FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA

Tutorial 2
Arsitektur Pemrograman Aplikasi Perusahaan
Semester Ganjil 2018/2019

Penerapan MVC dengan menggunakan Spring Boot Framework

Pada tutorial ini Anda akan berlatih menggunakan Spring Boot untuk mengerjakan tutorial. Spring Boot akan digunakan seterusnya pada tutorial, tugas, UTS, dan tugas akhir.

Outline:

- Membuat dan menggunakan Controller pada Spring Boot - project VIRAL
- Menerapkan routing pada project Spring Boot
- Latihan

Installation requirements:

- JDK 1.8 atau lebih
- Spring Tool Suite (STS) / Eclipse / Intellij Idea / IDE lainnya
- Maven / Gradle (bagi yang menggunakan IDE selain STS / Eclipse / Intellij IDEA)

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 ini dibuat dengan menggunakan Spring Tool Suite IDE. Oleh karena itu bagi yang menggunakan IDE lainnya dimohon untuk menyesuaikan masing-masing IDE.

Project VIRAL

Langkah-langkah

1. Pada STS, klik menu >> New >> Spring Starter Project
2. Isi bagian project name dengan tutorial-2. Kemudian isi bagian package dengan "**com.example.demo**". Bagian lainnya tidak wajib diisi
3. Pada halaman berikutnya Anda akan memilih *dependency*. Silakan memilih **Web, Thymeleaf, dan DevTool.** Bagi yang menggunakan IDE lain dapat mencari tahu dengan keyword *SpringBoot dependency installation using Maven*
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** dan isi class tersebut sesuai kode berikut:

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

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

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

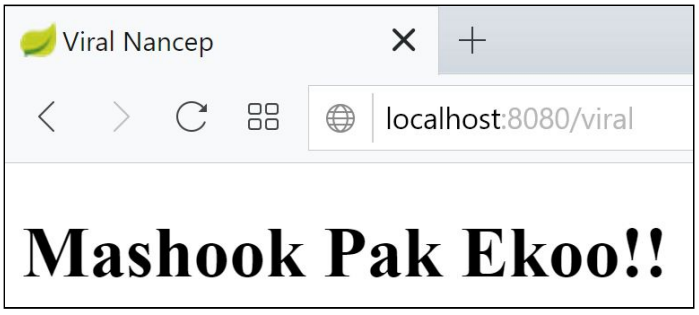
   Package harus berisi sesuai dengan lokasi dari file tersebut

   TIPS: Ketika menggunakan anotasi @Controller dan @RequestMapping Anda akan menemukan error bahwa anotasi belum ter-import. Untuk dapat melakukan import otomatis gunakan shortcut Ctrl+Shift+O/Command+Shift+O. Sehingga Anda tidak perlu lagi menuliskan baris import.

5. Pada folder **resource/templates**, buatlah file **viral.html** dengan isi sebagai berikut

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Viral Nancep</title>
    </head>
    <body>
        <h1>Mashook Pak Ekoo!!</h1>
    </body>
</html>
```

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



Latihan Project Viral

1. Ganti baris @RequestMapping("/viral") menjadi @RequestMapping("/viral2jari"). Kemudian run kembali.
   a. Apakah terjadi compile error? Tidak.
   b. Coba akses localhost:8080/viral, berikan alasan dan jelaskan apa yang terjadi. not found, url berubah.
   c. Apakah Kegunaan @RequestMapping("/viral") untuk menampilkan di url /viral.

   Kembalikan menjadi @RequestMapping("/viral")

2. Ganti nama method index() menjadi nama method viral()
   a. Apakah terjadi compile error? Tidak
   b. Coba akses localhost:8080/viral, berikan alasan dan jelaskan apa yang terjadi. Tidak ada error, method index() hanya nama

   Kembalikan menjadi index()

3. Ganti string return type menjadi menjadi return "viral2jari"
   a. Apakah terjadi compile error? Tidak
   b. Coba akses localhost:8080/viral, berikan alasan dan jelaskan apa yang terjadi. White label error.
   c. Apakah kegunaan return type pada method controller tersebut? Menampilkan file html yang namanya sama.

   Kembalikan return type menjadi "viral"

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-langkah:

1. Buatlah file html baru dengan nama **challenge.html** pada folder templates dengan isi sebagai berikut:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Viral Challenge</title>
    </head>
    <body>
        <p th:text="${name} + ' do you love me' + '?'">In My Feeling</p>
        <p th:text="' Are you riding? '"></p>
    </body>
</html>
```

2. Kemudian tambahkan method berikut pada **PageController**

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

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class PageController {

    @RequestMapping("/viral")
    public String index(){
        return "viral";
    }

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

}
```

3. Hentikan Spring Boot yang sedang berjalan, dan jalankan kembali (tidak perlu dilakukan apabila telah menginstall DevTools dan live reload)

4. Buka browser Anda dan akses **localhost:8080/challenge?name=kiki**

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

Latihan Request Parameter

Setiap ada perubahan pada kode, selalu usahakan run ulang

1. Mengapa tulisan "In My feeling" pada thymeleaf challenge tidak pernah muncul di browser? Karena 'body' dari tag disubstitusi oleh th:text
2. Ubah nilai anotasi Request Mapping "/challenge" menjadi "/viral/challenge". Deskripsikan hasil pada browser dan kembalikan ke bentuk sebelumnya! url berubah sesuai soal.
3. Akses localhost:8080/challenge.
    a. Deskripsikan hasilnya Whitelabel Error karena tidak ada parameter.
    b. Ubah methodnya menjadi seperti di bawah ini, deskripsikan lalu kembalikan ke bentuk awal! Parameter tidak menjadi keharusan. Parameter kosong = null.

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

    c. Ubah methodnya menjadi seperti di bawah ini, deskripsikan lalu kembalikan ke bentuk awal! Seperti jawaban pada soal b, namun ini default value nya "kiki"

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

Path Variable

Terdapat cara lain untuk melemparkan suatu data dari URL dengan menggunakan **path variable.** Untuk dapat mencoba, silahkan ikuti langkah dibawah ini:

Langkah:

1. Tambahkan method challengePath pada **PageController**

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

2. Hentikan program yang sedang berjalan. Kemudian jalankan program.
3. Buka browser Anda dan akses **localhost:8080/challenge/kiki**

Latihan

1. Buatlah sebuah halaman baru sebagai viral generator
2. Halaman ini menerima 2 buah parameter berupa dua angka
3. Pada view menampilkan viral generator berupa kata yang memiliki spesifikasi sebagai berikut:
    a. jika tidak ada parameter hanya menampilkan kata “hm”
    b. parameter pertama mengindikasikan jumlah huruf ‘m’ pada kata “hm”
    c. default kata untuk a = 0 atau a=1 atau b=0 atau b=1 adalah “hm”
    d. parameter kedua mengindikasikan kelipatan kata yang dihasilkan pada poin b
    e. perkata dipisahkan oleh satu spasi
4. Contoh:
    http://localhost:8080/generator

http://localhost:8080/generator?a=0&b=3

http://localhost:8080/generator?a=1&b=3

http://localhost:8080/generator?a=5&b=3

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 'Latihan' yang terdapat dalam tutorial ini. Tidak masalah jika dalam project tersebut juga ada hasil tutorial Anda.
2. *File write-up* berisi hasil jawaban dari setiap poin pada bagian latihan tutorial yang berwarna merah (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-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-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

22 September 2018, 23:59:59

Penalti

Penalti berlaku untuk kasus:

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