

Anindito Izdihardian Wibisono
1506757466
APAP – C

Dalam tutorial kali ini, saya belajar mengenai pembuatan proyek sederhana dengan menggunakan Spring Boot pada IDE Eclipse EE. Saya memahami dan dapat mempraktekkan cara menyiapkan berkas controller serta mengatur request parameter dan path variable.

LATIHAN 1: PROJECT HELLO WORLD

(screenshot ada di halaman-halaman selanjutnya)

1-1 Ganti baris tersebut [`@RequestMapping ("/hello")`] menjadi `@RequestMapping("/hello123")`

The screenshot shows a web browser window on the left and a document editor on the right. The browser window displays a "Whitelabel Error Page" with the following text:

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri Sep 15 20:03:52 ICT 2017
There was an unexpected error (type=Not Found, status=404).
No message available

The document editor on the right shows a document with the following content:

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:

localhost:8080/hello

Hello, world!

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?

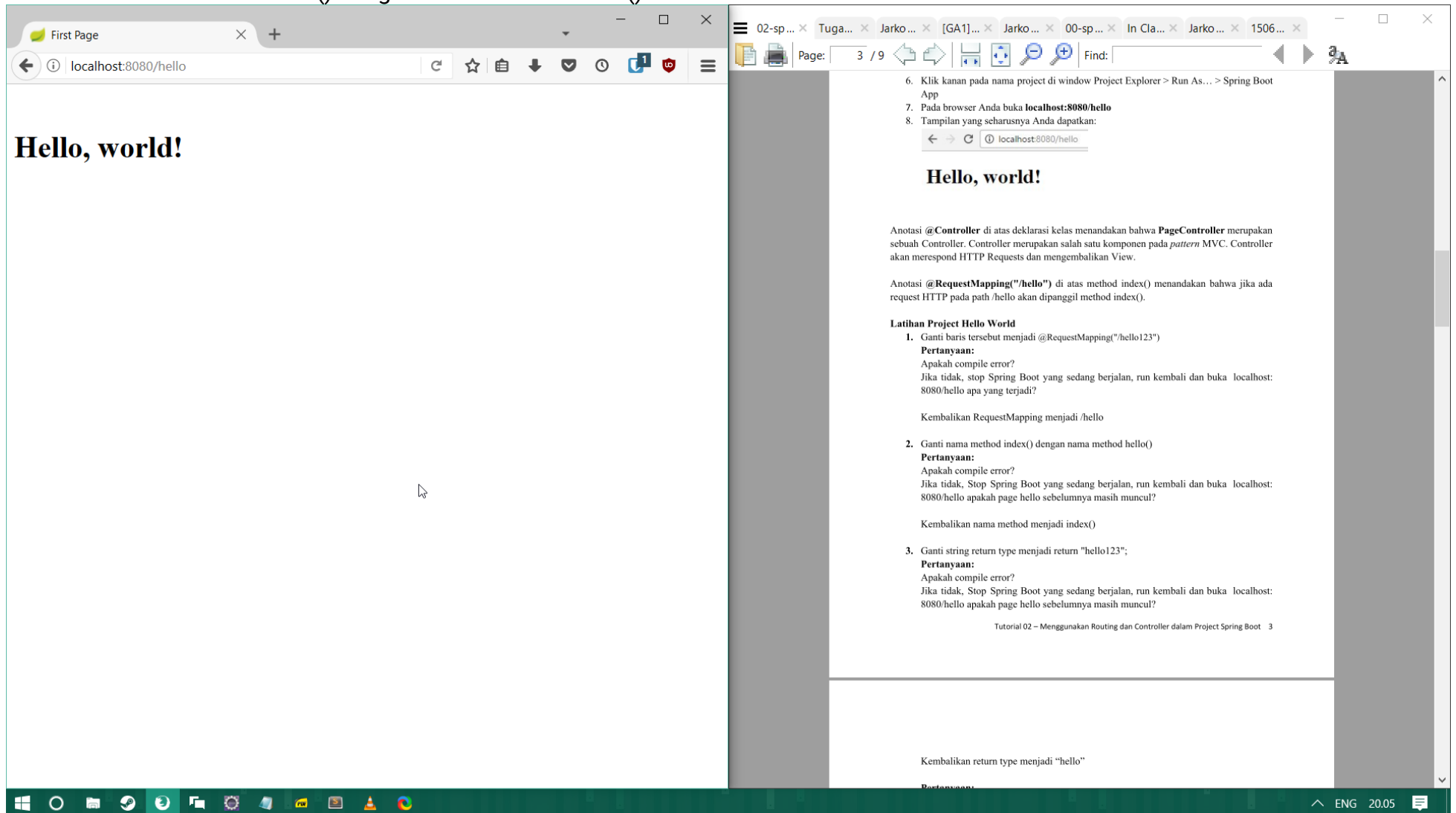
Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 3

Kembalikan return type menjadi `"hello"`

Pertanyaan:

Hasil setelah file PageController.java disave dengan perubahan tersebut dan dijalankan. Tidak ada compile error, namun terjadi Whitelabel Error (404 Not Found).

1-2 Ganti nama method `index()` dengan nama method `hello()`



Hasil setelah file `PageController.java` disave dengan perubahan tersebut dan dijalankan. Tidak ada compile error dan halaman "hello" tetap dapat ditampilkan.

1-3 Ganti string return type menjadi return "hello123"

The screenshot shows a web browser window on the left and a PDF document on the right. The browser window displays a "Whitelabel Error Page" for the URL `localhost:8080/hello`. The error message states: "This application has no explicit mapping for /error, so you are seeing this as a fallback. Fri Sep 15 20:06:44 ICT 2017. There was an unexpected error (type=Internal Server Error, status=500). Error resolving template 'hello123', template might not exist or might not be accessible by any of the configured Template Resolvers". The PDF document on the right is titled "Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 3". It contains text explaining the role of a Controller and RequestMapping, followed by a series of exercises (Latihan Project Hello World) that guide the user through modifying the `index()` method to `hello()` and changing the return type to `"hello123"`. The exercises include questions about compile errors and the state of the application.

localhost:8080/hello

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri Sep 15 20:06:44 ICT 2017
There was an unexpected error (type=Internal Server Error, status=500).
Error resolving template "hello123", template might not exist or might not be accessible by any of the configured Template Resolvers

02-sp... x Tuga... x Jarko... x [GA1]... x Jarko... x 00-sp... x In Cla... x Jarko... x 1506... x

Page: 3 / 9

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?

Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 3

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>`

Hasil setelah file `PageController.java` disave dengan perubahan tersebut dan dijalankan. Tidak ada compile error, namun terjadi Whitelabel Error (500 Internal Server Error). Hal ini karena berdasarkan return string `"hello123"`, proyek mencari `"hello123.html"` untuk ditampilkan di browser, namun file tersebut tidak ada di templates.

LATIHAN 2: REQUEST PARAMETER/QUERY STRING

2-1 Ubah nilai anotasi RequestMapping dari "/greeting" menjadi "/hello/greeting"

The screenshot shows a web browser window on the left and a PDF document on the right. The browser window displays a 'Whitelabel Error Page' for the URL `localhost:8080/greeting?name=chanek`. The error message states: 'This application has no explicit mapping for /error, so you are seeing this as a fallback. Sat Sep 16 10:02:22 ICT 2017. There was an unexpected error (type=Not Found, status=404). No message available'.

The PDF document on the right contains the following steps:

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>
<div th:text="'Selamat datang ' + ${name} + '!'"> Sapaan
    untuk user</div>
</body>
</html>
```
2. Selanjutnya pada PageController 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?

Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 4

Hasil setelah file PageController.java disave dengan perubahan tersebut dan dijalankan. Tidak ada compile error, namun terjadi Whitelabel Error (404 Not Found).

2-2 Akses localhost: 8080/greeting

The screenshot shows a web browser window on the left and a document editor on the right. The browser window displays a 'Whitelabel Error Page' for the URL 'localhost:8080/greeting'. The error message states: 'This application has no explicit mapping for /error, so you are seeing this as a fallback. Sat Sep 16 10:03:33 ICT 2017. There was an unexpected error (type=Bad Request, status=400). Required String parameter 'name' is not present'.

The document editor on the right contains a tutorial for Spring Boot. It includes steps for running the application, accessing the endpoint, and modifying the controller and view files. The tutorial is titled 'Tutorial 02 - Menggunakan Routing dan Controller dalam Project Spring Boot'.

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Sep 16 10:03:33 ICT 2017
There was an unexpected error (type=Bad Request, status=400).
Required String parameter 'name' is not present

Tutorial 02 - Menggunakan Routing dan Controller dalam Project Spring Boot

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?

Hasil setelah file `PageController.java` disave dengan perubahan tersebut dan dijalankan. Tidak ada compile error, namun terjadi Whitelabel Error (400 Bad Request).

The screenshot shows a web browser window on the left and a tutorial document on the right. The browser window displays the URL `localhost:8080/greeting` and the text "Selamat datang null!". The tutorial document on the right is titled "Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot" and contains the following content:

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?

Hasil setelah file `PageController.java` disave dengan perubahan header method `greeting` dan dijalankan. Tidak ada compile error, dan ketika browser menampilkan halaman `greeting`, nilai "null" menggantikan tempat yang seharusnya diisi nama pada parameter.

2-3 Ubah header method greeting (tambahkan atribut "defaultValue" pada header)

The screenshot shows a web browser on the left and a tutorial document on the right. The browser displays the URL `localhost:8080/greeting` and the text "Selamat datang dunia!". The tutorial document, titled "Latihan Request Parameter", contains the following steps:

- 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")`
- 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?
- 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?
- 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:

- Tambahkan method `greetingPath` pada **PageController**

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

Hasil setelah file `PageController.java` disave dengan perubahan tersebut dan dijalankan. Tidak ada compile error, dan teks "dunia" sebagai default/placeholder mengisi tempat yang seharusnya diisi parameter nama.

2-4 Mengapa tulisan “Sapaan untuk user” tidak pernah muncul?

Atribut “th:text” pada file html akan mengatur isi dari atribut tersebut sebagai body/isi tag bersesuaian yang berisi tag “th:text”. Oleh karena itu, tulisan body yang sebenarnya selalu digantikan oleh isi atribut “th:text”.

(ref: <http://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#using-thtext-and-externalizing-text>)

LATIHAN 3: PATH VARIABLE

3-1 Akses localhost:8080/greeting/chanek

The screenshot shows a web browser window on the left and a tutorial document on the right. The browser window displays the URL `localhost:8080/greeting/chanek` and the message "Selamat datang chanek!". The tutorial document on the right contains the following text:

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. 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.

Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 5

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

Akses **localhost:8080/greeting/**
Pertanyaan: Apa hasilnya?

Hasil setelah file `PageController.java` disave dengan tambahan method `greetingPath()` dan dijalankan.

3-2 Akses localhost:8080/greeting/

The image shows a side-by-side comparison of a web browser window and a tutorial document.

Browser Window (Left):

- Tab: Second Page
- Address bar: localhost:8080/greeting/
- Content: Selamat datang dunia!

Tutorial Document (Right):

Page: 5 / 9

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. 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.

Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 5

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

Akses **localhost:8080/greeting/**

Pertanyaan: Apa hasilnya?

Teks yang ditampilkan menggunakan placeholder dari method greeting().

3-3 Akses localhost:8080/greeting/ setelah method greetingPath() diubah

The screenshot shows a web browser window on the left and a document window on the right. The browser window displays a 'Whitelabel Error Page' for the URL 'localhost:8080/greeting/'. The error message states: 'This application has no explicit mapping for /error, so you are seeing this as a fallback.' Below this, it shows the date 'Sat Sep 16 10:11:06 ICT 2017' and the error details: 'There was an unexpected error (type=Internal Server Error, status=500). Ambiguous handler methods mapped for HTTP path 'http://localhost:8080/greeting/': {public java.lang.String com.example.demo.controller.PageController.greeting(java.lang.String,org.springframework.ui.Model), public java.lang.String com.example.demo.controller.PageController.greetingPath(java.util.Optional,org.springframework.ui.Model)}'.

The document window on the right contains a tutorial titled 'Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot'. It lists steps for adding a 'greetingPath' method to the 'PageController' and then accessing 'localhost:8080/greeting/' and 'localhost:8080/greeting/chanek'. The document also includes a 'Latihan Path Variable' section with exercises for accessing 'localhost:8080/greeting/' and 'localhost:8080/greeting/chanek'.

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Sep 16 10:11:06 ICT 2017

There was an unexpected error (type=Internal Server Error, status=500).

Ambiguous handler methods mapped for HTTP path 'http://localhost:8080/greeting/': {public java.lang.String com.example.demo.controller.PageController.greeting(java.lang.String,org.springframework.ui.Model), public java.lang.String com.example.demo.controller.PageController.greetingPath(java.util.Optional,org.springframework.ui.Model)}

Langkah:

1. 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.

Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 5

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

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.

Hasil setelah file PageController.java disave dengan perubahan tersebut dan dijalankan. Tidak ada compile error, namun terjadi Whitelabel Error karena adanya ambiguitas method (greeting atau greetingPath) yang semestinya digunakan Spring Boot.

The image shows a side-by-side comparison of a web browser window and a tutorial document.

Browser Window (Left):

- Address bar: `localhost:8080/greeting`
- Page content: "Selamat datang dengklek!"

Tutorial Document (Right):

Page: 6 / 9

Anda dapat mengirim data melalui path variable ke server pada Spring Boot.

Langkah:

1. 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.

Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 5

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

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`.

Apabila method `greeting()` di-comment, maka akses `localhost:8080/greeting` akan menggunakan placeholder dari method `greetingPath()`, yaitu "dengklek".

3-4 Akses localhost:8080/greeting/chanek setelah method greetingPath() diubah

The screenshot shows a web browser window on the left and a tutorial document on the right. The browser window displays the URL `localhost:8080/greeting/chanek` and the message "Selamat datang chanek!". The tutorial document on the right contains the following content:

Langkah:

1. 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.

Tutorial 02 – Menggunakan Routing dan Controller dalam Project Spring Boot 5

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

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`.

Hasil setelah file `PageController.java` disave dengan perubahan tersebut dan dijalankan. Tidak ada compile error, dan halaman greeting ditampilkan sesuai dengan input path nama ("chanek").