Gheafany W. P.
1506721756

Tutorial 3: Menggunakan Model dan Service pada Project Spring Boot

Membuat Service

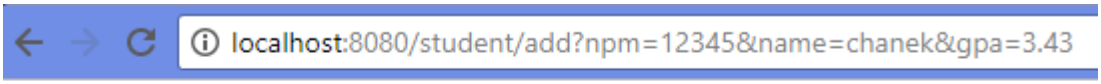
Implementasi method SelectStudent :

```
@Override
public StudentModel selectStudent(String npm) {
    for(int i=0; i<studentList.size(); i++) {
        StudentModel student = studentList.get(i);
        if(student.getNpm().equals(npm)) {
            return student;
        }

    }

    return null;
}
```

Method ini mengiterasikan ArrayList untuk menemukan mahasiswa dengan NPM yang sama dengan parameter yang dimasukan, jika NPM yang dimasukan sama maka akan me*return student* yang sesuai dengan NPM tersebut, sebaliknya jika tidak ada kecocokan maka akan *return* NULL yang berarti bahwa tidak ada *student* tersebut di dalam ArrayList.

Membuat Controller dan Fungsi Add

1. Jalankan program dan buka localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43



Data berhasil ditambahkan

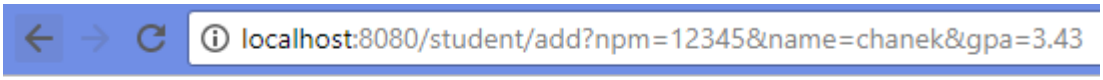
Pertanyaan 1: apakah hasilnya? Jika error, tuliskan penjelasan Anda.
Jawaban: Tidak terjadi *error*, menampilkan “Data berhasil ditambahkan” yang berarti data tersebut berhasil ditambahkan ke dalam ArrayList studentList

2. localhost:8080/student/add?npm=12345&name=chanek

Pertanyaan 2: apakah hasilnya? Jika error, tuliskan penjelasan Anda.
Jawaban: Terjadi Whitelabel Error Page (type=Bad Request, status=400) karena value GPA tidak dituliskan di dalam parameter method get (URL) dan required dalam RequestParam GPA adalah *true*

Method View by NPM

3. Jalankan program dan buka localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43



Data berhasil ditambahkan

lalu buka localhost:8080/student/view?npm=12345

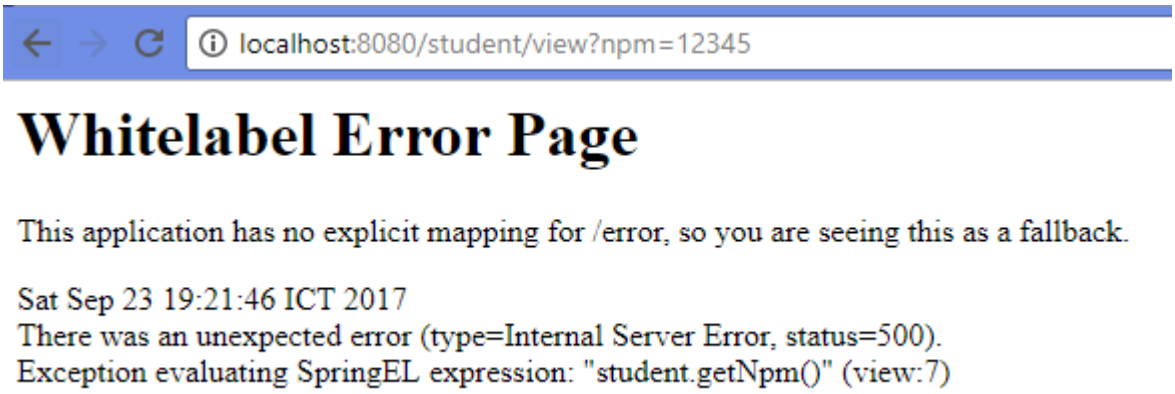
NPM = 12345

Name = chanek

GPA = 3.43

Pertanyaan 3: apakah data Student tersebut muncul? Jika tidak, mengapa?
Jawaban: Ya, data *student* muncul sesuai dengan yang sebelumnya dimasukkan pada parameter: **npm=12345&name=chanek&gpa=3.43** karena sudah terdapat di dalam object *student* dalam ArrayList studentList

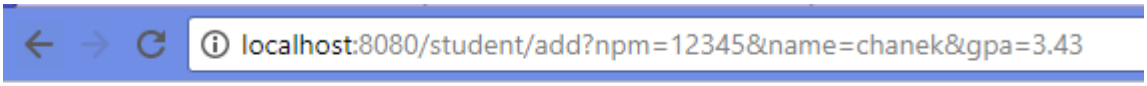
4. Coba matikan program dan jalankan kembali serta buka localhost:8080/student/view?npm=12345



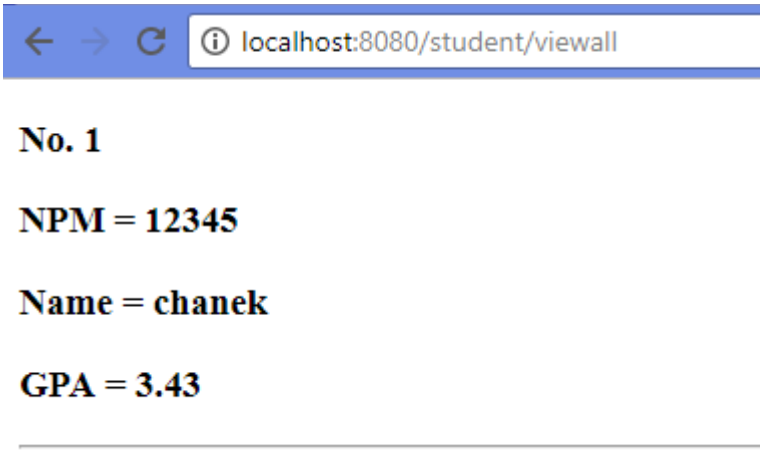
Pertanyaan 4: apakah data Student tersebut muncul? Jika tidak, mengapa?
Jawaban: Tidak muncul**,** karena data ada pada *session* yang ada setiap kali SpringBoot berjalan dan ArrayList akan terinisialisasi kembali sehingga isi Arraylist akan menjadi kosong, apabila SpringBoot di matikan maka *session* akan diperbaharui sepenuhnya dan object *student* akan menjadi *null* sehingga terjadi null *pointer exception* pada view.html yang menyebabkan terjadinya *error*.

Method View All

5. Jalankan program dan buka localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43



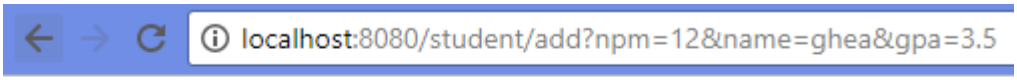
Lalu buka localhost:8080/student/viewall



Pertanyaan 5: apakah data Student tersebut muncul?
Jawaban: Ya data *student* tersebut muncul, ketika /student/viewall diakses akan menampilkan semua *student* di dalam ArrayList studentList, karena baru

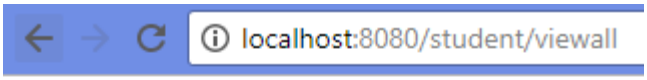
menambahkan pada parameter: **npm=12345&name=chanek&gpa=3.43** maka baru terdapat satu objek student yang terdapat di dalam ArrayList studentList

6. Coba tambahkan data Student lainnya dengan NPM yang berbeda



Data berhasil ditambahkan

Lalu buka localhost:8080/student/viewall



No. 1

NPM = 12345

Name = chanek

GPA = 3.43

No. 2

NPM = 12

Name = ghea

GPA = 3.5

Pertanyaan 6: Apakah semua data Student muncul?
Jawaban: Ya semua data *student* muncul sesuai dengan apa yang telah dimasukan ke dalam parameter dan sudah tersimpan di dalam ArrayList studentList

Latihan

1. Pada StudentController tambahkan sebuah method view Student dengan menggunakan Path Variable.

```
@RequestMapping(value = {"/student/view/", "/student/view/{npm}"})
public String view (@PathVariable Optional<String> npm, Model model) {
    if (npm.isPresent()) {
        StudentModel student = studentService.selectStudent(npm.get());
        if (student == null) {
            return "errorPage2";
        }
        model.addAttribute("student", student);
        return "view";
    }
    return "errorPage";
}
```

Penjelasan:
Pada method ini yang digunakan adalah tipe data Optional<String> untuk menangkap PathVariable NPM pada parameter yang akan dimasukan nantinya, jika NPM terdapat pada parameter, maka npm.isPresent() akan bernilai *true* sehingga proses pencarian *student* akan dilakukan, kemudian akan dilakukan proses pencarian dan akan mendapatkan objek student dari method selectStudent jika NPM ada di dalam ArrayList studentList, namun jika tidak ada maka bernilai null (object student yang dicari tidak ada di dalam ArrayList studentList) dan akan me*return* ke *errorPage2.html* yang berisi NPM tidak ditemukan. Jika tidak null maka *view.html* akan dikeluarkan dan menampilkan NPM, Nama dan GPA dari *student* yang dicari. Sedangkan jika npm.isPresent() bernilai *false* yaitu ketika tidak memasukan NPM pada parameter maka akan langsung me*return* ke *errorPage.html* yang berisi NPM kosong.

Misalnya, Anda ingin memasukkan data seorang Student yang memiliki NPM 14769, untuk melihat data yang baru dimasukkan tersebut dapat mengakses halaman localhost:8080/student/view/14769. Jika nomor NPM tidak diberikan atau tidak ditemukan kembalikan halaman error yang berisi informasi bahwa nomor NPM kosong atau tidak ditemukan.

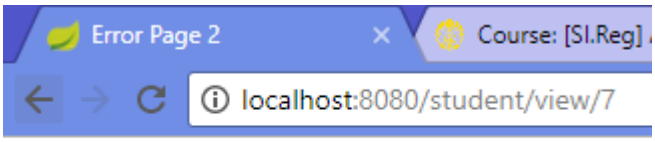
Halaman yang akan ditampilkan jika NPM kosong / tidak diberikan pada paramater

```
<!DOCTYPE html>
<html>
    <head>
        <title>Error Page</title>
    </head>
    <body>
        <h2>NPM kosong</h2>
    </body>
</html>
```

Halaman yang akan ditampilkan jika NPM yang dimasukan tidak ada di dalam ArrayList studentList

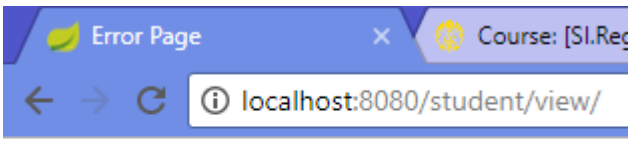
```
<!DOCTYPE html>
<html>
    <head>
        <title>Error Page 2</title>
    </head>
    <body>
        <h2>NPM tidak ditemukan</h2>
    </body>
</html>
```

Output jika NPM yang dimasukan tidak ada di dalam ArrayList studentList:



NPM tidak ditemukan

Output jika NPM yang dimasukan kosong / tidak diberikan pada paramater



NPM kosong

2. Tambahkan fitur untuk melakukan delete Student berdasarkan NPM.

```
@RequestMapping(value = {"/student/delete", "/student/delete/{npm}"})
public String delete (@PathVariable Optional<String> npm, Model model) {
    if (npm.isPresent()) {
        StudentModel student = studentService.selectStudent(npm.get());
        if(student == null) {
            return "errorPage2";
        }
        studentService.deleteStudent(npm.get());
        return "delete";

    }
    return "errorPage";
}
```

Penjelasan:
Pada method ini yang digunakan adalah tipe data Optional<String> untuk menangkap PathVariable NPM pada parameter yang akan dimasukan nantinya, jika NPM terdapat pada parameter, maka npm.isPresent() akan bernilai *true* sehingga proses *delete student* akan dilakukan jika NPM ada di dalam ArrayList studentList kemudian akan

menampilkan *delete.html,* namun jika tidak ada maka bernilai null (object student yang dicari tidak ada di dalam ArrayList studentList) dan akan me*return* ke *errorPage2.html* yang berisi NPM tidak ditemukan. Sedangkan jika npm.isPresent() bernilai *false* yaitu ketika tidak memasukan NPM pada parameter maka akan langsung me*return* ke *errorPage.html* yang berisi NPM kosong.

Implementasi method deleteStudent:

```
@Override
public void deleteStudent(String npm) {
    for (int i = 0; i<studentList.size(); i++) {
        StudentModel student = studentList.get(i);
        if(student.getNpm().equals(npm)) {
            studentList.remove(i);
        }

    }

}
```

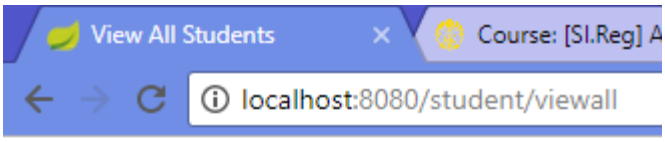
Penjelasan:
Pada method ini menerima parameter String npm, NPM tersebut akan dicari dalam ArrayList studentList dengan cara melakukan looping pada , jika terdapat *student* yang memiliki npm sesuai dengan parameter yang dimasukan maka data tersebut akan dihapus dari ArrayList studentList.

Misalnya, setelah melakukan add Student pada soal nomor 1, cobalah untuk melakukan delete data tersebut dengan mengakses halaman localhost:8080/student/delete/14769. Tampilkan sebuah halaman yang memberikan informasi bahwa data tersebut telah berhasil dihapus. Jika nomor NPM tidak diberikan atau tidak ditemukan kembalikan halaman error yang berisi informasi bahwa nomor NPM kosong atau tidak ditemukan dan proses delete dibatalkan.

Halaman untuk menunjukkan bahwa proses *delete* berhasil:

```
1 <!DOCTYPE html>
2     <html>
3     <head>
4         <title>Notification</title>
5     </head>
6     <body>
7         <h2>Data berhasil dihapus :(</h2>
8     </body>
9 </html>
```

Data ditambahkan ke dalam ArrayList studentList:



No. 1

NPM = 12

Name = ghea

GPA = 3.45

No. 2

NPM = 9

Name = danu

GPA = 3.84

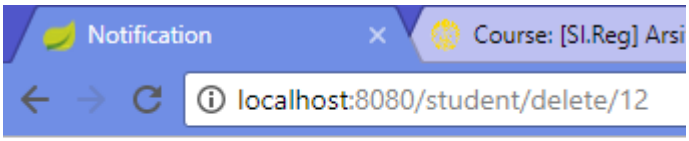
No. 3

NPM = 12345

Name = chanek

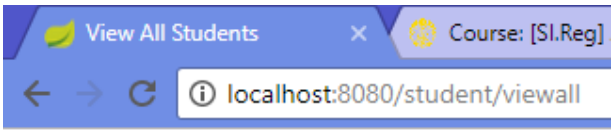
GPA = 3.43

Lalu, saya mendelete *student* Ghea dengan NPM = 12



Data berhasil dihapus :(

Sehingga objek yang ada di dalam ArrayList studentList hanya tersisa 2:



No. 1

NPM = 9

Name = danu

GPA = 3.84

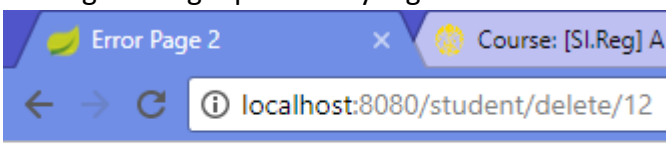
No. 2

NPM = 12345

Name = chanek

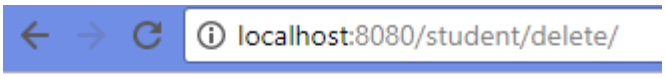
GPA = 3.43

Jika ingin menghapus NPM yang tidak ada di dalam ArrayList studentList



NPM tidak ditemukan

Jika ingin menghapus NPM yang kosong / tidak diberikan pada paramater



NPM kosong

Ringkasan:

Pada tutorial 3 ini, saya belajar lebih dalam mengenai penggunaan class Model dalam MVC serta service pada SpringBoot. Awalnya saya berpikir bahwa logic diletakkan pada setiap method yang ada di *controller*, namun ternyata logic-logic yang banyak itu lebih baik diletakkan di dalam *service* agar lebih rapi dan tidak menumpuk. Saya juga belajar mengenai bagaimana penggunaan *interface* yang baik dalam menggunakan MVC serta mempraktekan penggunaan java util seperti List. Melakukan looping pada view, engirimkan object melalui model, cara melakukan looping menggunakan thymeleaf, serta cara kerja penyimpanan data pada SpringBoot yang didapatkan melalui RequestParam dan PathVariable juga saya pelajari dalam tutorial kali ini.