

WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

A. RINGKASAN MATERI YANG DIPELAJARI

Pada tutorial 3, saya mempelajari lebih lanjut mengenai implementasi *model-view-controller* (MVC) dengan menggunakan Spring Boot. Bagian utama yang dipelajari dalam tutorial ini adalah mengenai *Model* dan *Service*.

Model yang dibuat pada tutorial kali ini adalah *class* StudentModel.java yang berisi variable *name* dengan tipe data *String*, *npm* dengan tipe data *String*, dan *gpa* dengan tipe data *double*.

Service yang dibuat pada tutorial ini adalah *class interface* StudentService.java yang diimplementasikan di *class* InMemoryStudentService.java. *Class interface* ini mendefinisikan *method-method* yang digunakan untuk memanipulasi *class* StudentModel. *Method* yang ada digunakan untuk fungsi-fungsi tertentu seperti melihat data *Student* dengan melakukan *input* npm, melihat data seluruh *Student* yang ada, serta melakukan penambahan *Student*.

Selain belajar mengenai *Model* dan *Service*, saya juga mengulas / meninjau kembali mengenai *List* dan *ArrayList* di tutorial 3 ini. Mempelajari kembali mengenai bagaimana menambahkan data ke dalam *List*, melihat data yang ada di *List*, serta menghapus data yang ada pada *List*.

Dalam pengerjaan tutorial 3 ini pula, saya mempelajari kembali mengenai *Path Variable* dan *Request Parameter* yang diimplementasikan pada *Controller Class*.

B. JAWABAN TUTORIAL

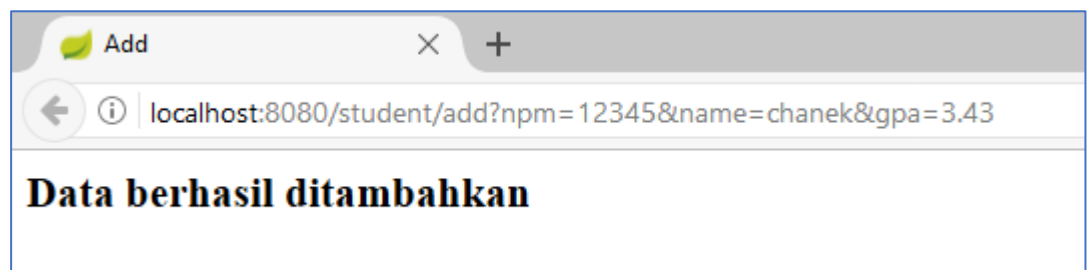
Membuat Controller dan Fungsi Add

Jalankan *program* dan buka :

- <localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43>

Pertanyaan 1 : apakah hasilnya? Jika *error*, tuliskan penjelasan Anda.

⇒ Tidak ada *error*, hasil yang ditampilkan adalah seperti *screenshot* dibawah ini.



Dari hasil tersebut menunjukkan bahwa data berhasil ditambahkan ke dalam *List of StudentModel*.

WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

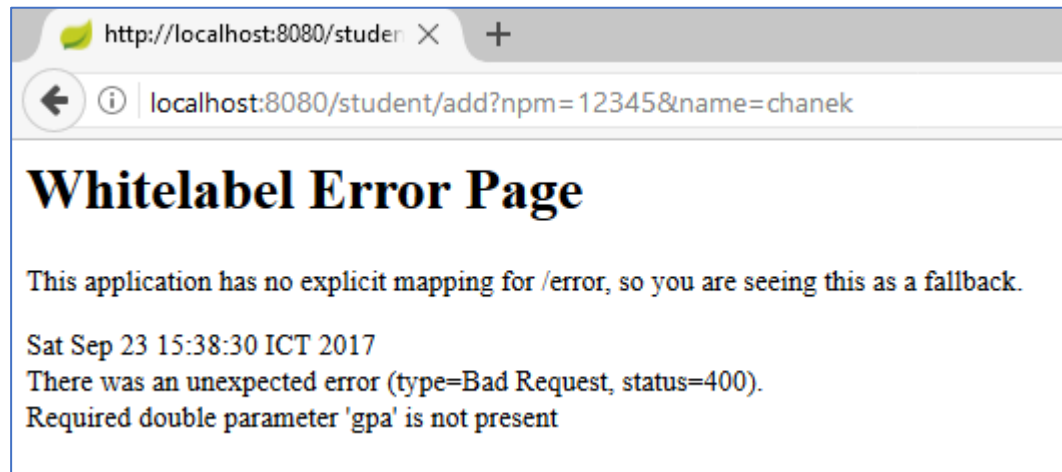
NPM : 1506689622

Kelas : ADPAP – B

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

Pertanyaan 2 : apakah hasilnya? Jika *error*, tuliskan penjelasan Anda.

⇒ Ada *error*, hasil yang ditampilkan adalah seperti *screenshot* dibawah ini



Hasil yang menunjukkan adanya *error* tersebut dikarenakan *value* untuk *parameter* gpa tidak ada. Padahal pada kode program *method* add di *class* StudentController *require* untuk parameter gpa di-set true.

```
@RequestMapping("/student/add")
public String add(@RequestParam(value = "npm", required = true) String npm,
                 @RequestParam(value = "name", required = true) String name,
                 @RequestParam(value = "gpa", required = true) double gpa) {
    StudentModel student = new StudentModel(npm, name, gpa);
    studentService.addStudent(student);
    return "add";
}
```

WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

Method View by NPM

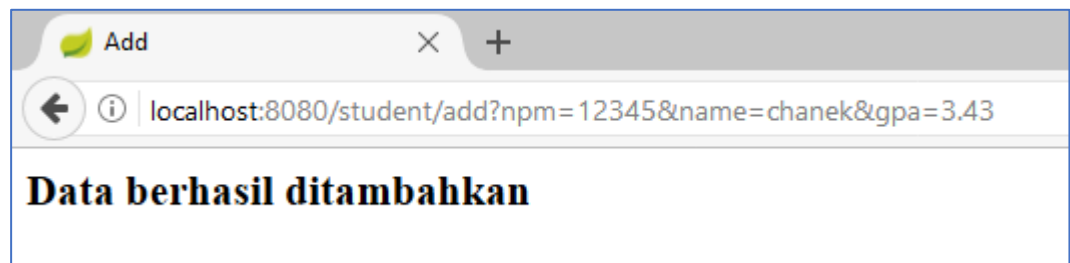
Jalankan *program* dan buka

- <localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43> lalu buka <localhost:8080/student/view?npm=12345>

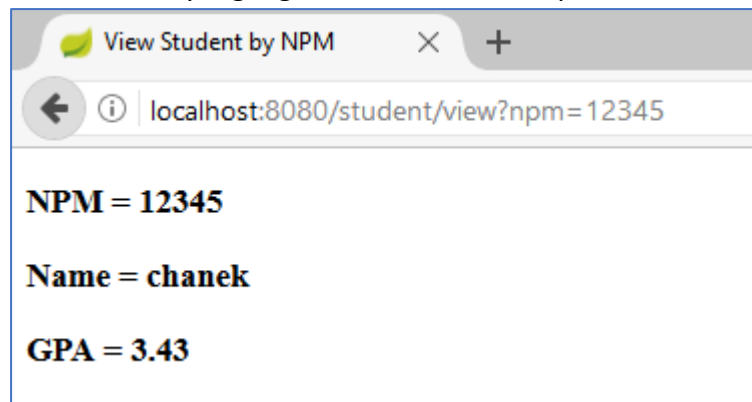
Pertanyaan 3 : apakah data Student tersebut muncul? Jika tidak, mengapa?

⇒ Ya data Student tersebut muncul,

Melakukan penambahan *Student* terlebih dahulu, hasilnya seperti berikut yaitu berhasil dilakukan penambahan.



Data *Student* yang ingin di-view muncul seperti hasil *screenshot* dibawah ini.



Coba matikan program dan jalankan kembali serta buka :

WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

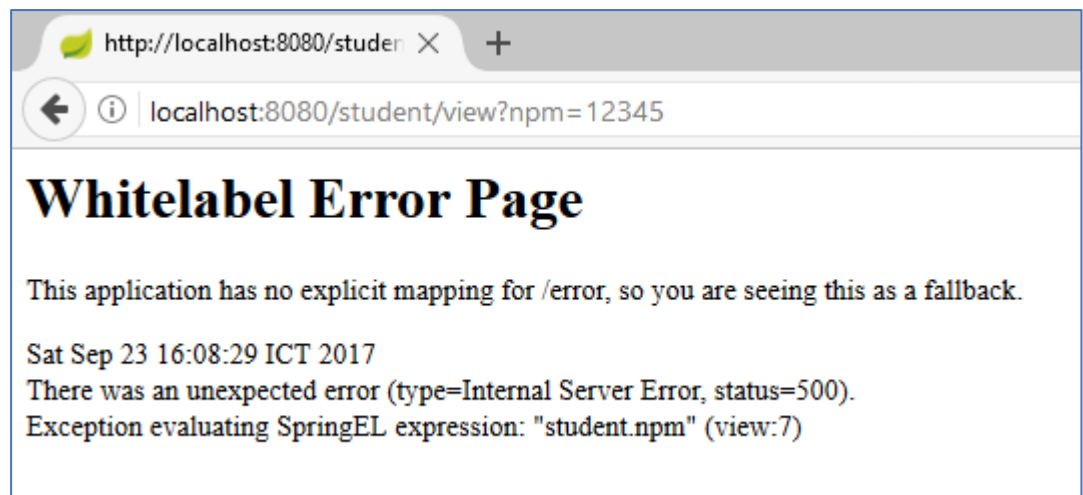
NPM : 1506689622

Kelas : ADPAP – B

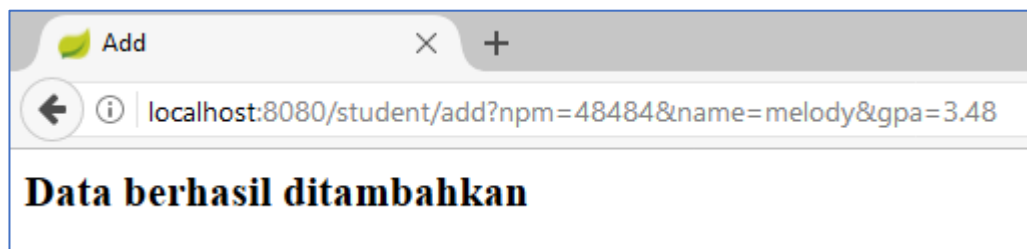
- localhost:8080/student/view?npm=12345

Pertanyaan 4 : apakah data Student tersebut muncul? Jika tidak, mengapa?

⇒ Data Student tidak muncul. Data yang ingin dilihat tidak muncul karena setelah program dimatikan, data yang telah ditambahkan ke *List* sebelumnya juga ikut terhapus.



Mencoba menambahkan data *Student* lainnya dengan NPM yang berbeda.



WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

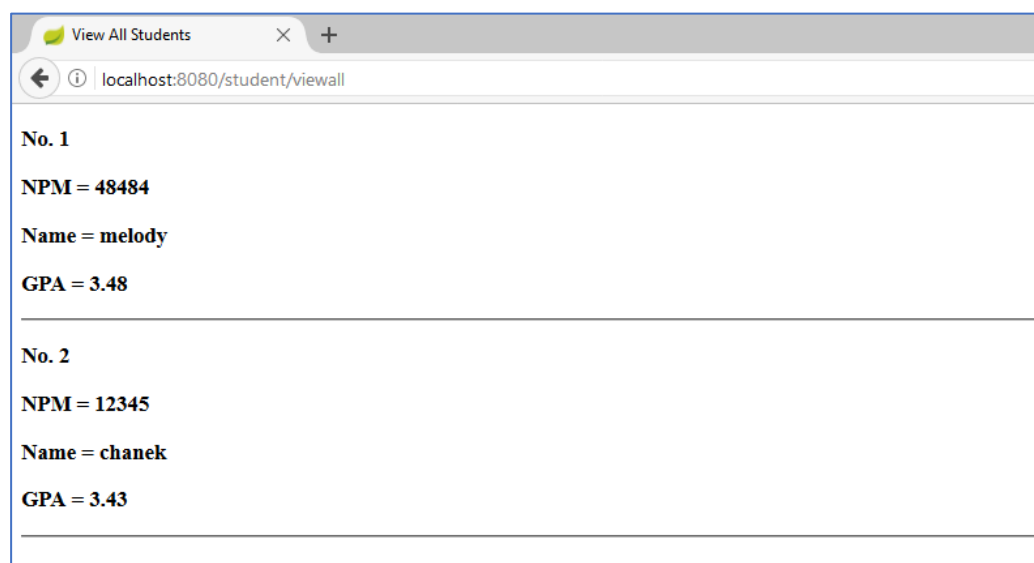
Method View All

Jalankan program dan buka

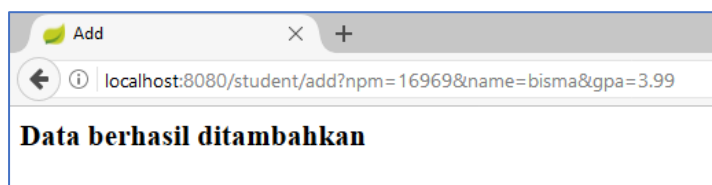
- <localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43> lalu buka <localhost:8080/student/viewall>

Pertanyaan 5 : apakah data Student tersebut muncul?

⇒ Ya, data *Student* tersebut muncul. Muncul pula data *Student* yang sebelumnya ditambahkan. Hasilnya dapat dilihat seperti *screenshot* dibawah ini :



Coba tambahkan data Student lainnya dengan NPM yang berbeda,



WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

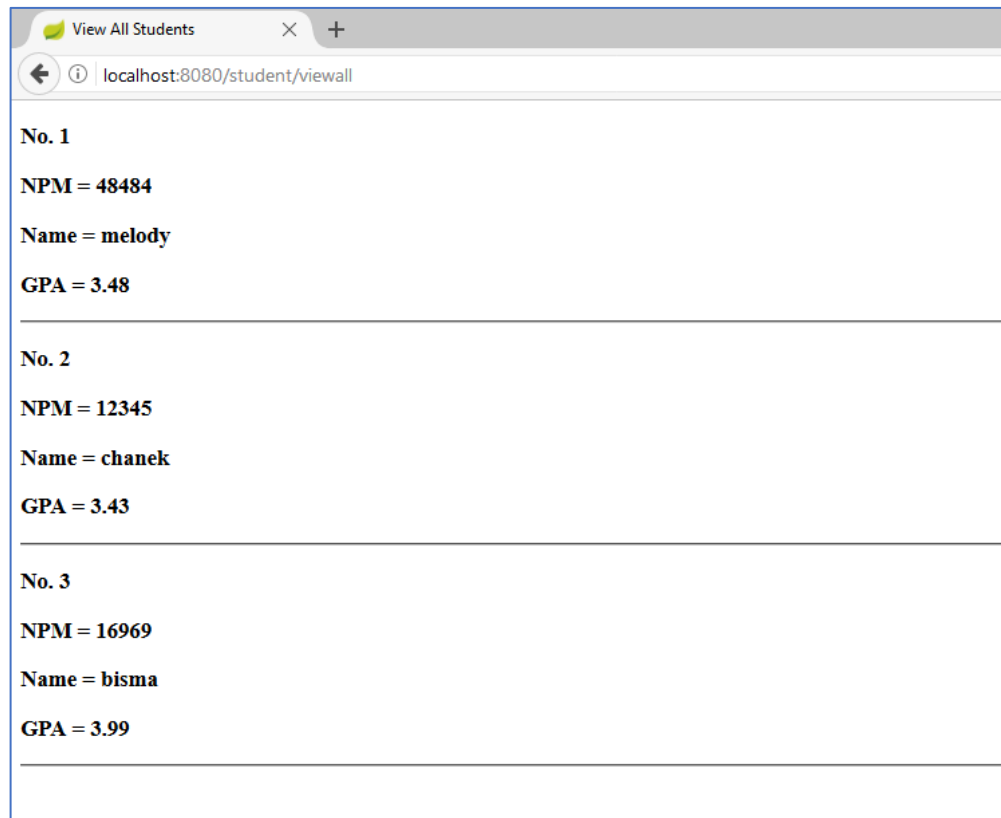
Kelas : ADPAP – B

lalu buka

- <localhost:8080/student/viewall>

Pertanyaan 6 : Apakah semua data Student muncul?

⇒ Ya, semua data *Student* muncul



The screenshot shows a web browser window with the title 'View All Students'. The address bar displays 'localhost:8080/student/viewall'. The page content lists three students, each with their ID, name, and GPA, separated by horizontal lines.

No. 1	NPM = 48484	Name = melody	GPA = 3.48
No. 2	NPM = 12345	Name = chaneke	GPA = 3.43
No. 3	NPM = 16969	Name = bisma	GPA = 3.99

WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

C. IMPLEMENTASI METHOD selectStudent

Berikut ini adalah kode program dari *method* selectStudent yang saya implementasikan pada kelas InMemoryStudentService.java.

```
@Override
public StudentModel selectStudent(String npm) {
    //Implementation with enhanced loop
    for(StudentModel student : studentList) {
        if(student.getNpm().equals(npm)) {
            return student;
        }
    }
    return null;
}
```

Menggunakan *enhanced loop* dalam menelusuri *object* StudentModel pada *List* studentList untuk mempermudah dalam melakukan inisiasi variable indeks maupun *update* terhadap variable indeks.

Cara kerjanya adalah dari *List* yang berisi kumpulan *object* StudentModel yang dinamai dengan variabel studentList saya ingin mencari *object* StudentModel yang dinamai dengan variabel student. Penelusuran iterasi *for* dilakukan hingga menemukan npm dari *object* StudentModel pada *List* yang memiliki npm sama dengan input yang diberikan pada parameter method yaitu *String npm*. Karena pada *class* StudentModel kita mempunyai *method* getNpm() yang mengembalikan npm dari *student*, kita tinggal mencocokkannya dengan npm yang menjadi parameter *method* selectStudent.

Jika npm yang didapat dari iterasi yang dilakukan pada *object* StudentModel didalam *List* sama dengan input yang diberikan maka akan mengembalikan *object* StudentModel yang dinyatakan dalam *for* dengan variable *student*. Jika tidak ada yang sama hingga seluruh *object* dalam *List* diperiksa maka akan keluar dari *loop for* dan mengembalikan *null* (*return null*).

WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

D. PENJELASAN LATIHAN

1. **View menggunakan Path Variable**

Berikut ini adalah *screenshot* dari kode program untuk *method* *viewPath* pada *StudentController.java* yang menggunakan *Path Variable* untuk melakukan *view Student*.

```
/* Latihan Soal No.1 View Student dengan Path Variable
 * NOTE : pada salah satu value mapping dibuat "/student/view/"
 * agar tidak ambigu dengan mapping pada method view yaitu
 * "student/view" akses di browser untuk tanpa input npm menjadi
 * "localhost:8080/student/view/" */

@RequestMapping(value = {"/student/view/", "student/view/{npm}"})
public String viewPath(@PathVariable Optional<String> npm, Model model) {
    if(npm.isPresent()) {
        StudentModel student = studentService.selectStudent(npm.get());
        if(student == null) {
            model.addAttribute("npm", npm.get());
            model.addAttribute("statement", " tidak ditemukan.");
        }else {
            model.addAttribute("student", student);
            return "view";
        }
    }else {
        model.addAttribute("npm", "");
        model.addAttribute("statement", " kosong.");
    }
    return "errorNPMview";
}
```

Catatan : Untuk mencegah terjadinya *mapping* yang ambigu antara *method view* dan *viewPath* dalam melakukan perintah tanpa adanya *input parameter npm*, solusinya untuk *method viewPath* dibuat menjadi *"/student/view/"*, sehingga pada *browser* buka **localhost:8080/student/view/** untuk input tanpa parameter *npm*.

Halaman *view.html* untuk *NPM* yang berhasil ditemukan

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>View Student by NPM</title>
  </head>
  <body>
    <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
    <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
    <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
  </body>
</html>
```


WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

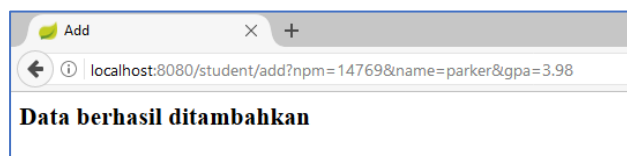
Halaman `errorNPMview.html` untuk npm yang tidak berhasil ditemukan atau tanpa input npm.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>Error View Student by NPM</title>
  </head>
  <body>
    <h3 th:text="'NPM ' + ${npm} + ${statement}'>Student NPM Invalid</h3>
  </body>
</html>
```

Cara kerja jalannya kode program adalah sebagai berikut :

- Program akan memeriksa apakah parameter npm dihadirkan atau dimasukan di *request mapping*.
- Jika ada parameter npm, maka akan dilakukan pengecekan apakah ada atau tidak npm yang di-input dalam kumpulan data *object StudentModel*. Jika ada maka model akan menambahkan *attribute student* tersebut dan mengembalikan *view* (*return "view"*) yang merujuk ke *view.html*. Jika tidak ada / tidak ditemukan npm yang ingin dilihat (*student == null*), maka akan ditambahkan *attribute npm* yang dicari dan *attribute statement* yang menyatakan "tidak ditemukan" kemudian keluar dari kondisi *if else statement* dan mengembalikan *errorNPMview* (*return "errorNPMview"*) merujuk pada *errorNPMview.html* yang berfungsi untuk menampilkan pesan *error* dari *view*.
- Jika tidak ada parameter (***localhost:8080/student/view/***), maka model akan menambahkan *String* kosong untuk *attribute npm* dan *attribute statement* yang menyatakan "kosong". Kemudian keluar dari *if-else* dan mengembalikan *String "errorNPMview"* (*return "errorNPMview"*).

Misalnya kita tambahkan *Student* yang memiliki NPM 14769



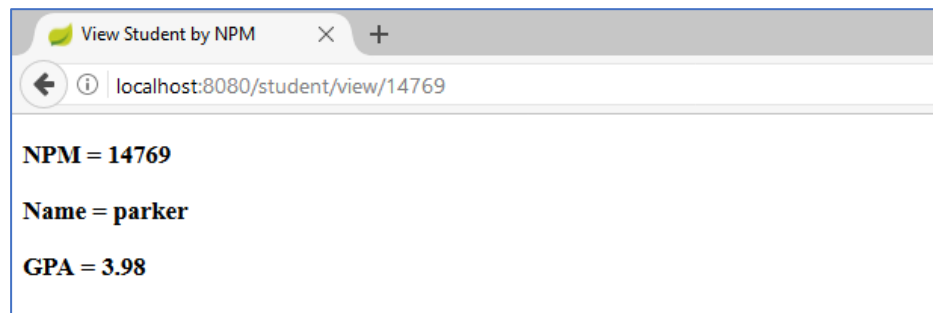
Kemudian *view* NPM tersebut : ***localhost:8080/student/view/14769***

WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

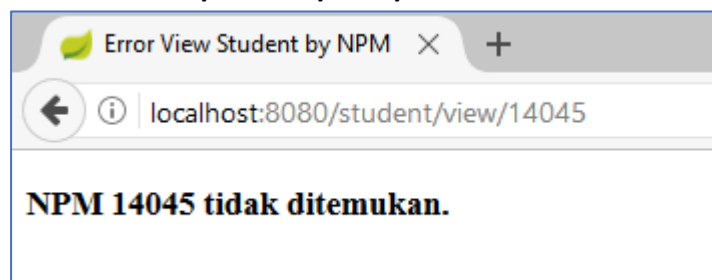
NPM : 1506689622

Kelas : ADPAP – B



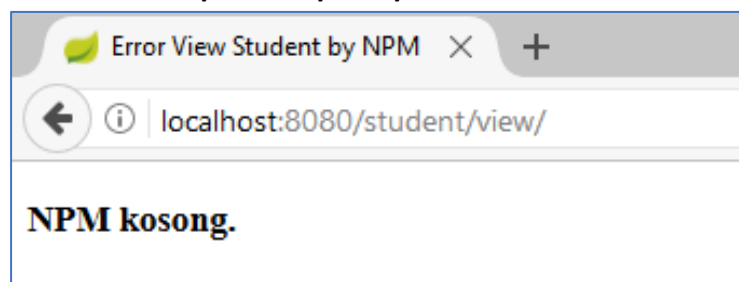
Untuk kasus NPM yang tidak terdaftar / tidak ditemukan, misalnya :

localhost:8080/student/view/14045



Untuk kasus parameter NPM yang tidak diinput, misalnya:

localhost:8080/student/view/



WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

2. Penjelasan Fitur untuk melakukan *delete by NPM*

Melakukan proses *delete* atau menghapus data *Student* kita menggunakan perintah yang ada pada *List* untuk menghapus data dari *List of object*. Berikut ini adalah kode program dari *method deletePath* pada *class StudentController.java*.

```
/* Latihan Soal No. 2 Delete Student dengan Path Variable */
@RequestMapping(value = {"/student/delete", "student/delete/{npm}"})
public String deletePath(@PathVariable Optional<String> npm, Model model) {
    if(npm.isPresent()) {
        StudentModel student = studentService.selectStudent(npm.get());
        if(student == null) {
            model.addAttribute("npm", npm.get());
            model.addAttribute("statement", " tidak ditemukan.");
        }
        else {
            int i = 0;
            while(i < studentService.selectAllStudents().size()) {
                if(studentService.selectAllStudents().get(i).getNpm().equals(npm.get())) {
                    model.addAttribute("student", student);
                    studentService.selectAllStudents().remove(i);
                }
                i++;
            }
            return "delete";
        }
    }
    else {
        model.addAttribute("npm", "");
        model.addAttribute("statement", " kosong.");
    }
    return "errorDelete";
}
```

Halaman delete.html untuk data yang berhasil di-*delete*

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Delete Student by NPM</title>
    </head>
    <body>
        <h2>Delete berhasil terhadap data student :</h2>
        <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
        <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
        <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
    </body>
</html>
```

WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

Halaman errorDelete.html untuk data yang tidak berhasil di-delete :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Error Delete Student by NPM</title>
</head>
<body>
<h2>Delete Dibatalkan</h2>
<h3 th:text="'NPM ' + ${npm} + ${statement}'">Student NPM Invalid</h3>
</body>
</html>
```

Cara kerja jalannya kode program pada *method* deletePath, sebagai berikut :

- Program akan memeriksa apakah parameter npm dihadirkan atau dimasukan di *request mapping*.
- Jika dimasukkan parameter npm, maka akan dilakukan pengecekan apakah ada atau tidak npm yang di-*input* tersebut di dalam kumpulan data *object StudentModel*.

Jika ada maka akan dicari data yang ingin dihapus berdasarkan npm pada *List of object StudentModel* dengan memanfaatkan *method* selectAllStudents() yang mengembalikan *list of object* dari StudentModel. Pencarian terlebih dahulu dengan mendapatkan ukuran *List of object* (banyaknya *students* dalam *List*) yaitu `studentService.selectAllStudents().size()`. Menggunakan variabel *int* *i* sebagai indeks pencarian dari *list of object* dilakukan *looping* selama variabel *i* kurang dari ukuran *ArrayList of object StudentModel* maka akan dilakukan pencarian terhadap npm yang ada pada *List*. Jika pada indeks tertentu yang diakses NPM nya sama dengan NPM yang di-*input*, maka model akan menambahkan *attribute* student tersebut (`model.addAttribute("student", student)`), menghapus *object* StudentModel pada indeks tersebut (`studentService.selectAllStudents().remove(i)`), kemudian keluar dari *loop* dan mengembalikan delete (`return "delete"`) yang merujuk pada delete.html.

Jika tidak ada / tidak ditemukan npm yang ingin dilihat (`student == null`), maka akan ditambahkan *attribute* npm yang dicari dan *attribute statement* yang menyatakan "tidak ditemukan" kemudian keluar dari kondisi *if else statement* dan mengembalikan *errorDelete* (`return "errorDelete"`) merujuk pada errorDelete.html yang berfungsi untuk menampilkan pesan *error* dari delete.

WRITE-UP TUTORIAL 3 ADPAP

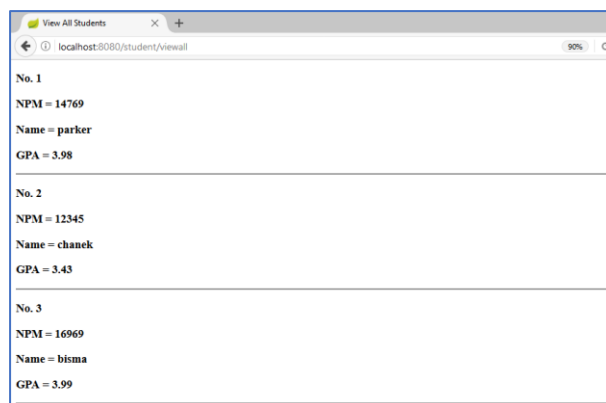
Nama : Yosua Bisma Putrapratama

NPM : 1506689622

Kelas : ADPAP – B

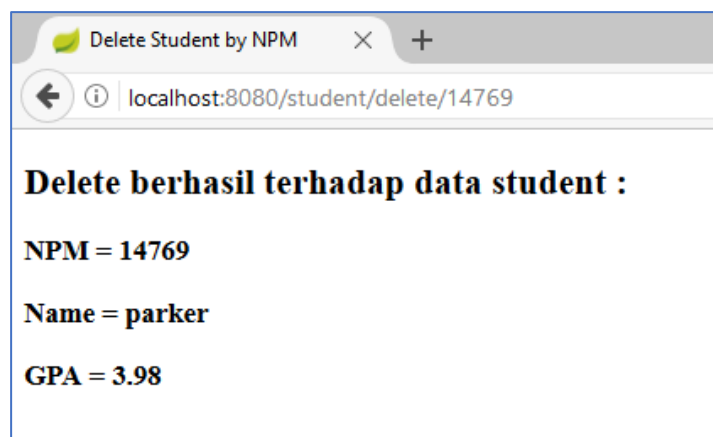
- Jika tidak ada parameter (**localhost:8080/student/delete**), maka model akan menambahkan *String* kosong ("") untuk *attribute* npm dan *attribute statement* yang menyatakan “kosong”. Kemudian keluar dari *if-else* dan mengembalikan *String* “errorDelete” (return “errorDelete”).

Berikut ini misalnya *List of Object* StudentModel (data *student* yang sudah ditambahkan)



No. 1
NPM = 14769
Name = parker
GPA = 3.98
No. 2
NPM = 12345
Name = chanek
GPA = 3.43
No. 3
NPM = 16969
Name = bisma
GPA = 3.99

Melakukan *delete* terhadap NPM yang ada dalam *list of object* StudentModel : **localhost:8080/student/delete/14769**



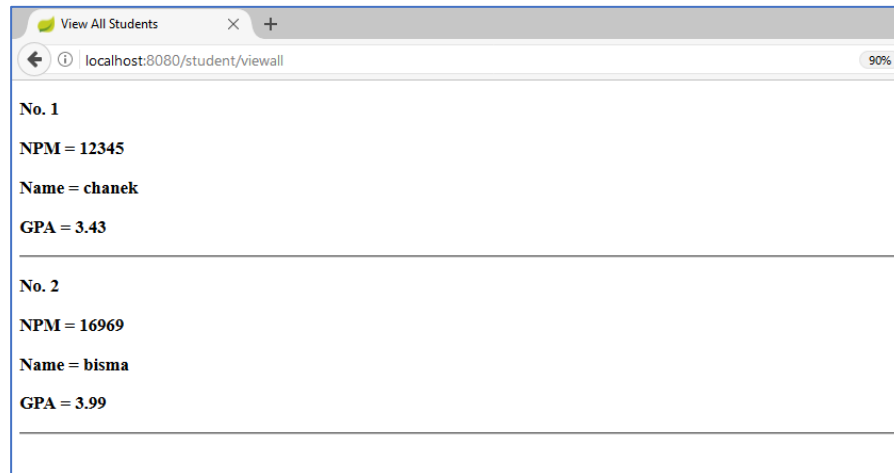
WRITE-UP TUTORIAL 3 ADPAP

Nama : Yosua Bisma Putrapratama

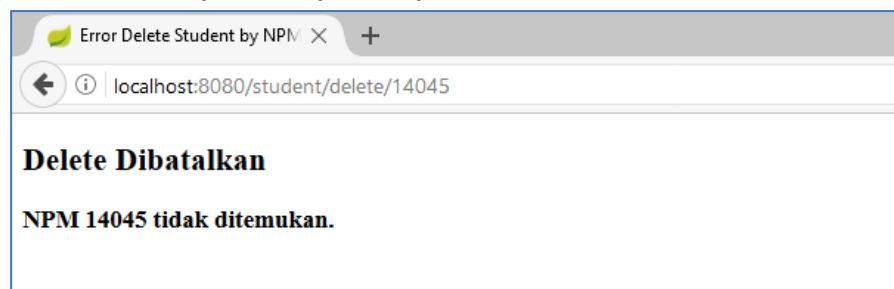
NPM : 1506689622

Kelas : ADPAP – B

Maka data *student* dengan NPM 14769 sudah dihapus dari kumpulan *object* StudentModel, berikut jika kita *viewall* lagi :



Untuk kasus NPM yang tidak terdaftar / tidak ditemukan, misalnya : **localhost:8080/student/delete/14045**



Untuk kasus parameter NPM yang tidak diinput, misalnya: **localhost:8080/student/delete**

