

Tutorial 3 : Menggunakan Model dan Service pada Project Spring Boot

Membuat Class Model

1. Method selectStudent setelah berhasil di implementasikan:

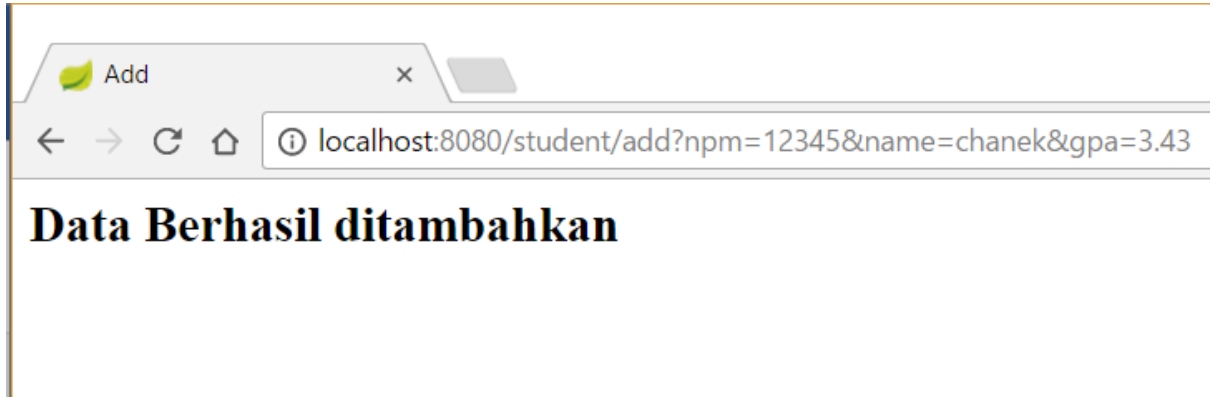
```
@Override
public StudentModel selectStudent(String npm){
    for(int i = 0; i < studentList.size(); i++){
        if(studentList.get(i).getNpm().equalsIgnoreCase(npm)){
            return studentList.get(i);
        }
    }
    //Implement
    return null;
}
```

Pada metode berikut, mengiterasikan arraylist untuk menemukan mahasiswa dengan NPM yang sama dengan yang dimasukan dalam parameter, jika ada yang sama kemudian *return* dari mahasiswa yang didapatkan, jika tidak maka NULL menandakan bahwa tidak ada mahasiswa tersebut.

Membuat Service

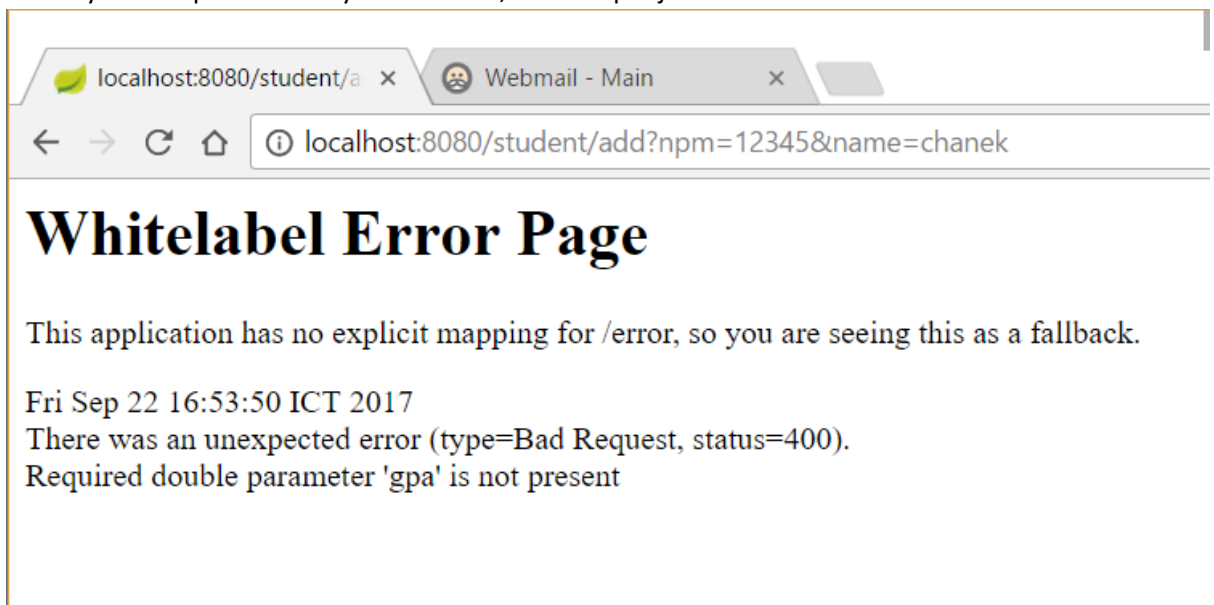
1. localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43

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



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

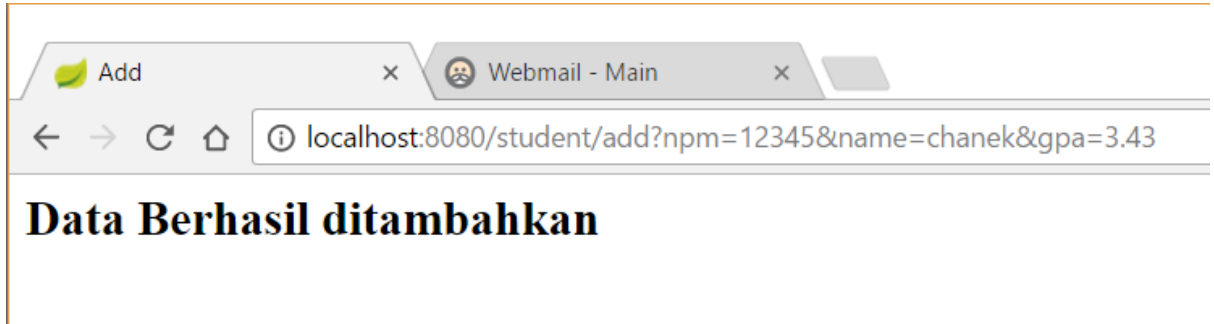
Pertanyaan 2: apakah hasilnya? Jika error, tuliskan penjelasan Anda.



Terjadi Whitelabel Error Page (type=Bad Request, status=400), dibutuhkan parameter 'gpa' untuk RequestParam GET

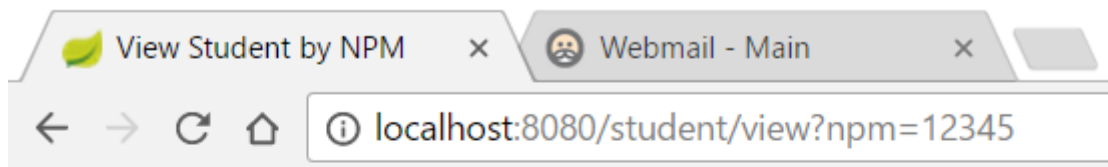
Method View by NPM

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



lalu buka

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



NPM = 12345

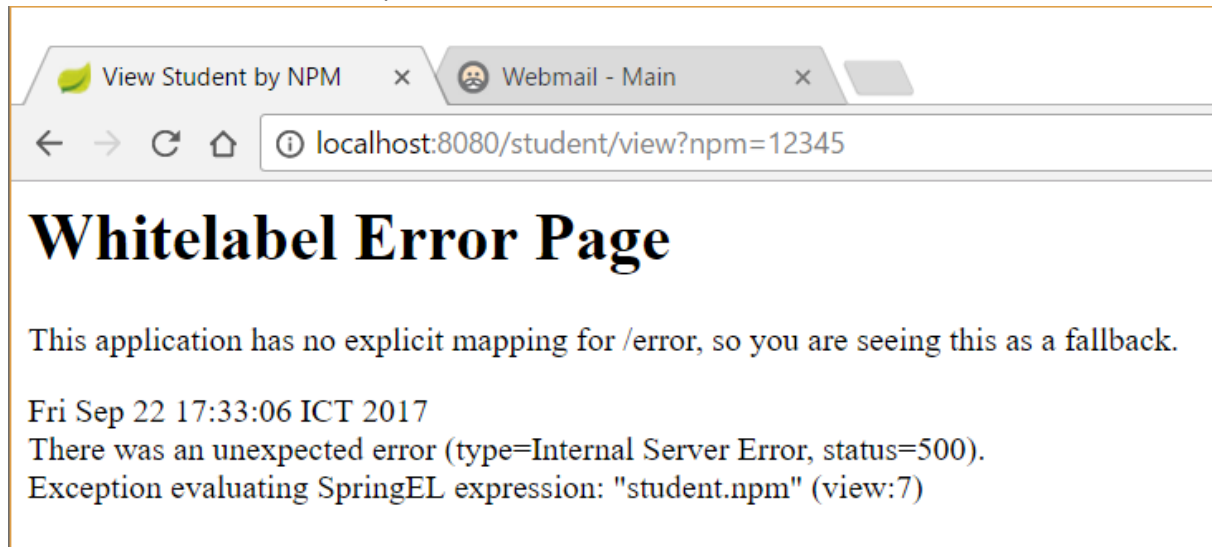
Name = chanek

GPA = 3.43

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

Muncul data student yang dimaksud, sesuai dengan yang sebelumnya dimasukkan pada parameter: **npm=12345&name=chanek&gpa=3.43**

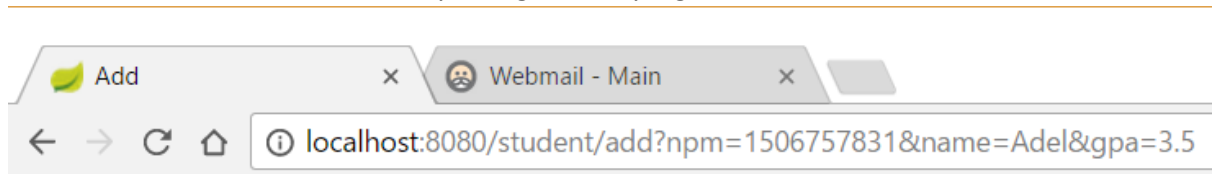
2. Coba matikan program dan jalankan kembali serta buka localhost:8080/student/view?npm=12345



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

Tidak muncul, karena data ada pada session yang ada setiap kali springboot berjalan, apabila springboot di matikan maka session akan diperbaharui sepenuhnya.

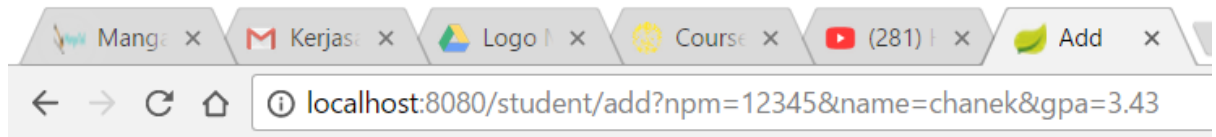
3. Coba tambahkan data Student lainnya dengan NPM yang berbeda.



Data Berhasil ditambahkan

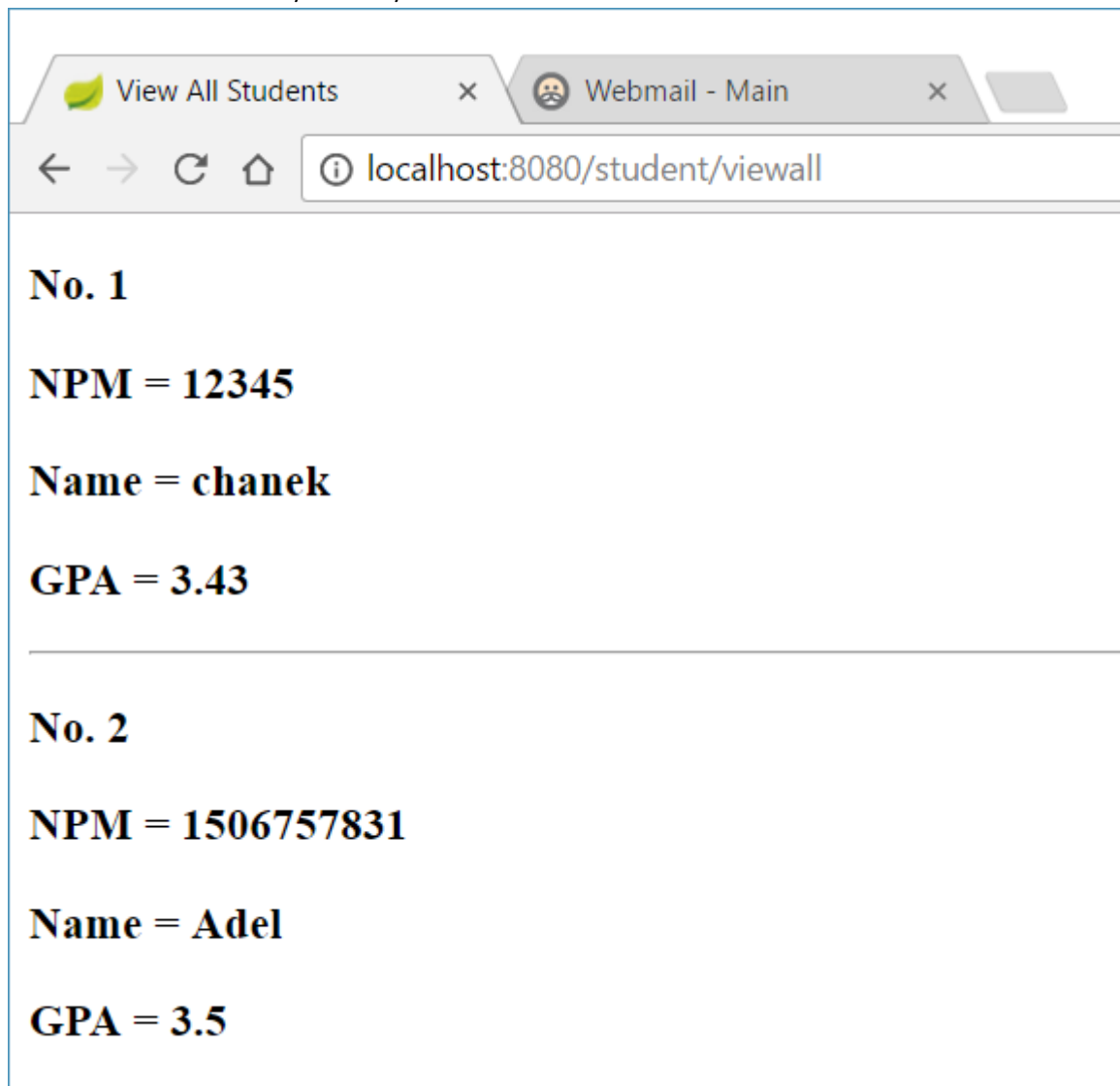
Method View All

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

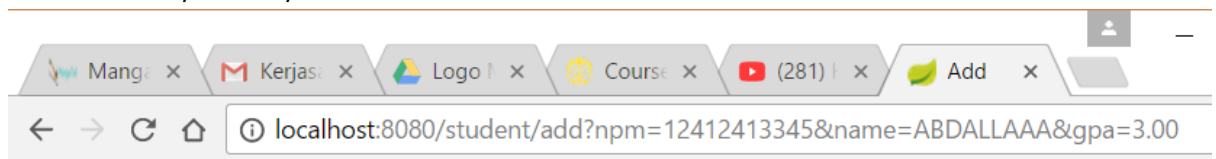


Data Berhasil ditambahkan

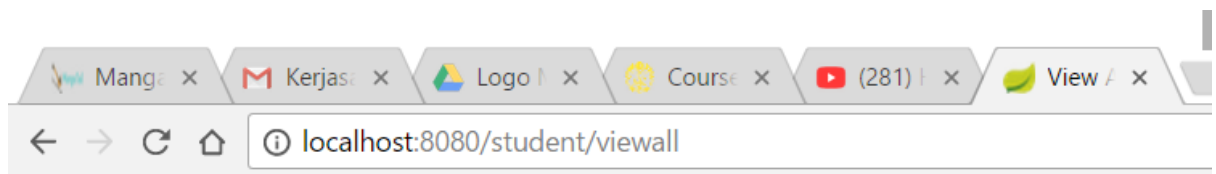
lalu buka `localhost:8080/student/viewall`



2. Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka localhost:8080/student/viewall



Data Berhasil ditambahkan



No. 1

NPM = 12345

Name = chanek

GPA = 3.43

No. 2

NPM = 1506757831

Name = Adel

GPA = 3.5

No. 3

NPM = 12412413345

Name = ABDALLAAA

GPA = 3.0

Pertanyaan 6: Apakah semua data Student muncul?

Muncul

Latihan

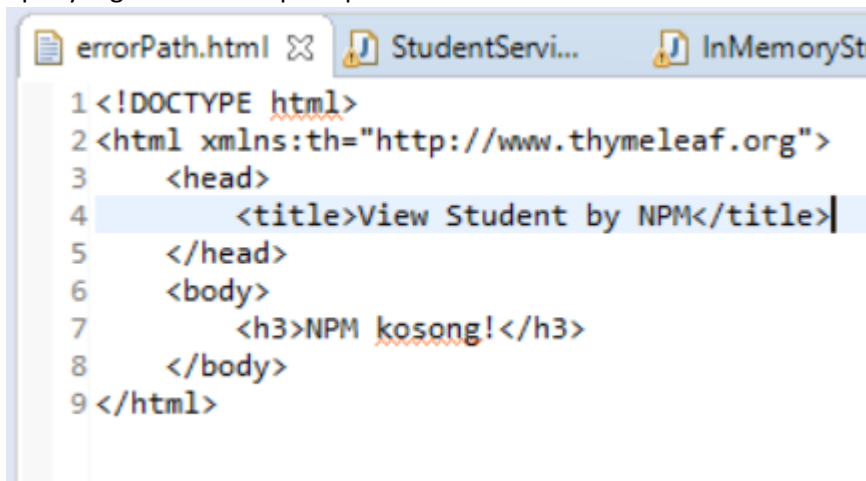
1. Pada StudentController tambahkan sebuah method view Student dengan menggunakan Path Variable.

Langkah:

Pertama, membuat method pada controller, dengan request mapping “/student/view/” dan “/student/view/{npm}”

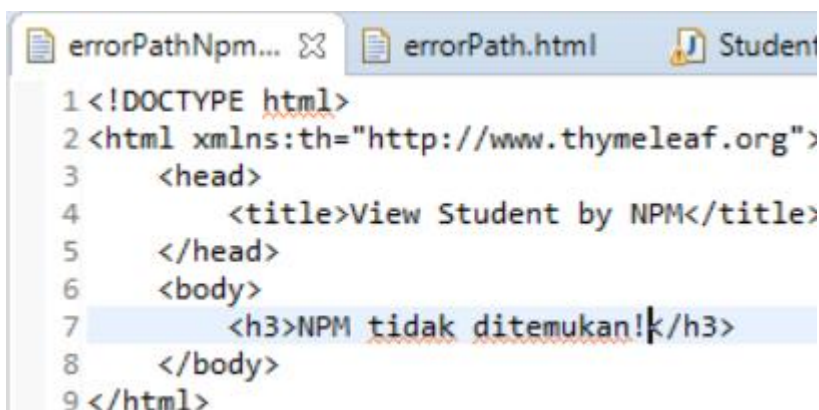
```
@RequestMapping(value={"/student/view", "/student/view/{npm}"})  
public String studentNpmPath (@PathVariable Optional<String> npm, Model model)  
{
```

Kemudian membuat html untuk error page, dimana page error untuk “Npm kosong” apabila tidak ada parameter npm, dan “Npm tidak ditemukan” apabila tidak ada mahasiswa dengan npm yang dimasukkan pada parameter



```
1 <!DOCTYPE html>  
2 <html xmlns:th="http://www.thymeleaf.org">  
3   <head>  
4     <title>View Student by NPM</title>  
5   </head>  
6   <body>  
7     <h3>NPM kosong!</h3>  
8   </body>  
9 </html>
```

ErrorPath akan keluar apabila npm tidak diisi pada parameter



```
1 <!DOCTYPE html>  
2 <html xmlns:th="http://www.thymeleaf.org">  
3   <head>  
4     <title>View Student by NPM</title>  
5   </head>  
6   <body>  
7     <h3>NPM tidak ditemukan!</h3>  
8   </body>  
9 </html>
```

ErrorPathNpm akan keluar apabila npm tidak ditemukan pada arrayList

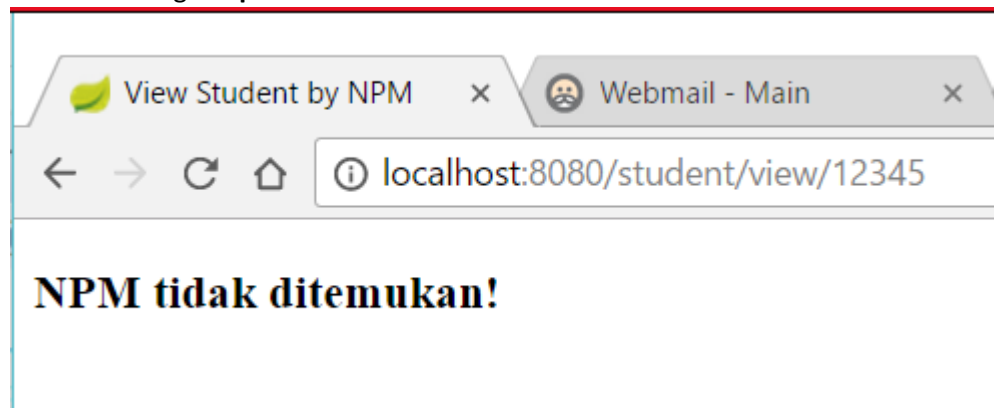
Kemudian, pada metode untuk controller, diimplementasikan untuk condition2 error diatas, dengan juga condition apabila berhasil menemukan npm

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

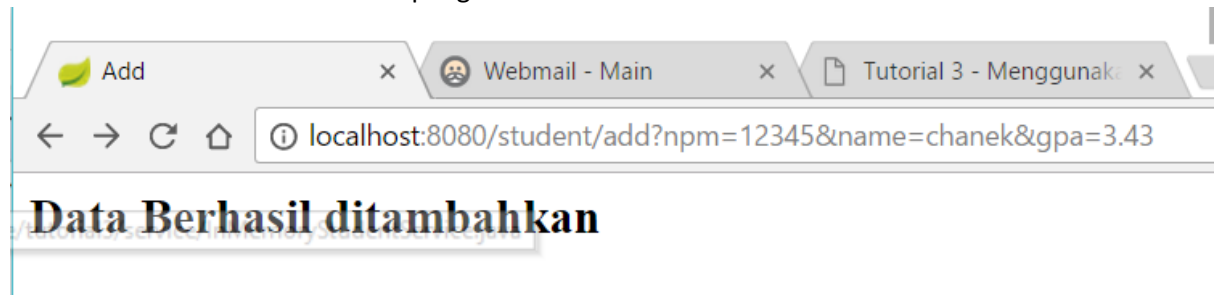
        if(students == null){
            return "errorPathNpm";
        } else{
            model.addAttribute("studentName", students.getName());
            model.addAttribute("studentNpm", students.getNpm());
            model.addAttribute("studentGpa", students.getGpa());
            return "view";
        }
    } else {
        return "errorPath";
    }
}
```

Maka keluarannya:

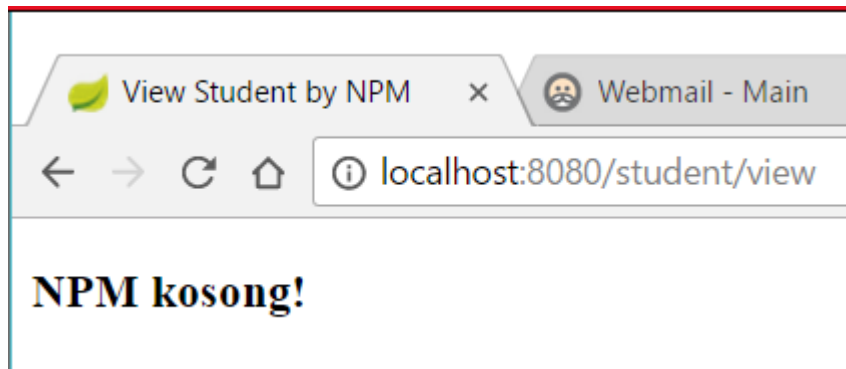
Jika error dengan **npm tidak ditemukan**



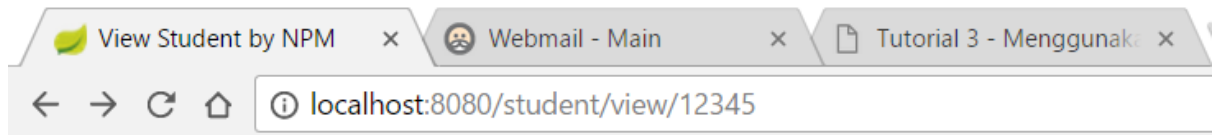
Menambahkan mahasiswa untuk pengecekan fitur



Jika error dengan **npm** tidak diberikan pada parameter



Jika berhasil menemukan npm

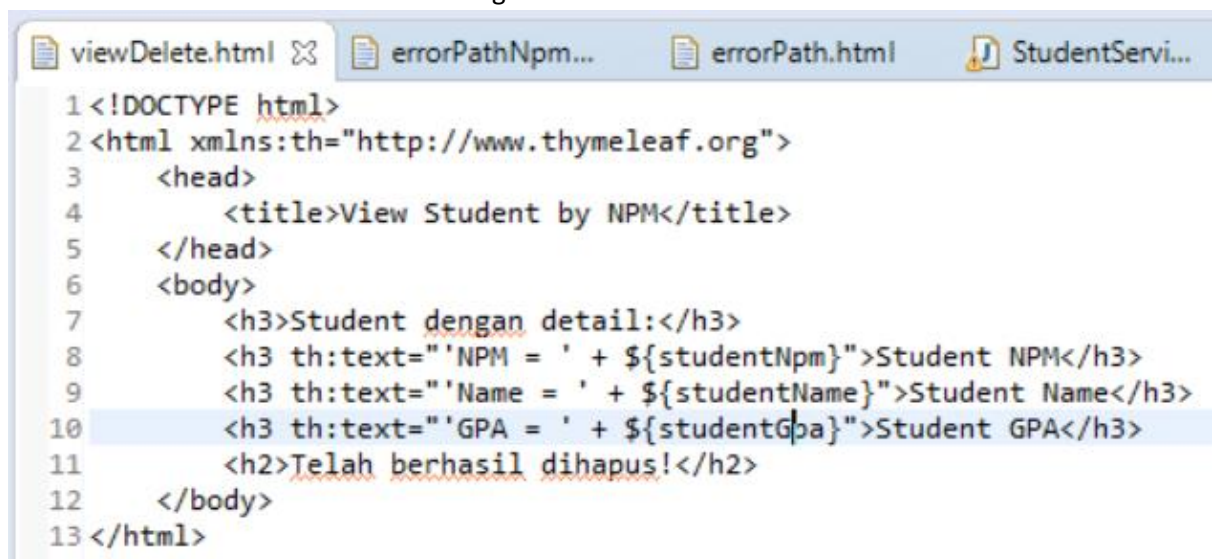


2. Untuk fitur Delete, saya menggunakan metode yang sama dengan view dengan Path Variable, namun saya ubah nama methodnya

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

        if(students == null){
            return "errorPathNpm";
        } else{
            model.addAttribute("studentName", students.getName());
            model.addAttribute("studentNpm", students.getNpm());
            model.addAttribute("studentGpa", students.getGpa());
            return "view";
        }
    } else {
        return "errorPath";
    }
}
```

Kemudian bentuk view untuk delete dengan nama file viewDelete.html



```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>View Student by NPM</title>
5 </head>
6 <body>
7 <h3>Student dengan detail:</h3>
8 <h3 th:text="'NPM = ' + ${studentNpm}">Student NPM</h3>
9 <h3 th:text="'Name = ' + ${studentName}">Student Name</h3>
10 <h3 th:text="'GPA = ' + ${studentGpa}">Student GPA</h3>
11 <h2>Telah berhasil dihapus!</h2>
12 </body>
13 </html>
```

Lalu membuat method baru deleteStudent(String npm) pada interface2 yang ada

Pada StudentService

```
StudentSeri... viewDelete.html errorPathNpm.  
1 package com.example.tutorial3.service;  
2  
3 import java.util.List;  
4 import java.util.Optional;  
5  
6 import com.example.tutorial3.model.StudentMode  
7  
8 public interface StudentService {  
9     StudentModel selectStudent(String npm);  
10  
11     StudentModel deleteStudent(String npm);  
12  
13     List<StudentModel> selectAllStudents();  
14  
15     void addStudent(StudentModel student);  
16 }  
17
```

Pada InMemoryStudentService

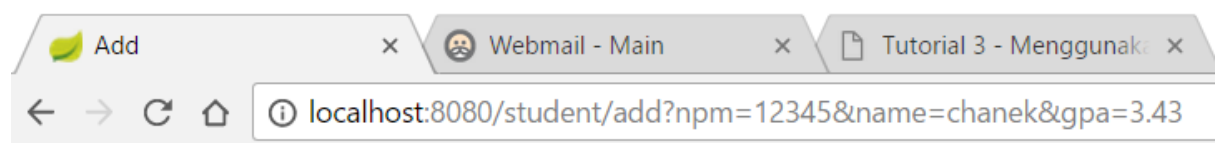
```
@Override  
public StudentModel deleteStudent(String npm) {  
    for(int i = 0; i < studentList.size(); i++){  
        if(studentList.get(i).getNpm().equalsIgnoreCase(npm)){  
  
            StudentModel temporaryStudent = studentList.get(i);  
            studentList.remove(i);  
  
            return temporaryStudent;  
        }  
    }  
    //Implement  
    return null;  
}
```

Kemudian method pada controller disesuaikan dengan kondisi delete yang baru, jika berhasil maka jalankan method deleteStudent() kemudian menggunakan data student Yang telah di delete.

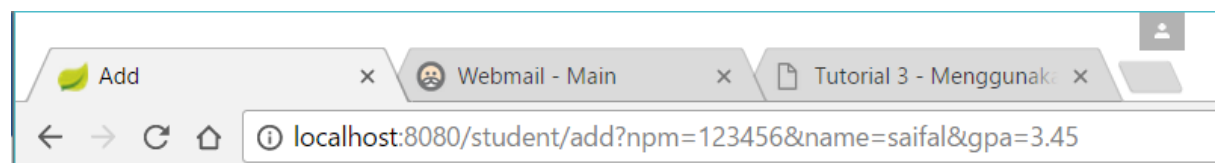
```
@RequestMapping(value={"/student/delete", "/student/delete/{npm}"})
public String studentDeletePath (@PathVariable Optional<String> npm, Model model)
{
    if (npm.isPresent()) {
        StudentModel students = studentService.selectStudent(npm.get());

        if(students == null){
            return "errorPathNpm";
        } else{
            StudentModel deleted = studentService.deleteStudent(npm.get());
            model.addAttribute("studentName", deleted.getName());
            model.addAttribute("studentNpm", deleted.getNpm());
            model.addAttribute("studentGpa", deleted.getGpa());
            return "viewDelete";
        }
    } else {
        return "errorPath";
    }
}
```

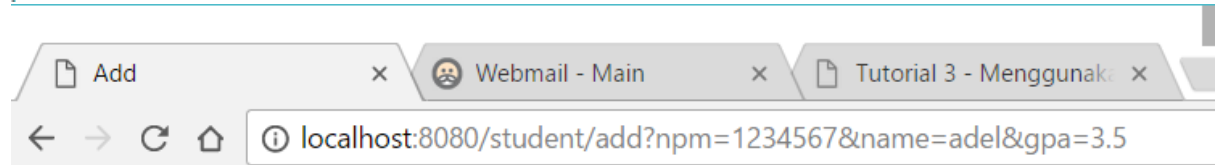
Memvalidasi apakah metode berhasil:



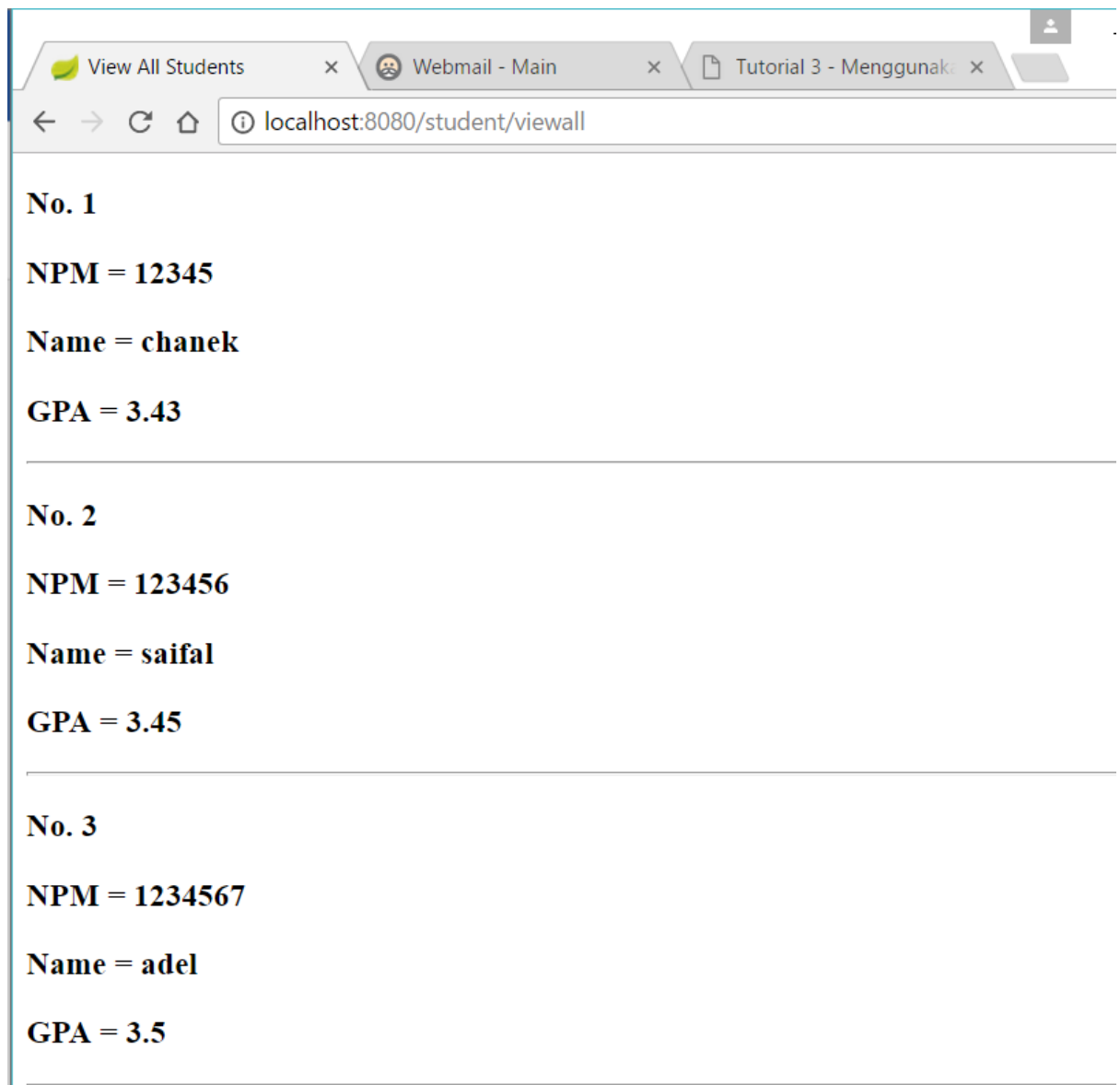
Data Berhasil ditambahkan



Data Berhasil ditambahkan



Data Berhasil ditambahkan

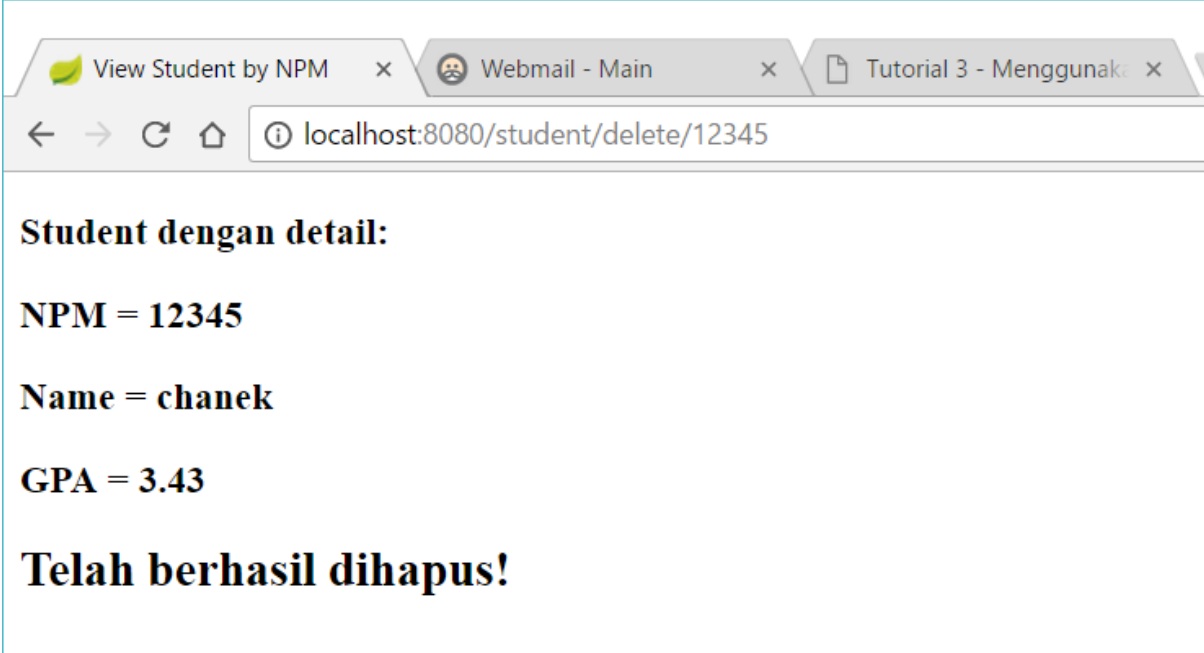


The screenshot shows a web browser window with the address bar displaying `localhost:8080/student/viewall`. The browser has three tabs open: "View All Students", "Webmail - Main", and "Tutorial 3 - Menggunakan". The page content displays a list of students, with the first two visible as follows:

No.	NPM	Name	GPA
No. 1	NPM = 12345	Name = chanek	GPA = 3.43
No. 2	NPM = 123456	Name = saifal	GPA = 3.45
No. 3	NPM = 1234567	Name = adel	GPA = 3.5

Pada screenshot diatas, dapat dilihat bahwa ada 2 mahasiswa bernama chanek, saifal, dan Adel (setelah ditambahkan)

Lalu kita coba delete mahasiswa dengan NPM 12345 (chanek)



The screenshot shows a web browser with three tabs: 'View Student by NPM', 'Webmail - Main', and 'Tutorial 3 - Menggunakan'. The address bar displays 'localhost:8080/student/delete/12345'. The page content shows the details of the student to be deleted:

Student dengan detail:

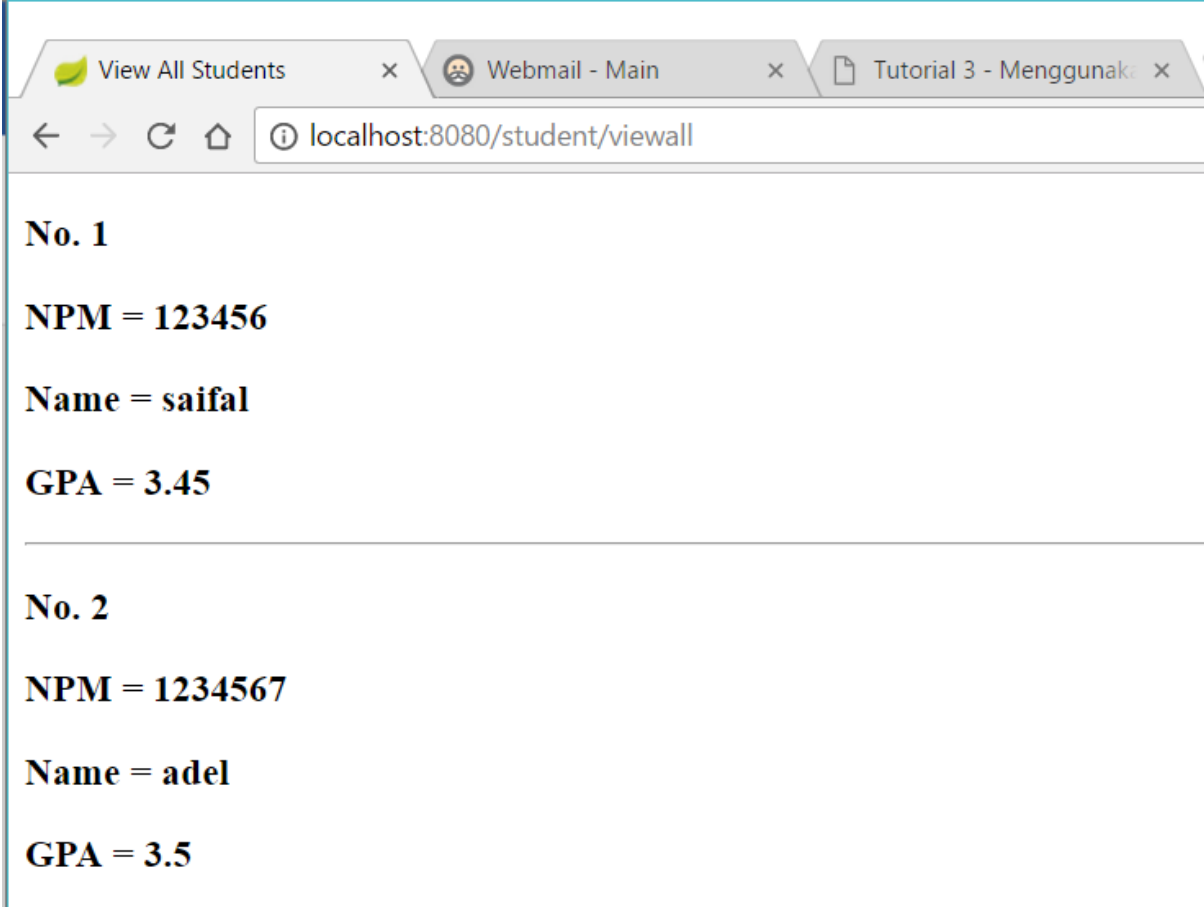
NPM = 12345

Name = chanek

GPA = 3.43

Telah berhasil dihapus!

Lalu kita lihat apakah terhapus



The screenshot shows a web browser with three tabs: 'View All Students', 'Webmail - Main', and 'Tutorial 3 - Menggunakan'. The address bar displays 'localhost:8080/student/viewall'. The page content shows a list of students:

No. 1

NPM = 123456

Name = saifal

GPA = 3.45

No. 2

NPM = 1234567

Name = adel

GPA = 3.5

Mahasiswa dengan NPM 12345 (chanek) berhasil dihapus

Ringkasan

Pada tutorial kali ini, saya mempelajari lebih dalam mengenai class Model dalam MVC, dalam tutorial ini saya juga belajar mengenai bagaiman penggunaan interface yang baik dalam menggunakan MVC, juga mempraktekan penggunaan java util seperti List. Selain itu, saya belajar mengenai bagaimana cara kerja penyimpanan data untuk springboot, yang didapatkan melalui requestparam dan pathvariable. Serta untuk tambahan terakhir, saya juga mempelajari bagaimana cara melakukan looping pada view.