunakan Java banyak digunakan untuk membuat aplikasi berskala besar. Beberapa perusahaan besar seperti Traveloka, Go-Jek, dan Blibli menggunakan Java sebagai back-end language mereka. Salah satu web framework Java yang paling populer adalah Spring.

“Saya pernah mendengar Spring, Spring MVC, dan Spring Boot. Apakah ketiganya sama?”

Ya, sekaligus tidak. Ketiganya berbeda tetapi memiliki kesamaan. Spring MVC merupakan Spring yang telah mengadopsi *architectural pattern* MVC. Sementara Spring Boot merupakan Spring MVC yang telah terkonfigurasi. Pada Spring/Spring MVC Anda perlu melakukan konfigurasi yang cukup rumit agar server dapat berjalan dengan lancar. Pada mata kuliah ini Anda hanya akan fokus bagaimana mengembangkan aplikasi berskala besar menggunakan Spring tanpa perlu pusing memikirkan konfigurasi yang **lebih rumit**. Oleh karena itu, mata kuliah ini menggunakan framework Spring Boot.

Tutorial Routing dan Controller pada Project Spring Boot

Perhatian: Jangan copy-paste kode pada pdf ini ke eclipse/IDE Anda. Kode Anda tidak akan berjalan atau dikenali oleh compiler karena karakter tidak memiliki encoding yang sama.

Project Hello World

Pada tutorial ini Anda akan memulai dengan project Hello World.

Langkah:

1. Pada Eclipse klik File > New > Spring Starter Project, tekan Next
2. Beri nama project, description, package dll.
Pada kolom name isi dengan nama project yang akan dibuat. Pada kolom group, artifact, demo, version, dan description, dan package tidak perlu diganti.
3. Untuk memudahkan pengerjaan project Spring boot, maka membutuhkan beberapa *external library* dalam bentuk *dependency*. Pilih Dependency **Web**, **Thymeleaf**, dan **DevTools**. Anda bisa menggunakan kolom pencarian untuk mencari dependency yang dibutuhkan.
4. Package merupakan suatu cara pengelompokkan/pengorganisasian kelas-kelas dan interface yang saling terkait menjadi suatu unit tunggal dalam library.

Untuk menambahkan package Controller dapat dilakukan dengan cara klik kanan pada package **com.example.demo** > New > Package. Pada package **com.example.demo.controller** buatlah sebuah file Class dengan nama class **PageController.java**. Kemudian isi class tersebut sesuai kode 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";
    }
}
```

package harus berisi sesuai dengan lokasi dari file tersebut.

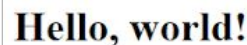
TIPS: Ketika mengetikkan anotasi **@Controller** dan **@RequestMapping** Anda akan menemukan error bahwa anotasi belum ter-import. Untuk melakukan import otomatis

gunakan shortcut Ctrl+Shift+O/Command+Shift+O. Anda tidak perlu lagi menuliskan baris import.

5. Pada folder **resources/templates** tambahkan berkas **hello.html** dengan isi sebagai berikut

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

6. Klik kanan pada nama project di window Project Explorer > Run As... > Spring Boot App
7. Pada browser Anda buka **localhost:8080/hello**
8. Tampilan yang seharusnya Anda dapatkan:



Anotasi **@Controller** di atas deklarasi kelas menandakan bahwa **PageController** merupakan sebuah Controller. Controller merupakan salah satu komponen pada *pattern* MVC. Controller akan merespond HTTP Requests dan mengembalikan View.

Anotasi **@RequestMapping("/hello")** di atas method `index()` menandakan bahwa jika ada request HTTP pada path `/hello` akan dipanggil method `index()`.

Latihan Project Hello World

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

Pertanyaan:

Apakah compile error?

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

Kembalikan `RequestMapping` menjadi `/hello`

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

Pertanyaan:

Apakah compile error?

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

Kembalikan nama method menjadi index()

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

Pertanyaan:

Apakah compile error?

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

Kembalikan return type menjadi "hello"

Pertanyaan:

Menandakan apakah String yang di-return tersebut?

Request Parameter (Query String)

Pada tutorial sebelumnya Anda telah menggunakan query string untuk mengirimkan data ke server PHP. Spring Boot juga dapat mengolah query string.

Langkah:

1. Silakan buat page baru dengan nama **greeting.html** pada folder templates dengan isi sebagai 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>
```

2. Selanjutnya pada PageController tambahkan import berikut:

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

Dan tambahkan method berikut:

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

3. Stop Spring Boot yang sedang berjalan dan run kembali (Tidak perlu melakukan ini kalau sudah menginstall dev tools dan live reload)
4. Buka **localhost:8080/greeting?name=chanek**

Anda baru saja mengirim parameter melalui GET request. Method greeting kemudian menangkap parameter name dari query string dengan menambahkan parameter `@RequestParam(value = "[namaparameter]")` String name sebagai parameter method greeting. Objek Model digunakan untuk passing nilai ke view. Dalam contoh ini ke view `greeting.html`.

Latihan Request Parameter

1. Ubah nilai anotasi RequestMapping dari `"/greeting"` menjadi `"/hello/greeting"`

Buka **localhost:8080/hello/greeting?name=chanek**

Pertanyaan: apakah hasilnya?

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

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

Pertanyaan: Apakah hasilnya?

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?

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?

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?

Path Variable

Cara lain untuk passing suatu data dari URL adalah menggunakan path variable. Contoh dari penggunaan path variable misalkan “/user/delete/{id}” dimana id merupakan path variable. Anda dapat mengirim data melalui path variable ke server pada Spring Boot.

Langkah:

1. PageController tambahkan import berikut:

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

Dalam Spring Boot, {name} merupakan path variable yang dapat diolah oleh controller.

Latihan Path Variable:

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

Pertanyaan: Apa hasilnya?

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", "apap");
    }
    return "greeting";
}
```

dan tambahkan import

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

Akses **localhost:8080/greeting/**

Pertanyaan: Apa hasilnya?

Akses **localhost:8080/greeting/chanek**

Pertanyaan: Apa hasilnya?

Path variable dan request parameter (query string) akan sering digunakan kedepannya dalam membuat request. Pastikan Anda memahami fungsi dan penggunaan kedua hal tersebut. Hal lain seputar penggunaan Spring Boot dapat Anda lihat pada dokumentasi Spring Boot pada tautan berikut <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. Untuk dokumentasi Thymeleaf silakan ke tautan <http://www.thymeleaf.org/documentation.html>. Materi Thymeleaf lebih detail akan dibahas pada perkuliahan.

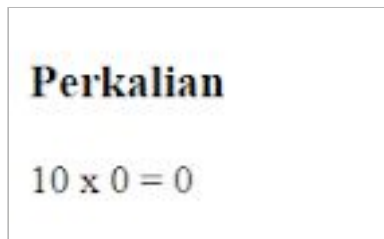
Latihan

1. Buatlah satu halaman baru untuk melakukan operasi perkalian.
2. Halaman ini menerima **2 buah** parameter dari query string yaitu a dan b.
3. Pada view tampilkan hasil perkalian a dan b.
4. Jika a atau b tidak ada, nilai defaultnya 0.
5. Contoh hasil tampilannya

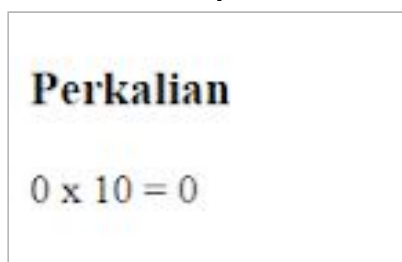
- **localhost:8080/perkalian**



- **localhost:8080/perkalian?a=10**



- **localhost:8080/perkalian?b=10**



- localhost:8080/perkalian?a=10&b=10

Perkalian

$$10 \times 10 = 100$$

Deliverables

Pada tutorial ini, ada beberapa *deliverables* yang Anda perlu kerjakan dan kumpulkan. *Deliverables* tersebut adalah sebagai berikut:

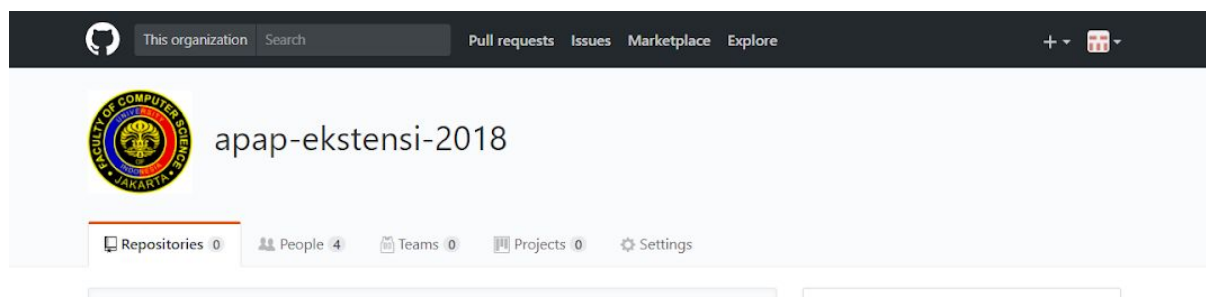
1. *Folder project* Anda yang berisi implementasi dari bagian ‘Perkalian’ yang terdapat dalam tutorial ini. Tidak masalah jika dalam project tersebut juga ada hasil tutorial Anda.
2. *File write-up* berisi penjelasan apa yang Anda pelajari pada tutorial kali ini, yaitu:
 - a. Ringkasan dari materi yang Anda telah pelajari pada tutorial kali ini
 - b. Hasil jawaban dari setiap poin pada bagian tutorial (dapat didukung dengan *screenshot*)

Format txt atau pdf. Masukkan dalam *folder project* dan pastikan *file write-up* juga di-*push* ke repositori GitHub.

Pengumpulan

Buat sebuah **project** di organisasi GitHub /**apap-ekstensi-2018** dengan format nama tutorial2_[NPM]. Contoh: **tutorial2_1501234567**. Push seluruh isi folder project Anda, termasuk file write-up, ke repositori project tersebut. Anda tidak perlu membuat branch baru. Cukup push ke branch master saja.

Perhatikan dalam membuat **public** project tersebut. Pastikan Anda berada di dalam organisasi GitHub /**apap-ekstensi-2018** terlebih dahulu.



Commit yang akan dinilai adalah commit terakhir yang di push ke repositori sebelum deadline. Waktu yang akan dijadikan patokan adalah waktu commit di server GitHub. Pastikan Anda tidak push commit lagi setelah deadline jika tidak ingin terkena penalti.

Deadline

21 Februari 2018, 23:55:00

Penalti

Penalti berlaku untuk kasus:

- Keterlambatan

Nilai total akan ditambahkan -10 poin untuk setiap 1 jam keterlambatan