

Sum Up yang dipelajari pada Tutorial 3:

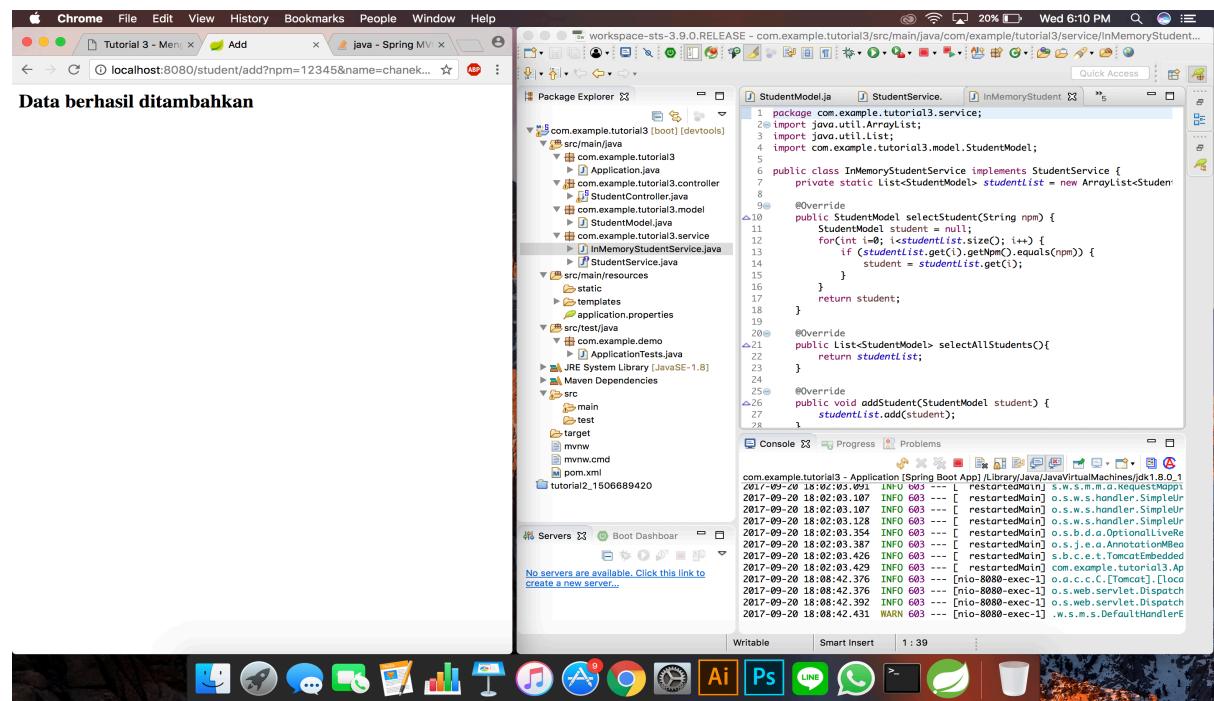
Hal yang saya pelajari pada Tutorial 3 adalah beberapa hal sebagai berikut – yang pertama yaitu penggunaan *model* dan *service* pada project spring boot. Saya juga belajar mengenai pengertian *model* yang merupakan sebuah objek yang merepresentasikan dan menyimpan suatu informasi. Sedangkan *service* merupakan suatu lapisan yang menjadi mediator antara *controller* dengan *database*. Latihan yang disediakan juga membantu pemahaman saya dalam melakukan implementasi dengan ide saya sendiri.

Jalankan program dan buka

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

Pertanyaan 1: apakah hasilnya?

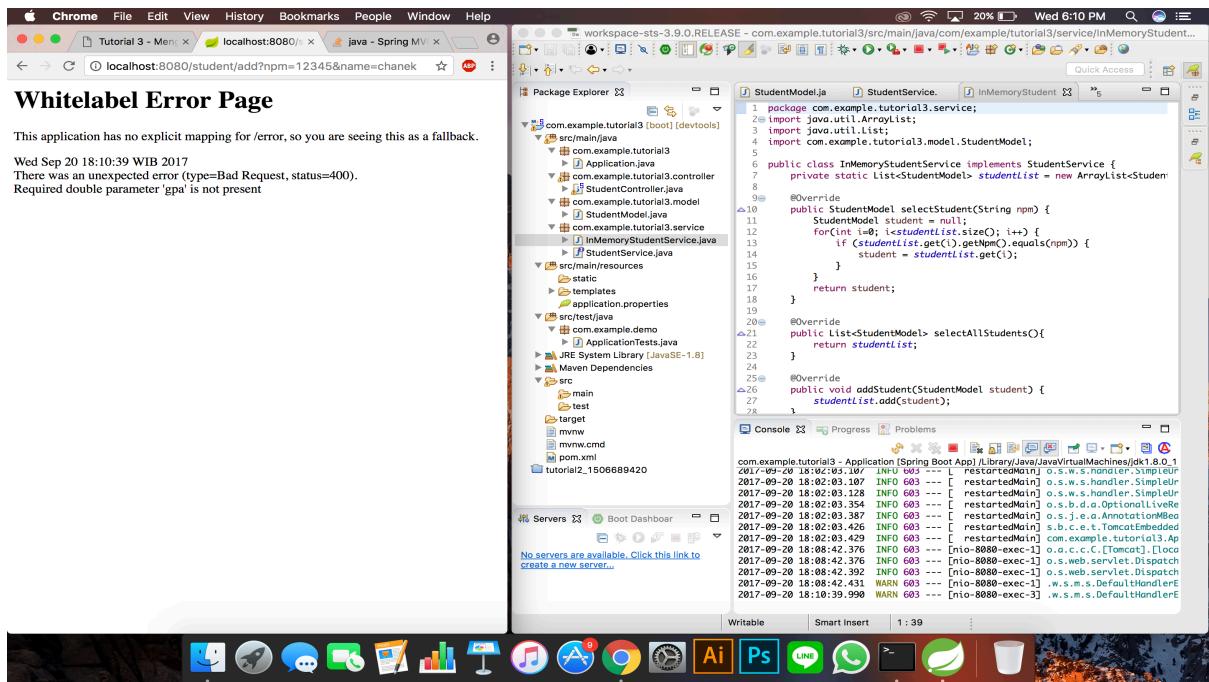
Berhasil.



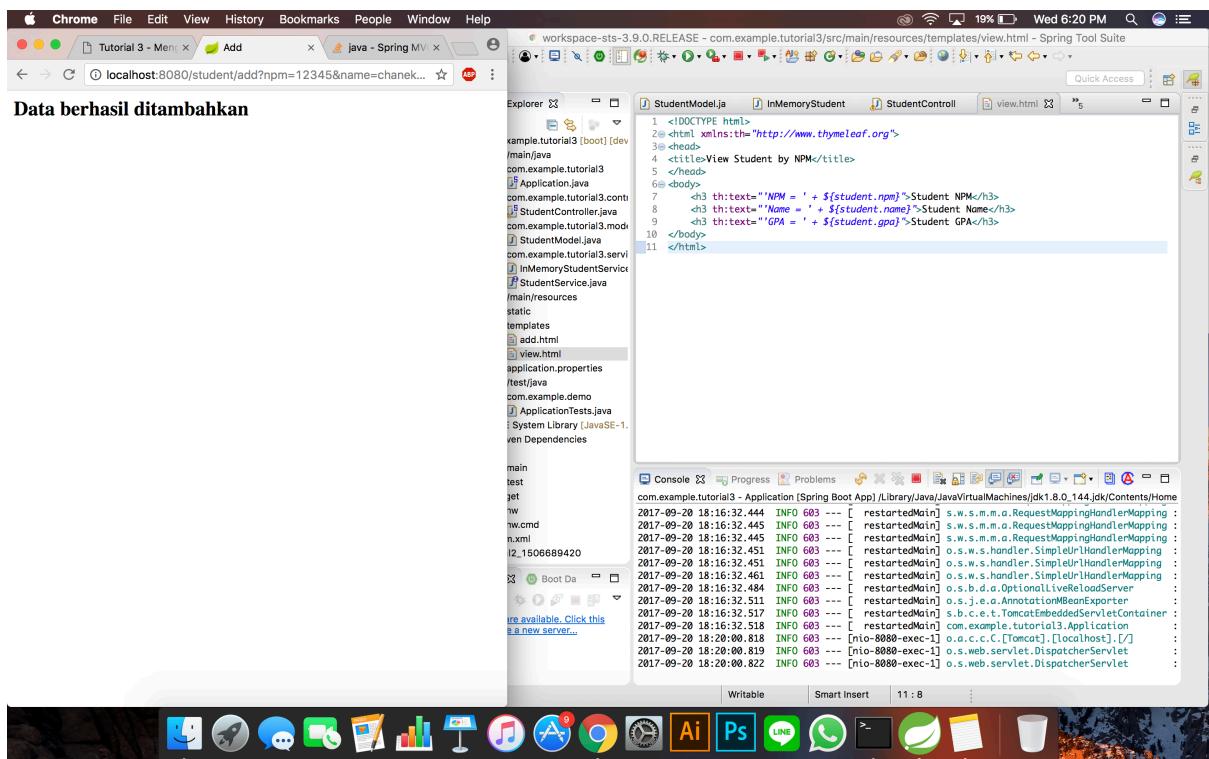
localhost:8080/student/add?npm=12345&name=chanek

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

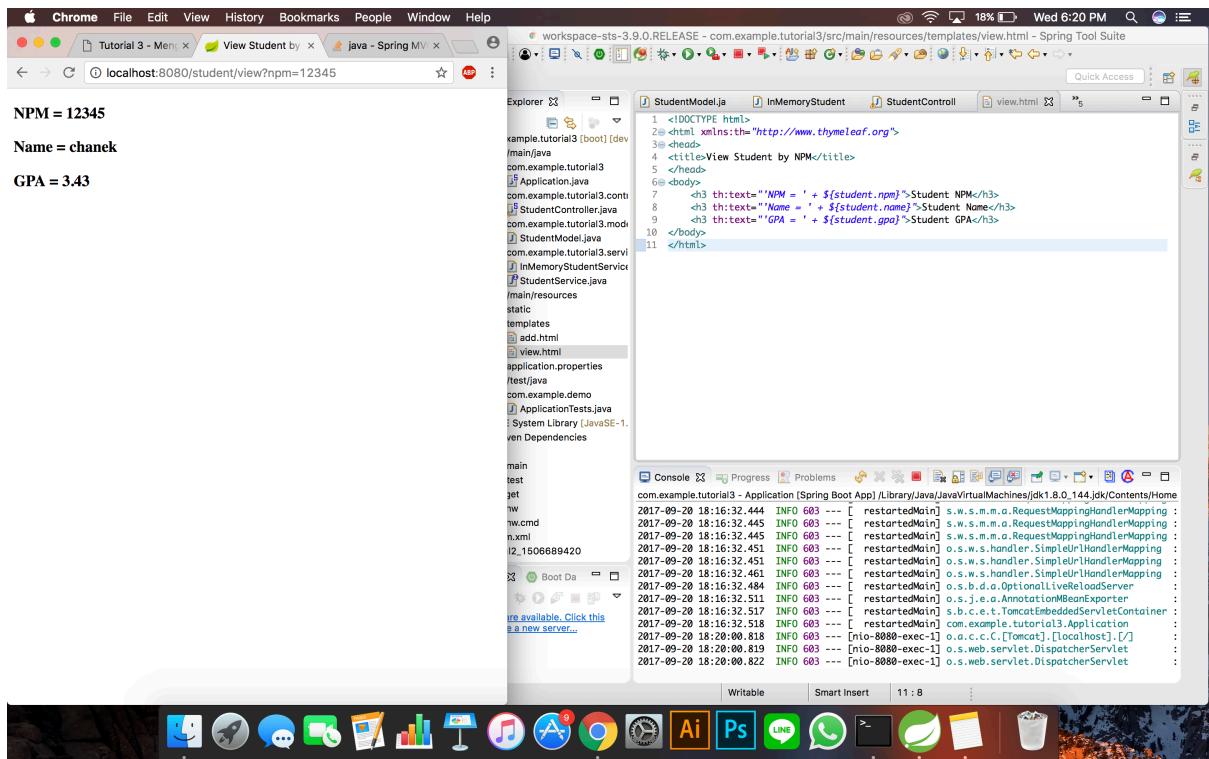
Error, karena pelajar dengan npm tersebut sudah ada dan perintah tersebut tidak memasukkan gpa yang merupakan requirement.



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



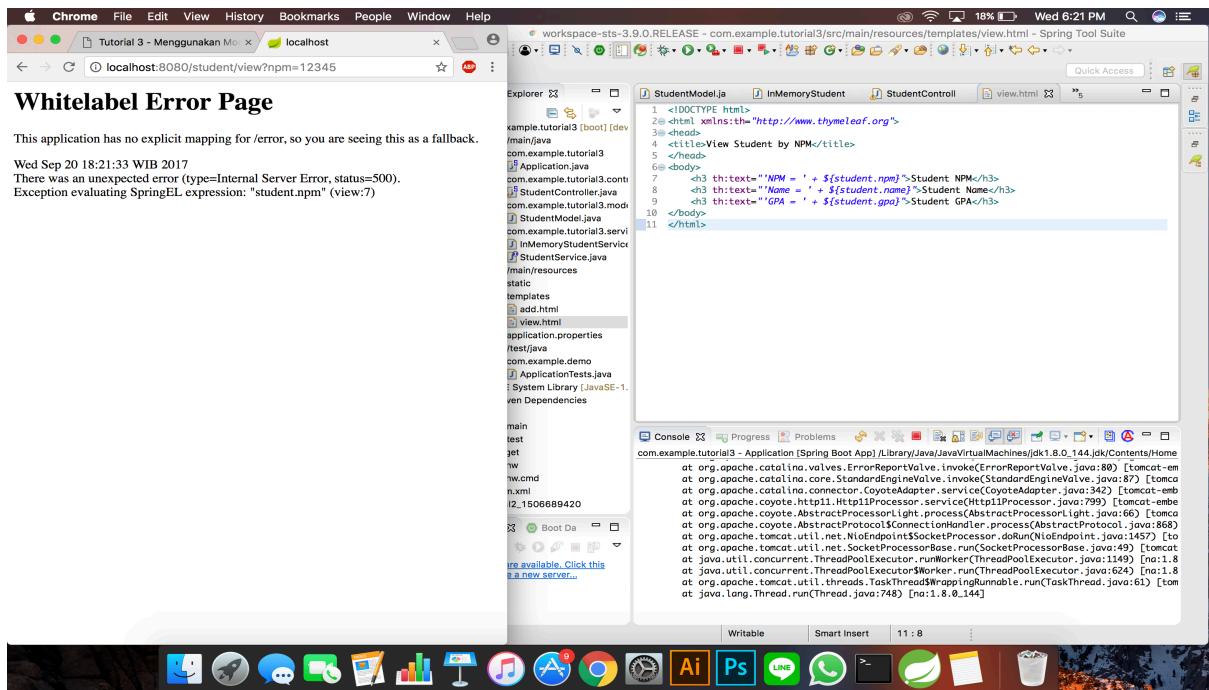
Lalu buka localhost:8080/student/view?npm=12345
Pertanyaan 3: apakah data Student tersebut muncul? Jika tidak, mengapa?
Muncul.



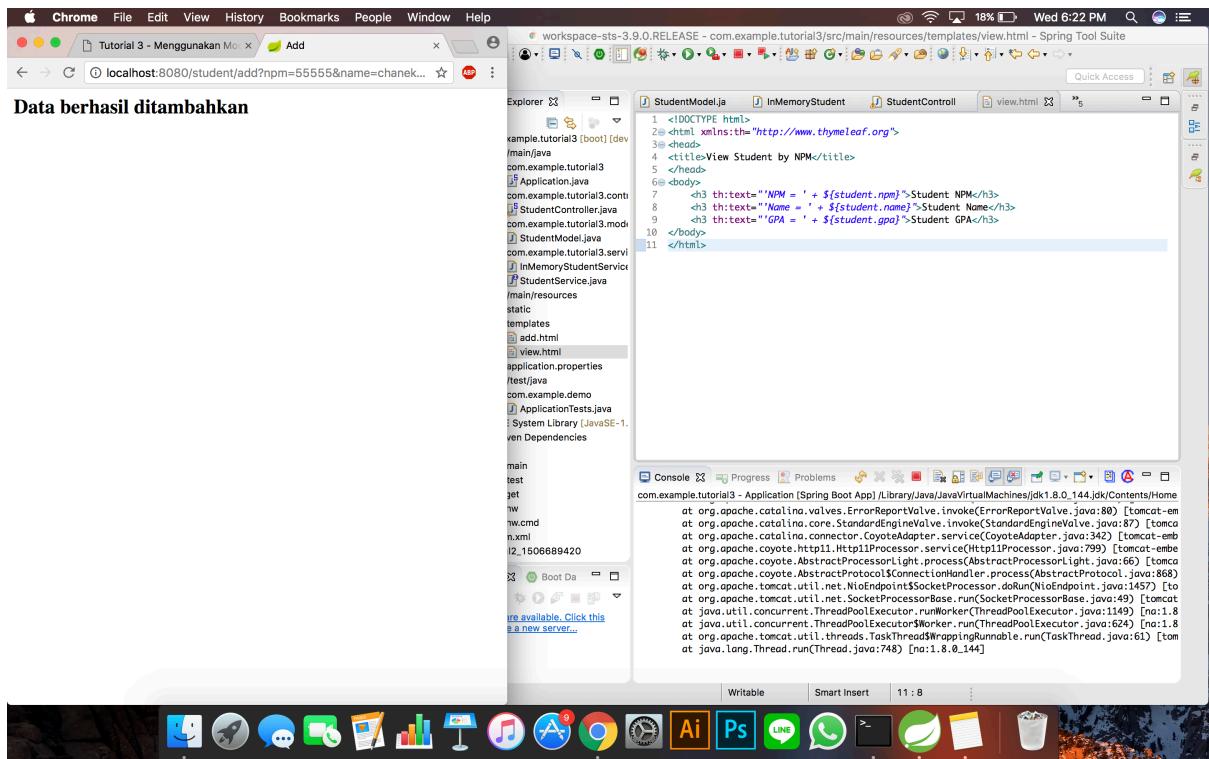
Coba matikan program dan jalankan kembali serta buka localhost:8080/student/view?npm=12345

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

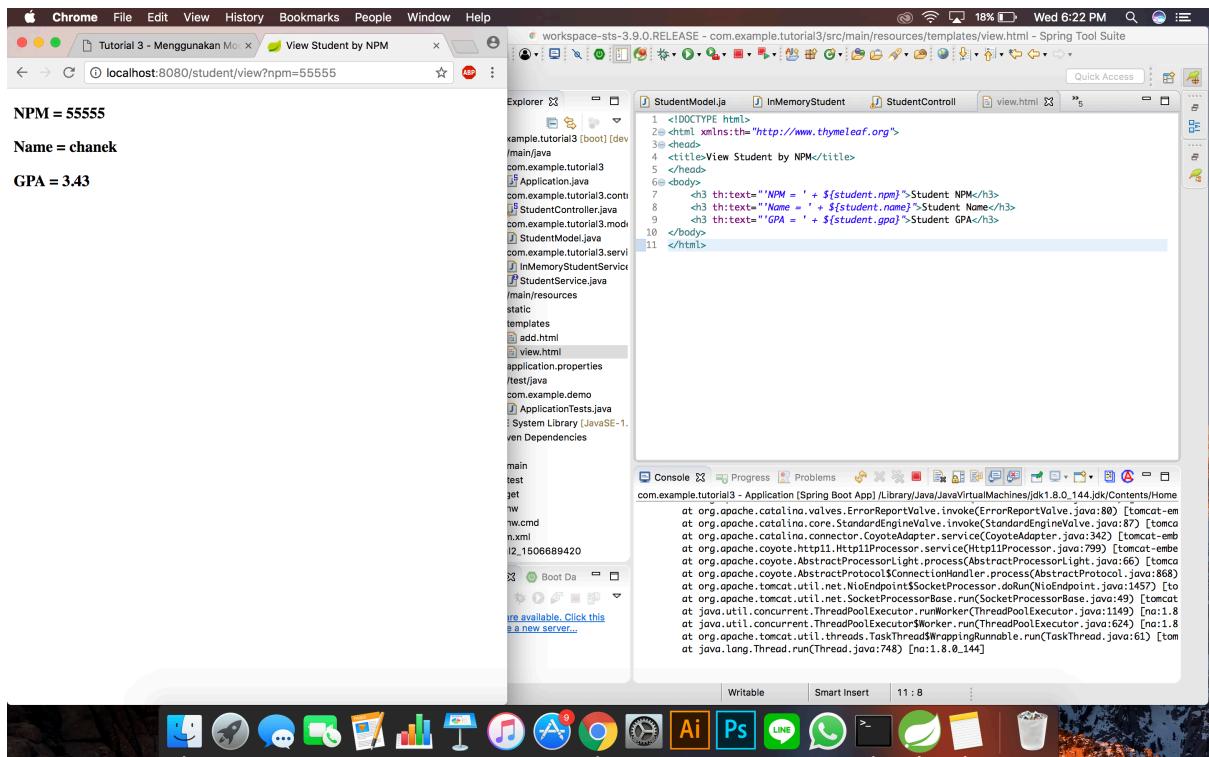
Tidak. Karena program dimatikan terlebih dahulu sehingga data yang sudah dimasukkan telah hilang.



Coba tambahkan data Student lainnya dengan NPM yang berbeda.

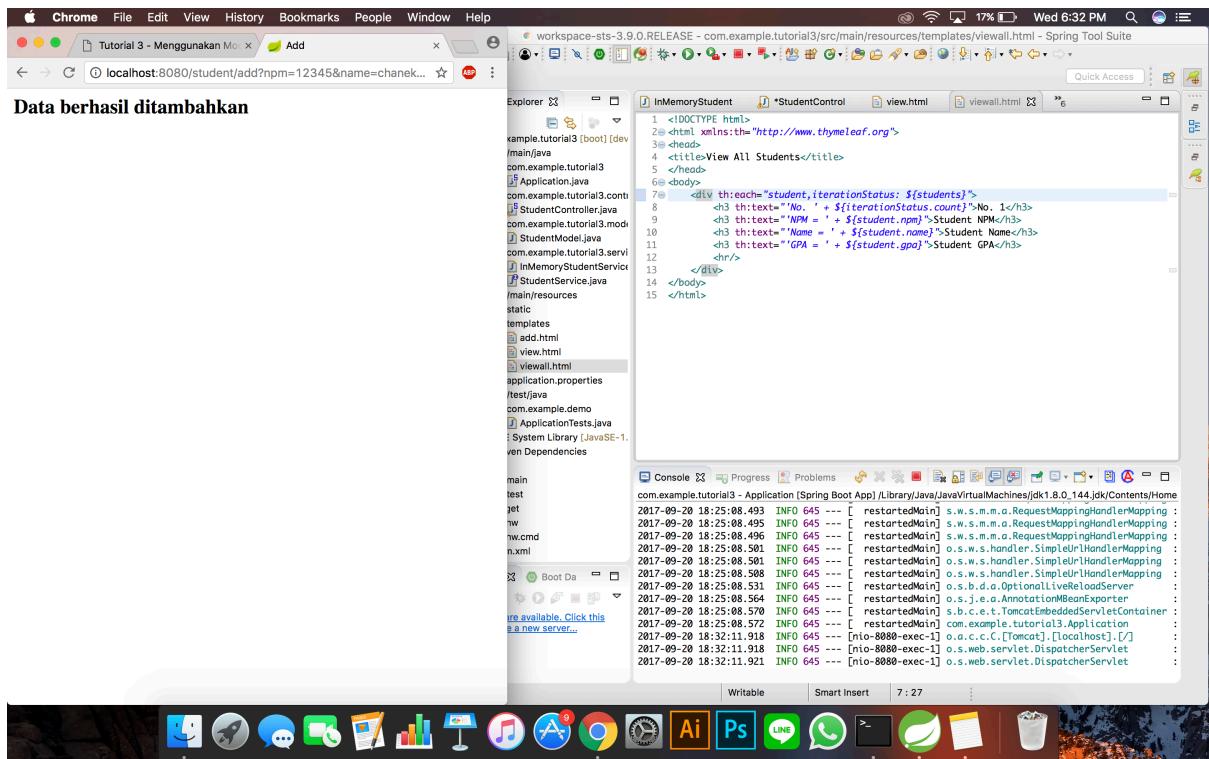


localhost:8080/student/view?npm=5555

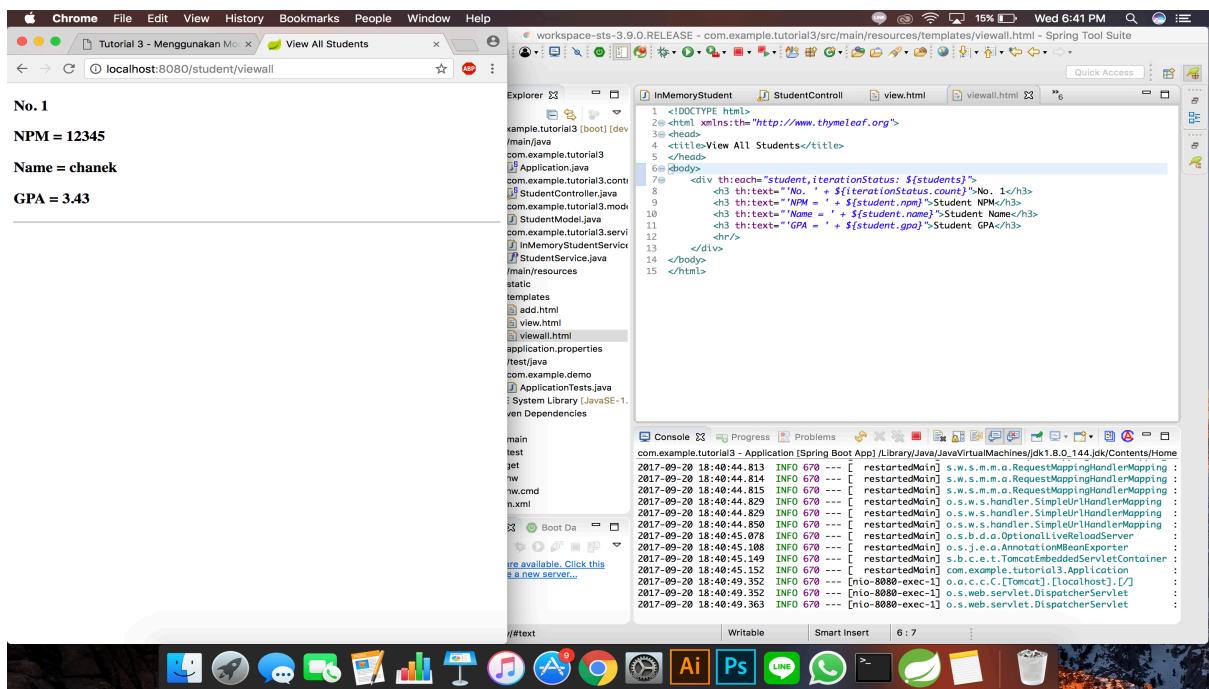


Jalankan program dan buka

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



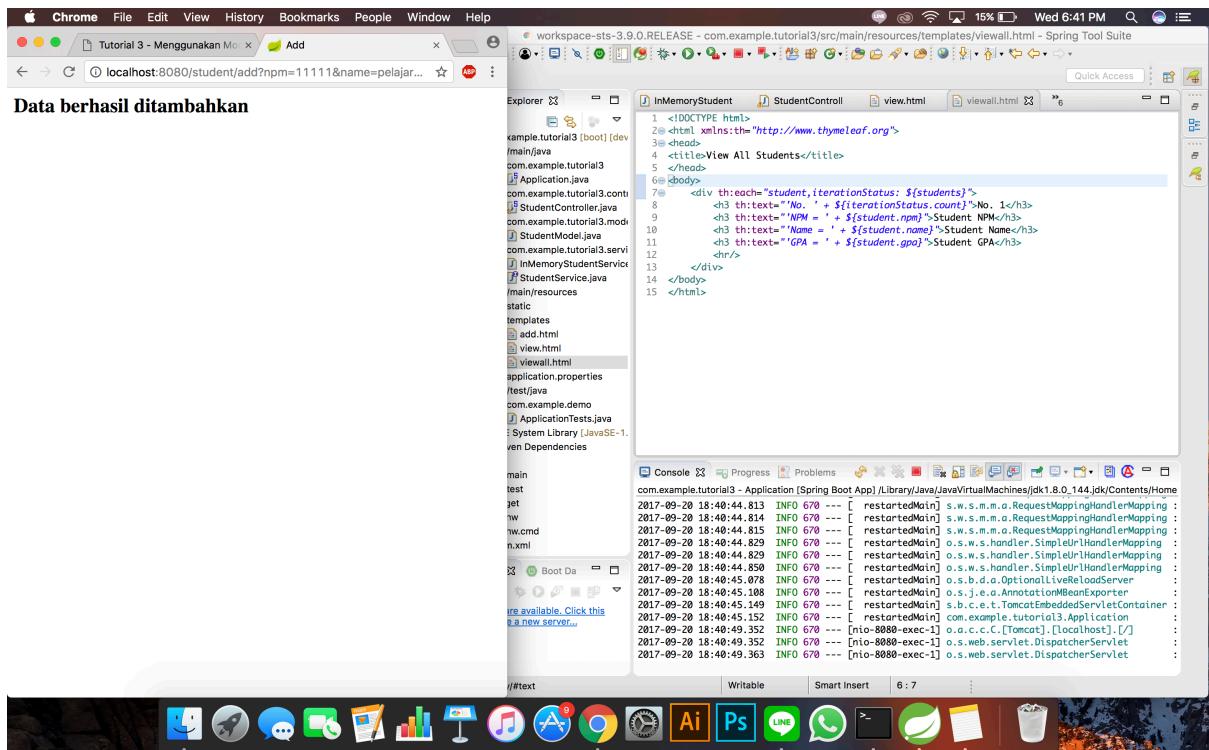
lalu buka localhost:8080/student/viewall,



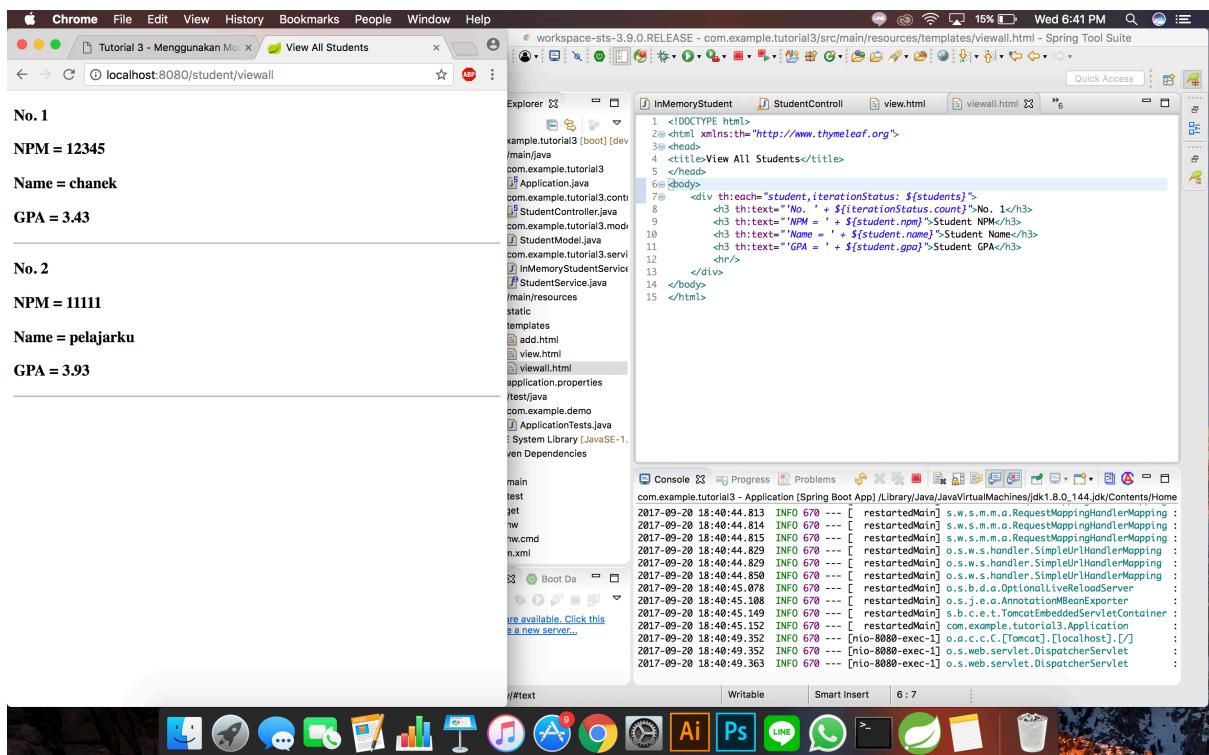
Pertanyaan 5: apakah data Student tersebut muncul?

Ya.

Coba tambahkan data Student lainnya dengan NPM yang berbeda



lalu buka localhost:8080/student/viewall,



Pertanyaan 6: Apakah semua data Student muncul?

Ya.

Latihan

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

Jawab

StudentController.java:

```
@RequestMapping("/student/view/{npm}")
public String view(@PathVariable String npm, Model model) {
    StudentModel student = studentService.selectStudent(npm);
    if(student == null) {
        return "404";
    } else {
        model.addAttribute("student", student);
        return "view";
    }
}
```

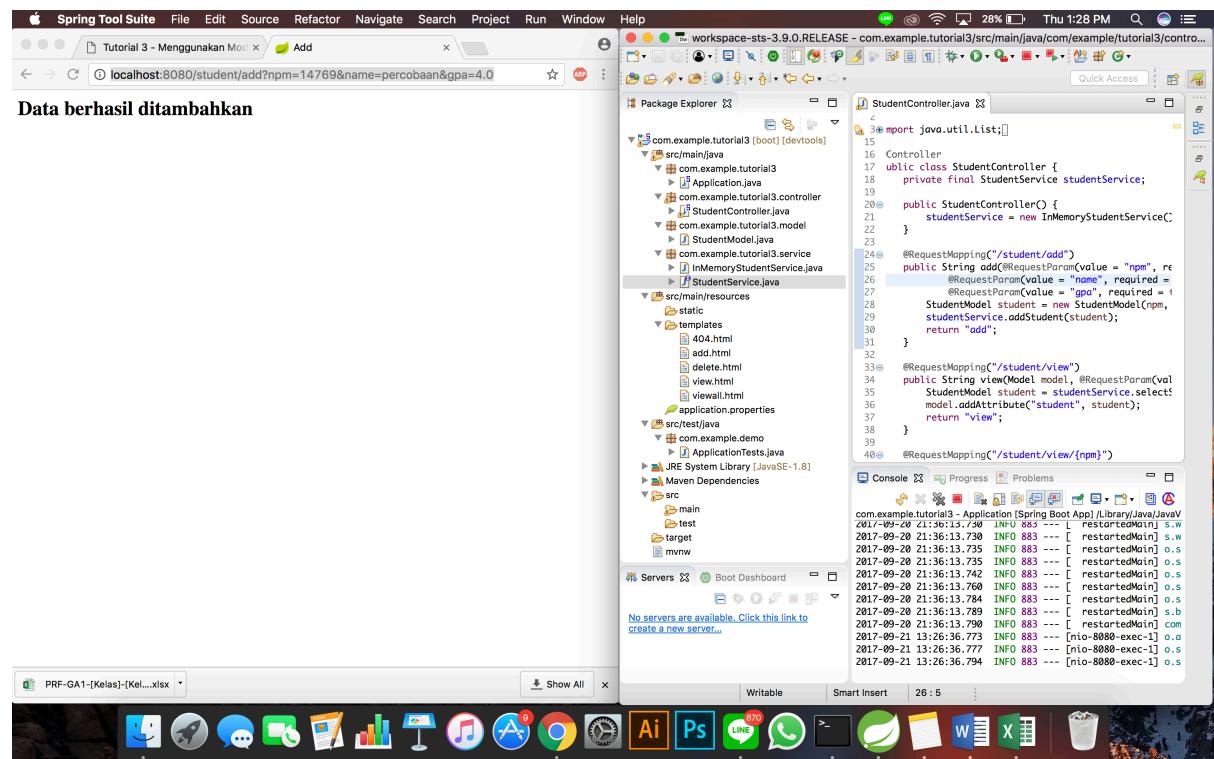
/templates – view.html:

```
<!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>
```

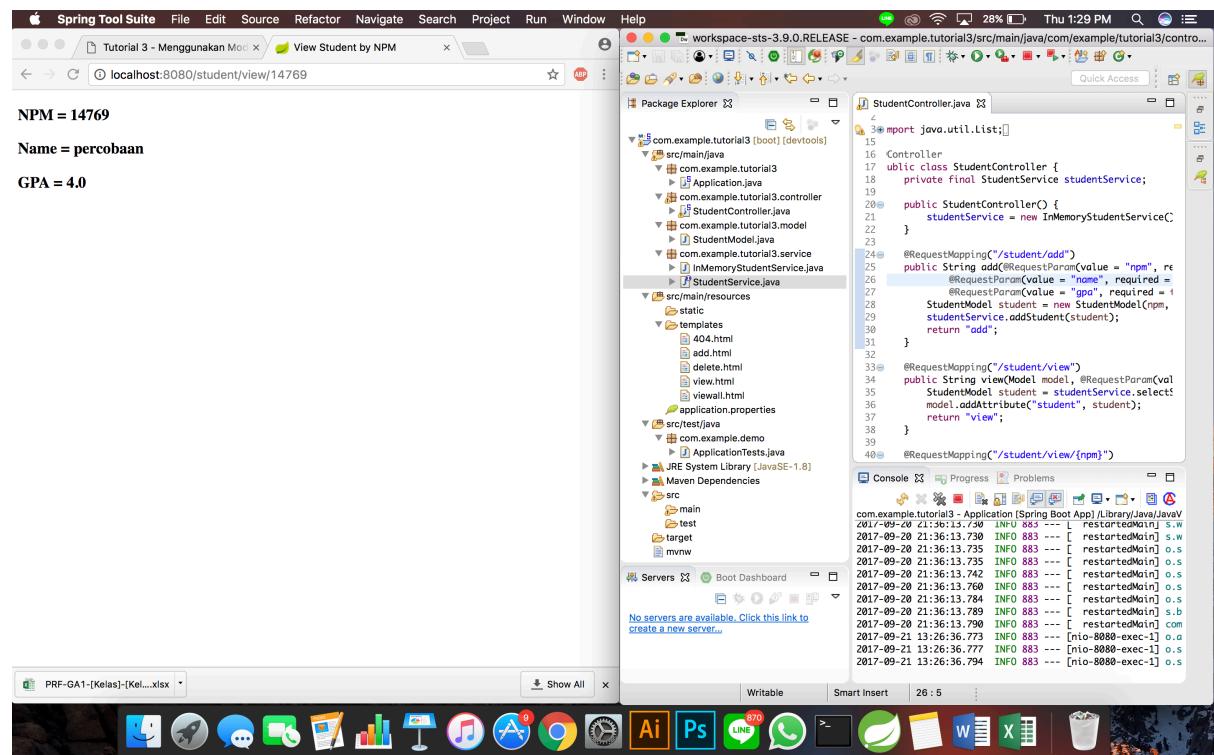
/templates – 404.html:

```
<html>
<head>
<title>404</title>
</head>
<body>
    <h2>Nomor NPM kosong atau tidak ditemukan</h2>
</body>
</html>
```

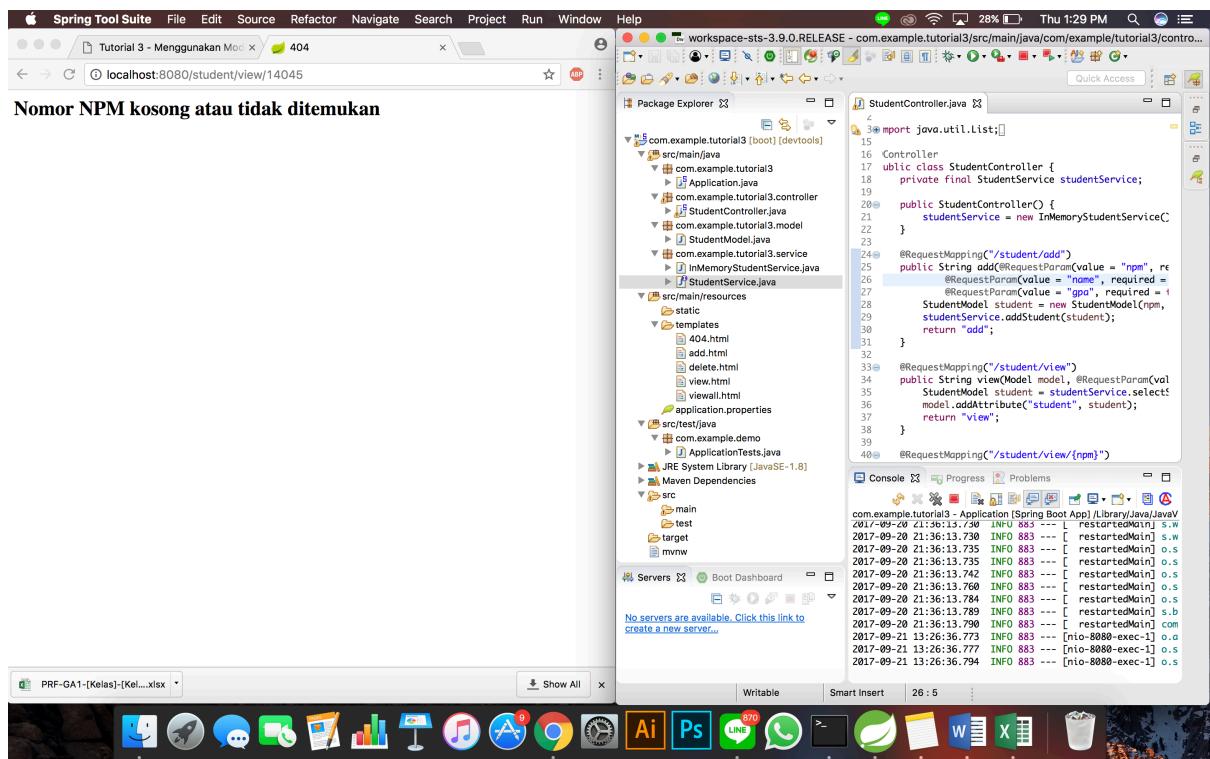
localhost:8080/student/add?npm=14769&name=percobaan&gpa=4.0



localhost:8080/student/view/14769



localhost:8080/student/view/14045



Penjelasan:

Pertama, saya membuat @RequestMapping yang mirip dengan view yang sudah ada sebelumnya. Saya menambahkan "/{npm}" agar dapat dicari dengan memasukkan npm pada url. Selanjutnya apabila nomor npm tidak ditemukan maka return 404.html, apabila ditemukan maka return view.html.

2. Tambahkan fitur untuk melakukan delete Student berdasarkan NPM. 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.

Jawab

InMemoryStudentService.java:

```
@Override  
public void deleteStudent(StudentModel student) {  
    studentList.remove(student);  
}
```

StudentService.interface:

```
void deleteStudent(StudentModel student);
```

Student Controller.java:

```
@RequestMapping("/student/delete/{npm}")  
public String delete(@PathVariable String npm, Model model) {  
    StudentModel student = studentService.selectStudent(npm);  
    if(student == null) {  
        return "404";  
    } else {  
        studentService.deleteStudent(student);  
        return "delete";  
    }  
}
```

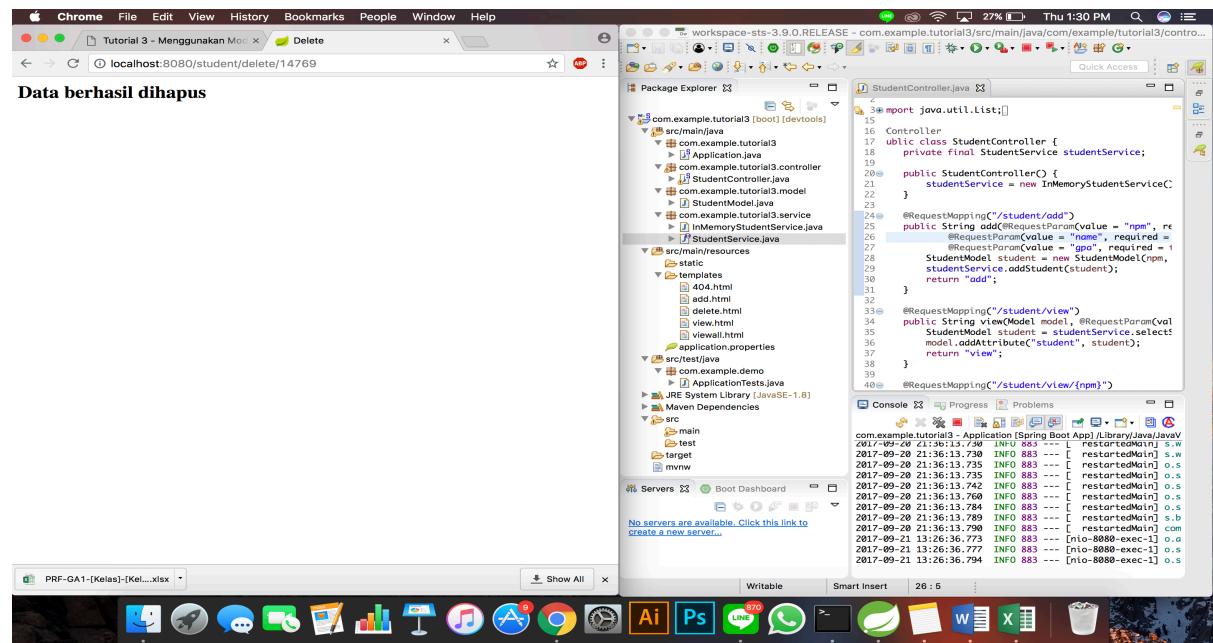
/templates – delete.html:

```
<html>  
<head>  
<title>Delete</title>  
</head>  
<body>  
    <h2>Data berhasil dihapus</h2>  
</body>  
</html>
```

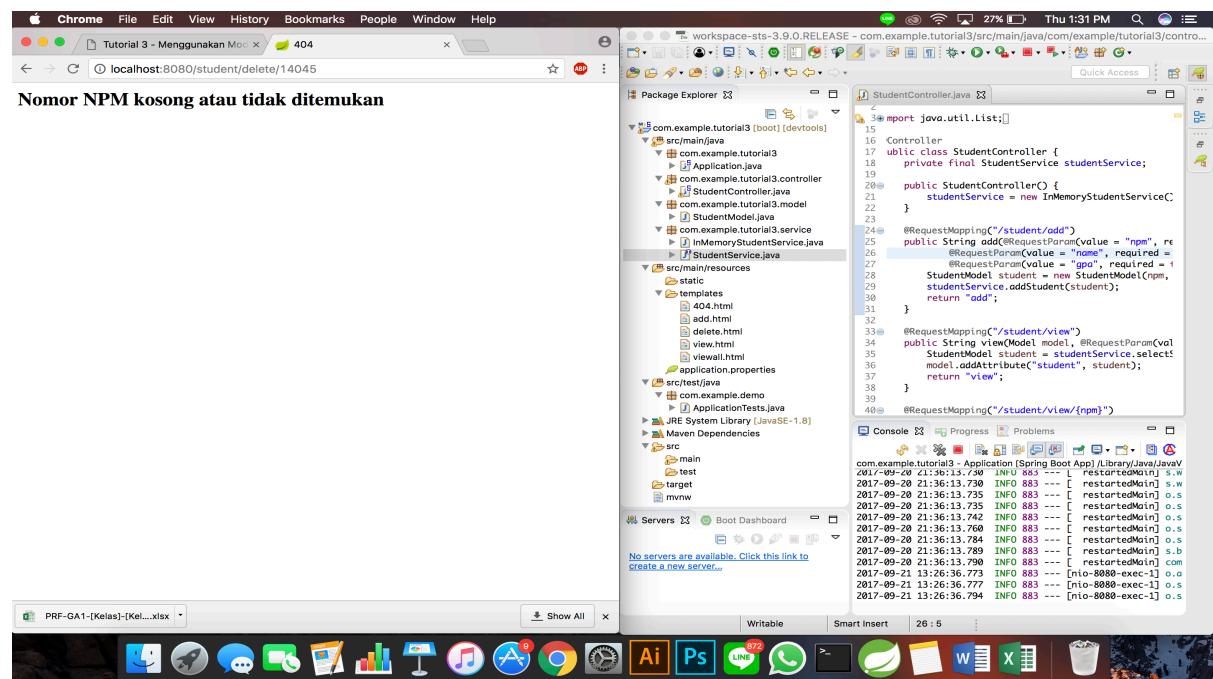
/templates – 404.html:

```
<html>
<head>
<title>404</title>
</head>
<body>
    <h2>Nomor NPM kosong atau tidak ditemukan</h2>
</body>
</html>
```

localhost:8080/student/delete/14769



localhost:8080/student/delete/14045



Penjelasan:

Pertama, saya membuat @RequestMapping delete. Saya menambahkan “delete/{npm}” agar dapat dihapus dengan memanggil delete dan memasukkan npm pada url. Selanjutnya apabila nomor npm tidak ditemukan maka return 404.html, apabila ditemukan maka return delete.html.