

Atikah Luthfiana

1506689250

ADPAP B

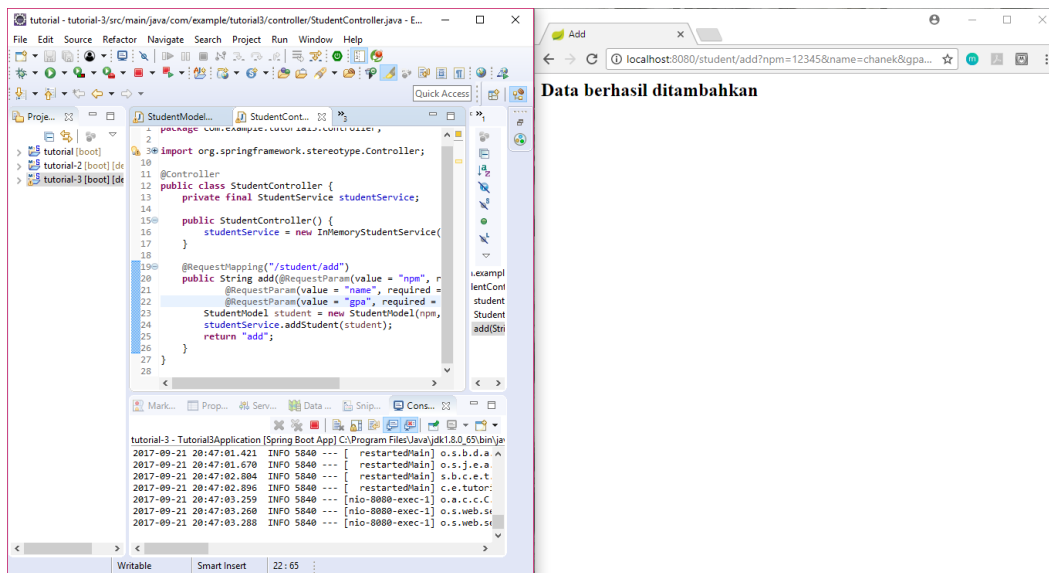
## Tutorial 3: Menggunakan Model dan Service pada Project Spring Boot

a. Tutorial kali ini membahas mengenai penggunaan *interface* yang sangat membantu dalam melakukan operasi-operasi yang dalam mvc dan html web yang digunakan. Pada tutorial ini saya juga belajar bahwa data yang ada pada program yang dijalankan akan hilang setelah program telah diberhentikan kemudian dijalankan kembali.

b. Tutorial dan Latihan

### - **Add Student**

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



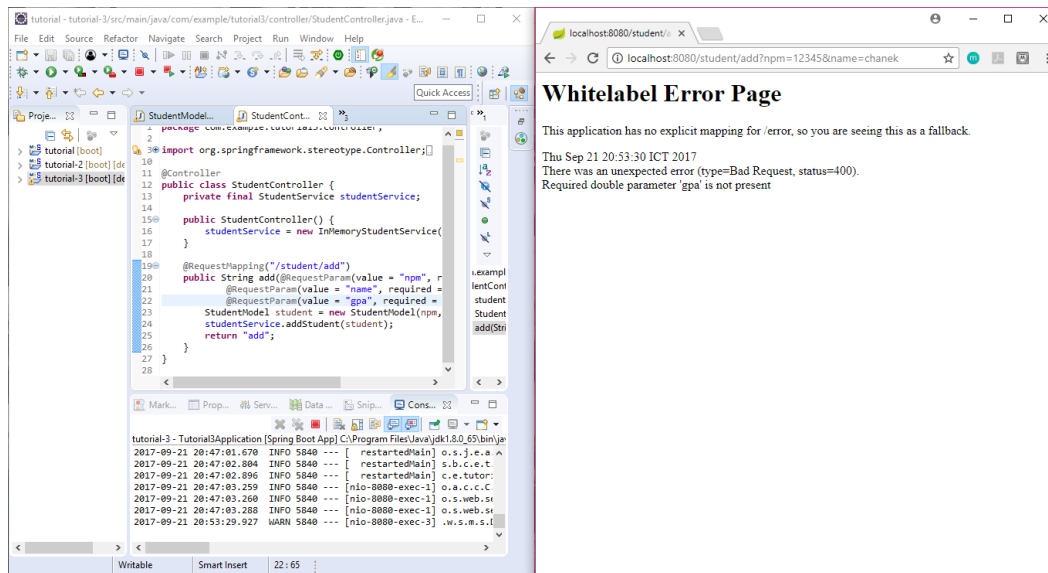
Tidak ada *error*, *student* chanek berhasil ditambahkan.

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

Atikah Luthfiana

1506689250

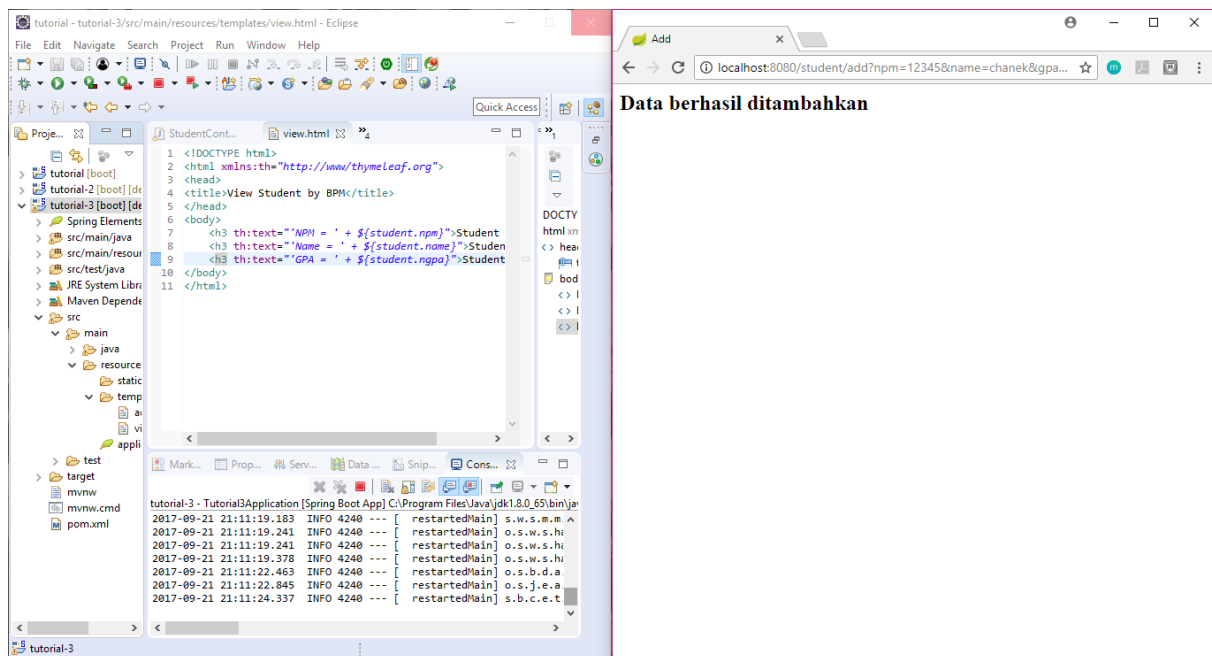
ADPAP B



Terjadi *error*. Hal ini disebabkan oleh tidak adanya nilai dari GPA yang diinput *user*.

#### - **View Student**

3. Hasil dari `localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43`

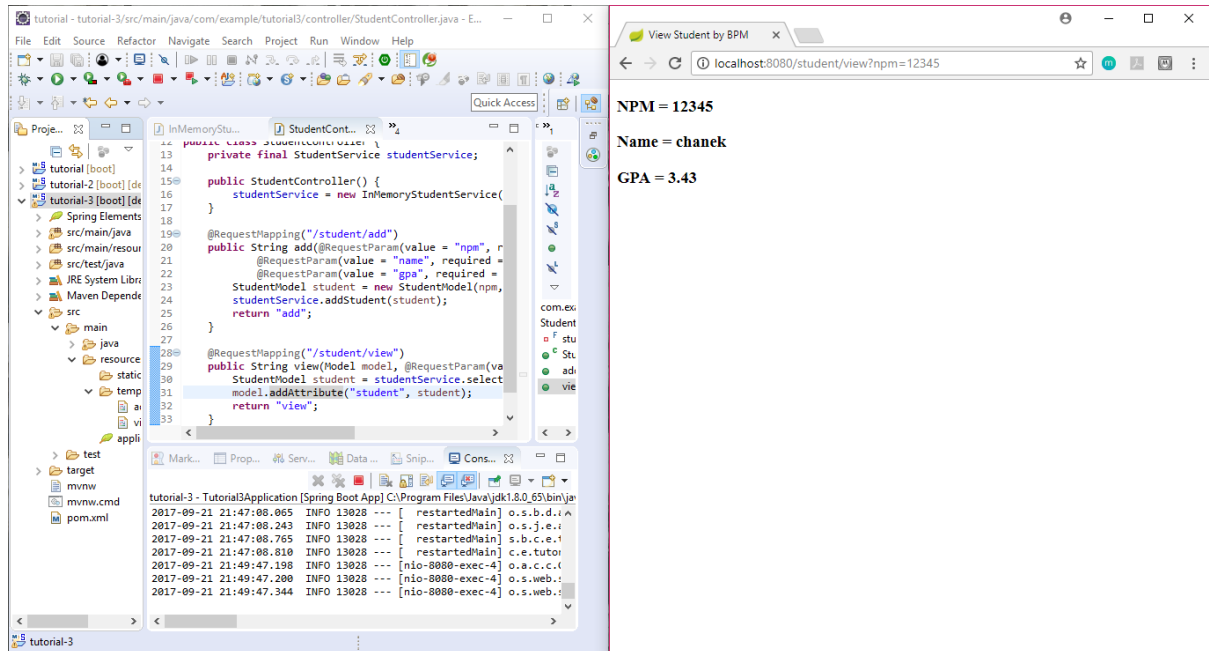


Chanek berhasil ditambahkan. Lalu dijalankan `localhost:8080/student/view?npm=12345`

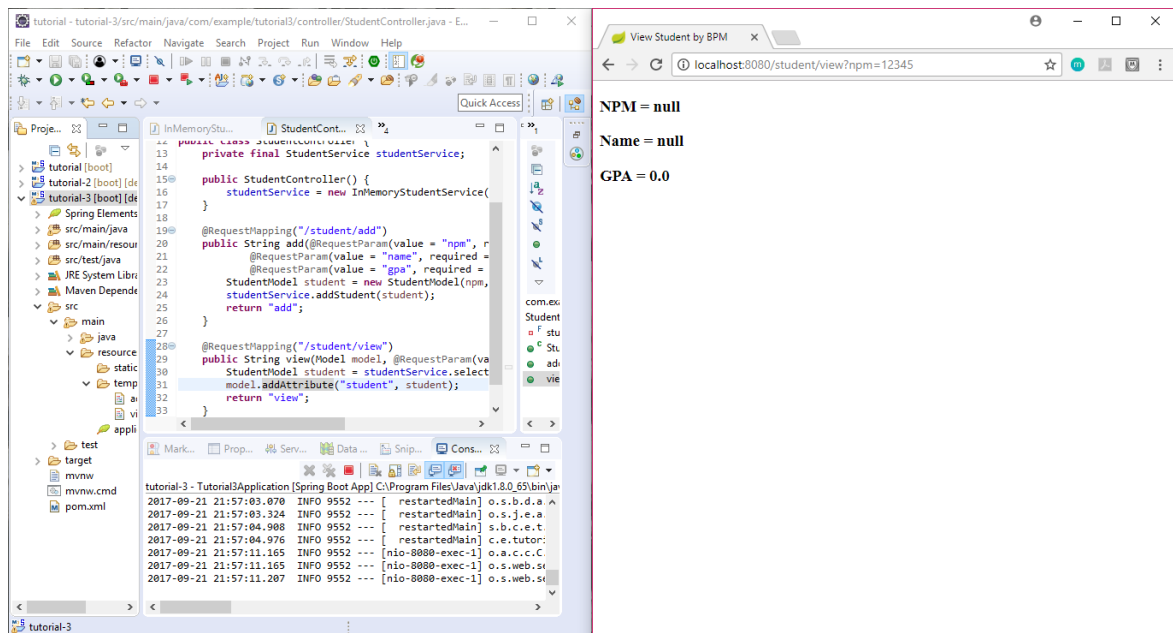
Atikah Luthfiana

1506689250

ADPAP B



#### 4. hasil dari localhost:8080/student/view?npm=12345



Data *student* tersebut tidak muncul dikarenakan kita belum melakukan *add student* sebelumnya. *Add student* pada nomor 4 hanya berlaku untuk *session* tersebut. Sehingga

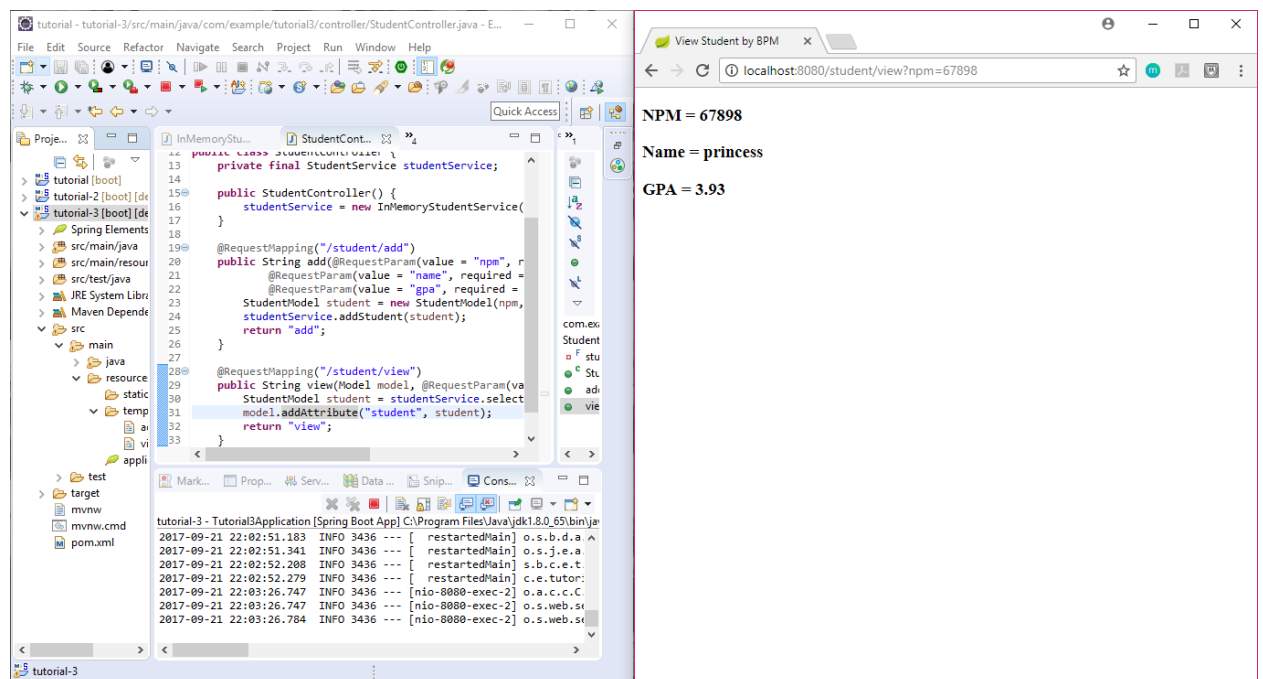
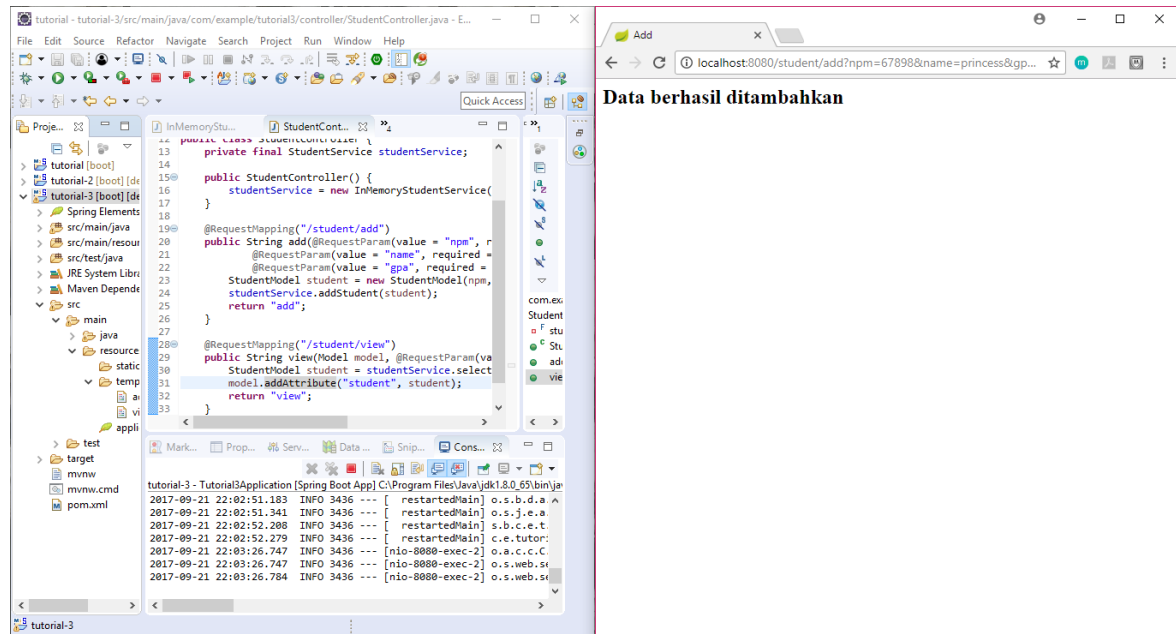
Atikah Luthfiana

1506689250

ADPAP B

ketika program dimatikan, *session* kembali ke awal dan data *student list* yang ada masih kosong.

##### 5. Menambahkan student baru dengan npm berbeda



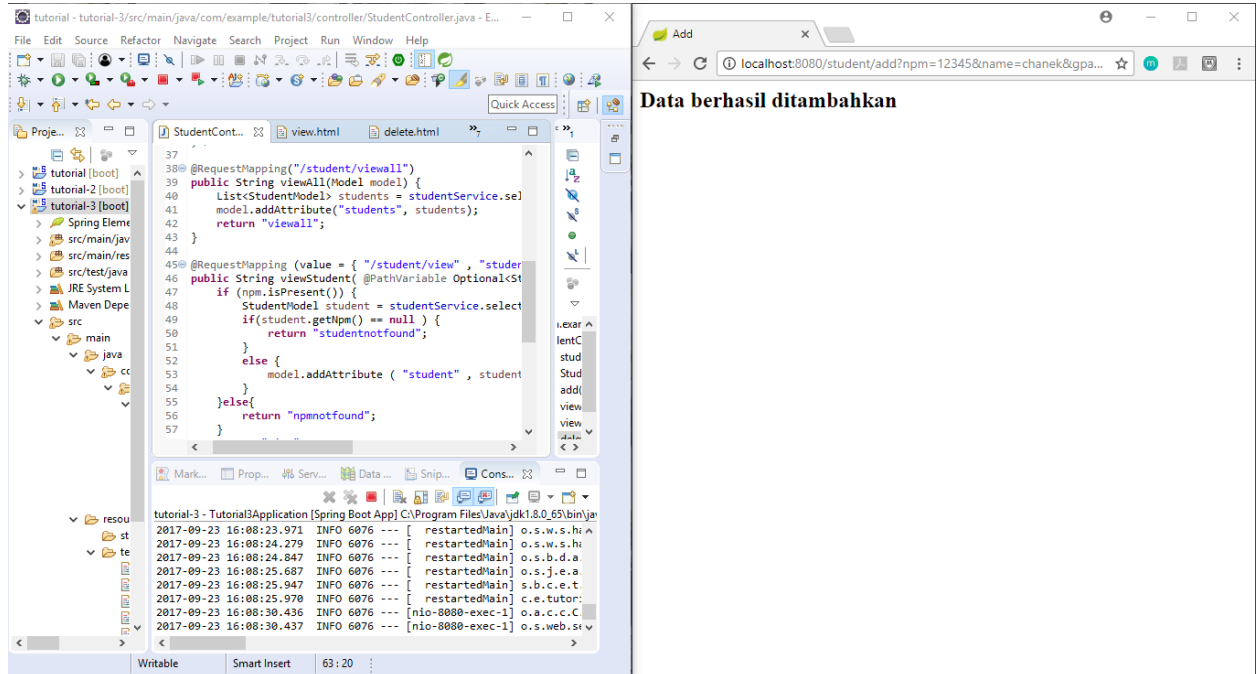
- View All Student

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

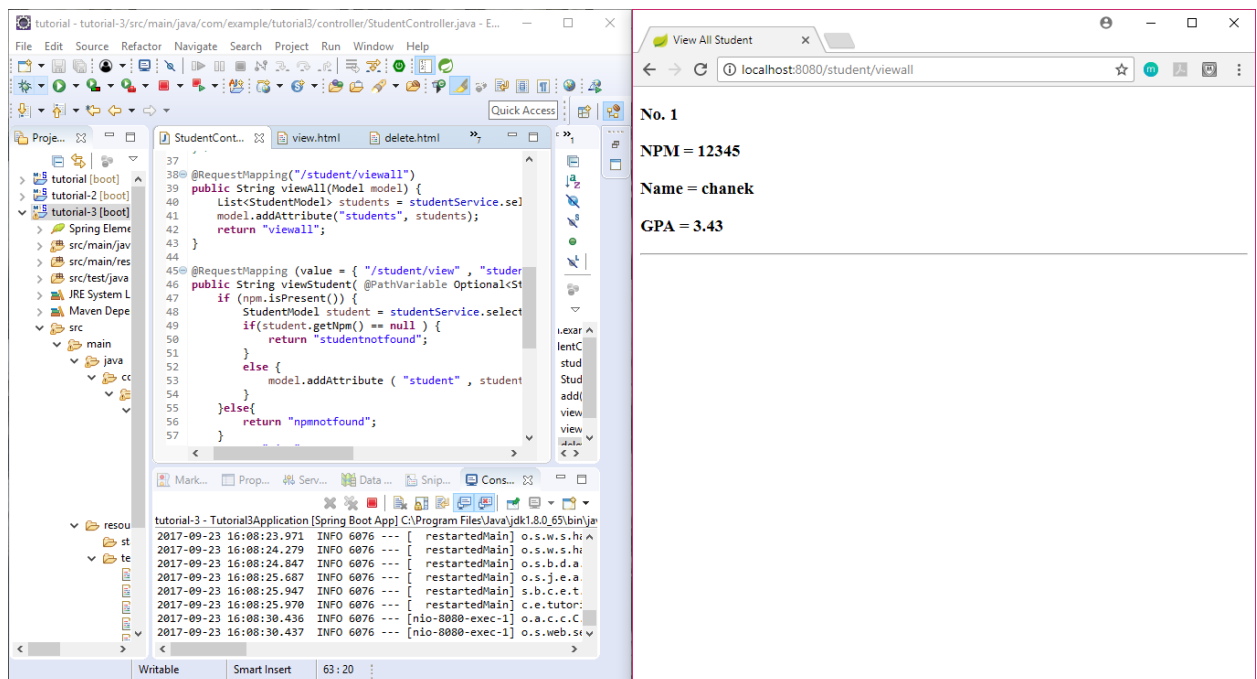
Atikah Luthfiana

1506689250

ADPAP B



Lalu menjalankan `localhost:8080/student/viewall`



Data *student* chanek muncul

Atikah Luthfiana

1506689250

ADPAP B

## 7. Menambahkan data lain

The first screenshot shows the IDE with the `StudentController.java` file. The `viewAll` method is highlighted, showing a list of students being added to the model. The browser window shows the URL `localhost:8080/student/add?npm=93373&name=cinta&gpa=3...` and the message "Data berhasil ditambahkan".

The second screenshot shows the IDE with the `viewStudent` method highlighted, which returns a single student model. The browser window shows the URL `localhost:8080/student/viewall` and displays two student records:

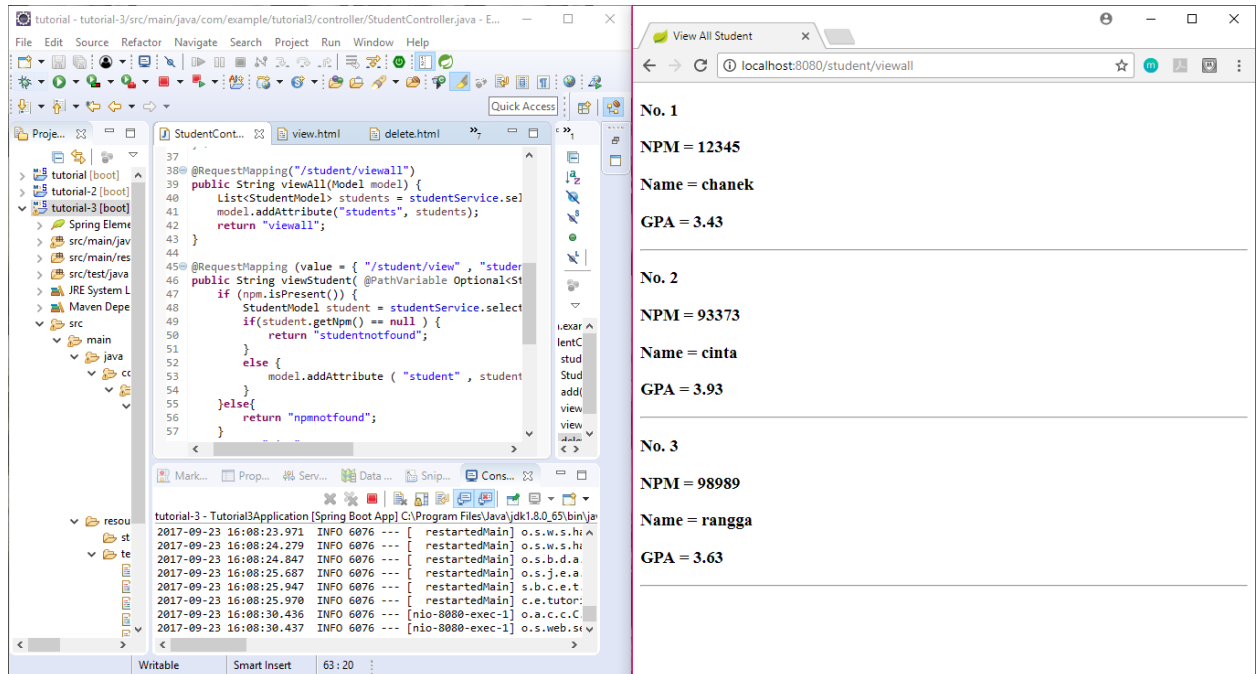
No.	NPM	Name	GPA
No. 1	12345	chanek	3.43
No. 2	93373	cinta	3.93

The third screenshot shows the IDE with the `viewAll` method highlighted. The browser window shows the URL `localhost:8080/student/add?npm=98989&name=rangga&gpa=...` and the message "Data berhasil ditambahkan".

Atikah Luthfiana

1506689250

ADPAP B



Semua data *student* yang berhasil dimasukkan muncul pada *html viewall*.

Latihan

1. Melihat data student berdasarkan npm

Langkah :

- Comment method view sebelumnya agar tidak terjadi error.

```
StudentService - new InMemoryStudentService(
20 }
21
22 @RequestMapping("/student/add")
23 public String add(@RequestParam(value = "npm", r
24                 @RequestParam(value = "name", required =
25                 @RequestParam(value = "gpa", required =
26                 StudentModel student = new StudentModel(npm,
27                 studentService.addStudent(student);
28                 return "add";
29 }
30 /*
31 @RequestMapping("/student/view")
32 public String view(Model model, @RequestParam(va
33                 StudentModel student = studentService.select
34                 model.addAttribute("student", student);
35                 return "view";
36 */
37
38 @RequestMapping("/student/viewall")
39 public String viewAll(Model model) {
40     List<StudentModel> students = studentService
```

- Membuat method viewStudent dengan menggunakan Path Variable

```
StudentService InMemoryStu... StudentCont... delete.html npmnotfound... studentnotf...
38 @RequestMapping("/student/viewall")
39 public String viewAll(Model model) {
40     List<StudentModel> students = studentService.selectAllStudents();
41     model.addAttribute("students", students);
42     return "viewall";
43 }
44
45 @RequestMapping(value = { "/student/view", "student/view/{npm}" })
46 public String viewStudent(@PathVariable Optional<String> npm, Model model) {
47     if (npm.isPresent()) {
48         StudentModel student = studentService.selectStudent(npm.get());
49         if (student.getNpm() == null) {
50             return "studentnotfound";
51         }
52         else {
53             model.addAttribute("student", student);
54         }
55     } else {
56         return "npmnotfound";
57     }
58     return "view";
59 }
60 }
```

Pada *method* ini digunakan request param dan path variable seperti yang telah diajarkan pada tutorial 2. Setelah berhasil mendapatkan npm input, dilakukan pencarian *student* dengan npm tersebut. Jika *student* dengan npm tersebut tidak ditemukan, maka dikeluarkan *error* bahwa data yang dicari terkait npm tersebut tidak ditemukan (*redirect to studentnotfound*). Namun, jika dari awal *user* tidak



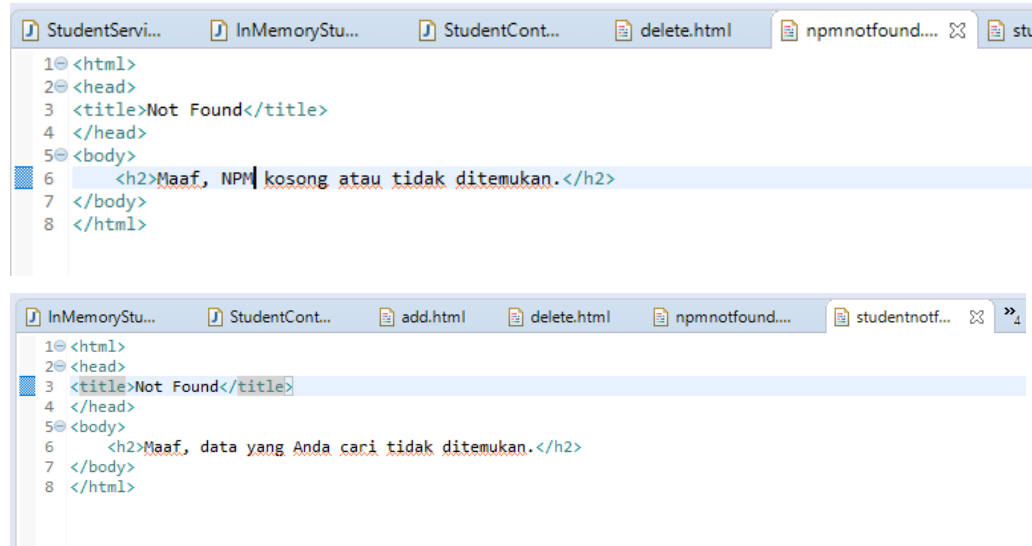
Atikah Luthfiana

1506689250

ADPAP B

melakukan *input* npm sama sekali, maka akan keluar *error* bahwa npm kosong atau tidak ditemukan (*redirect to npmnotfound*).

- Selanjutnya membuat halaman html untuk *view*, *npmnotfound*, dan *studentnotfound*. Karena pada saat tutorial telah tersedia *view.html*, saya menggunakan halaman yang sama untuk *viewStudent* ini. Sehingga sekarang saya hanya akan membuat html untuk *npmnotfound*, dan *studentnotfound*.



The image contains two screenshots of a code editor. The top screenshot shows a file named 'npmnotfound....' with the following HTML code:

```
1 <html>
2 <head>
3   <title>Not Found</title>
4 </head>
5 <body>
6   <h2>Maaf, NPM kosong atau tidak ditemukan.</h2>
7 </body>
8 </html>
```

The bottom screenshot shows a file named 'studentnotf...' with the following HTML code:

```
1 <html>
2 <head>
3   <title>Not Found</title>
4 </head>
5 <body>
6   <h2>Maaf, data yang Anda cari tidak ditemukan.</h2>
7 </body>
8 </html>
```

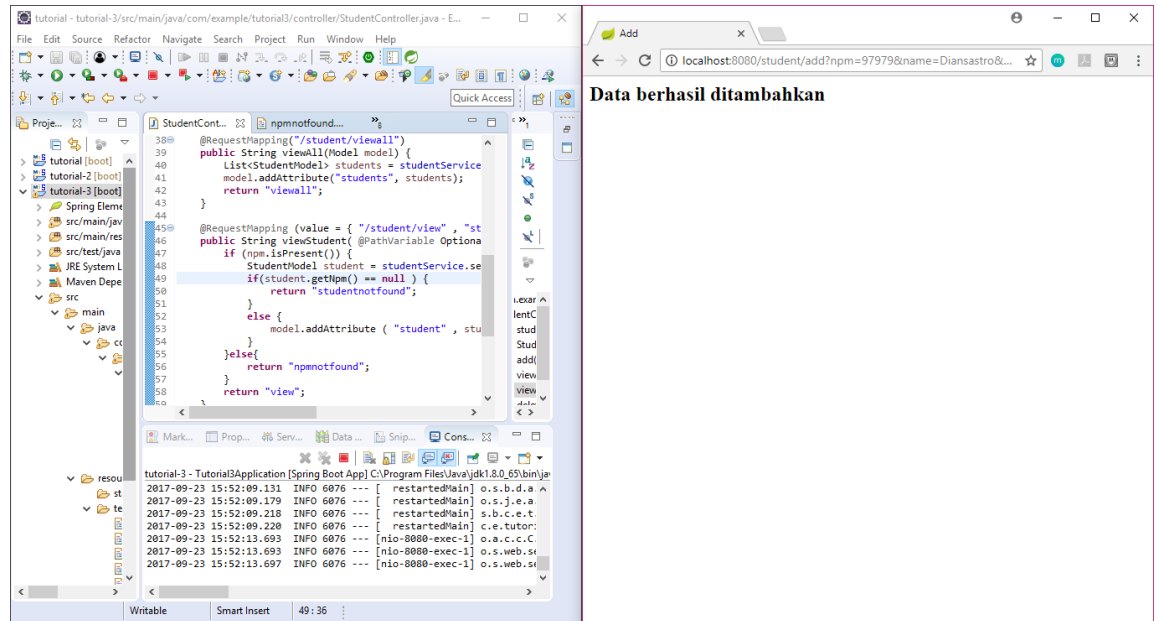
Selanjutnya dijalankan

- Menambahkan student baru

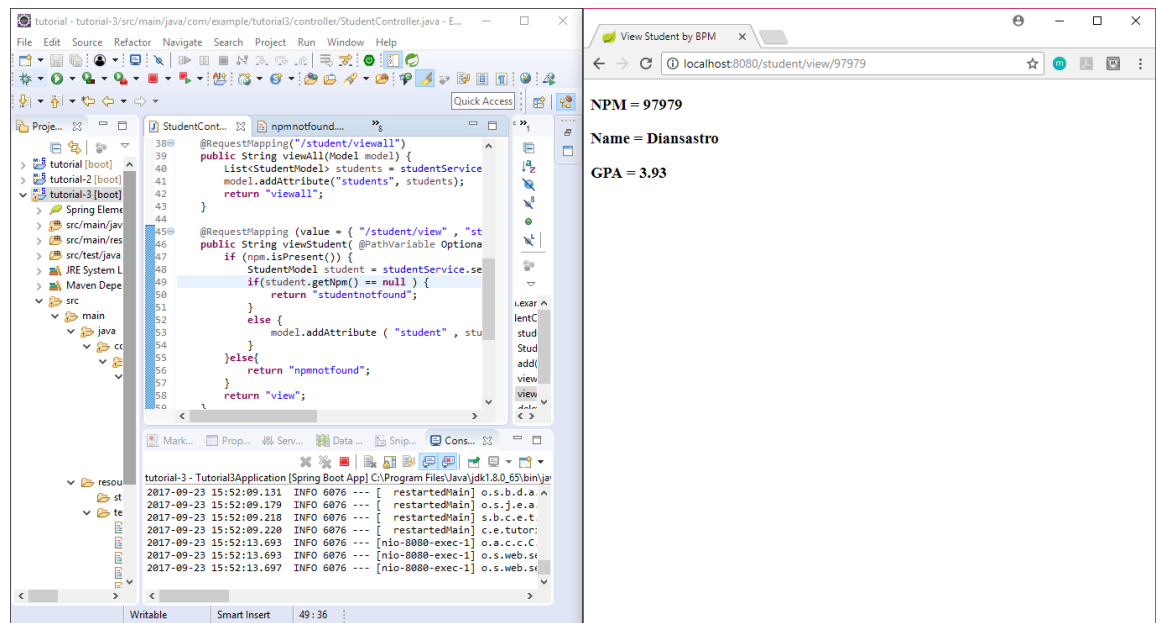
Atikah Luthfiana

1506689250

ADPAP B

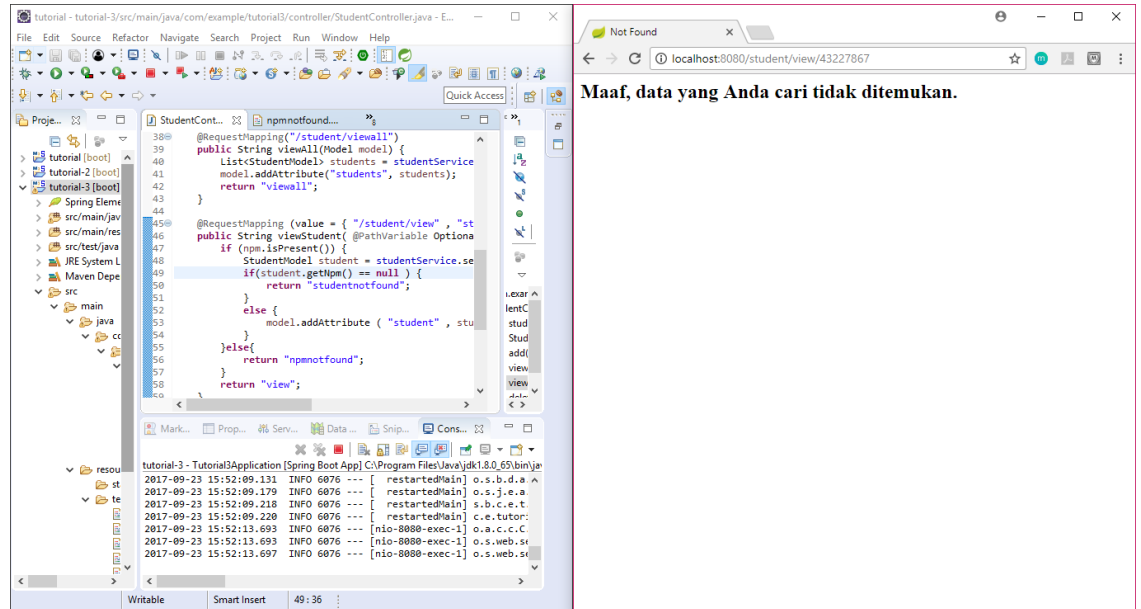


- Melihat student yang baru ditambahkan

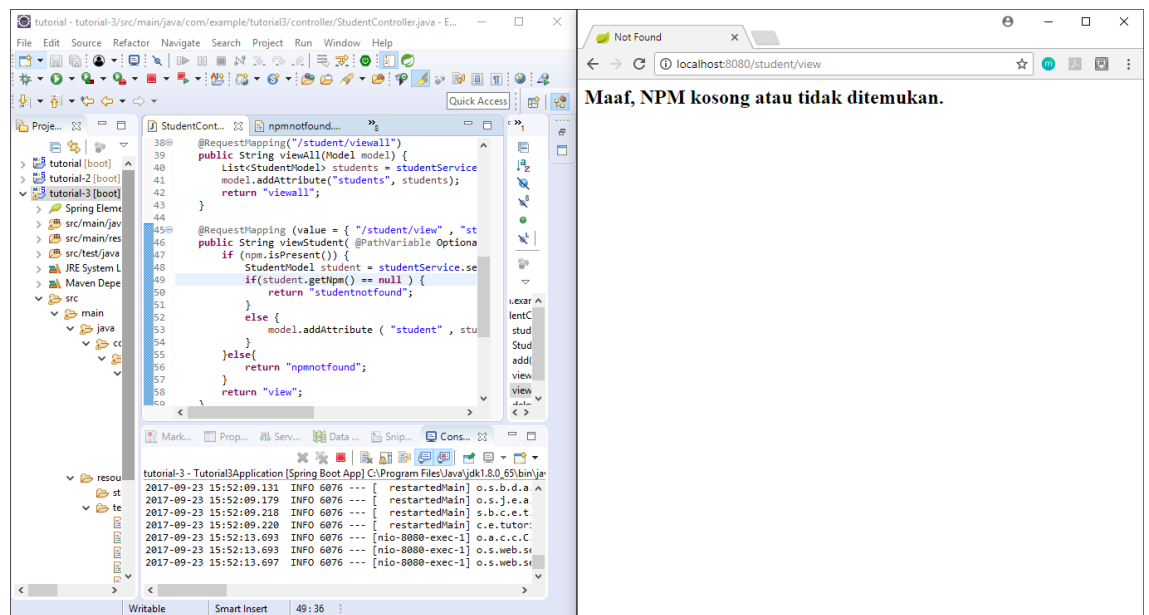


Atikah Luthfiana  
1506689250  
ADPAP B

- Melihat hasil dari npm yang tidak ada



- Melihat hasil tanpa memasukkan npm



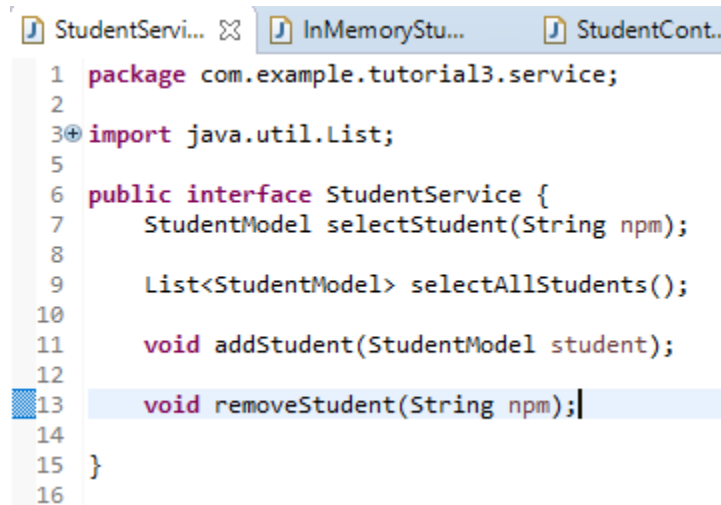
Atikah Luthfiana

1506689250

ADPAP B

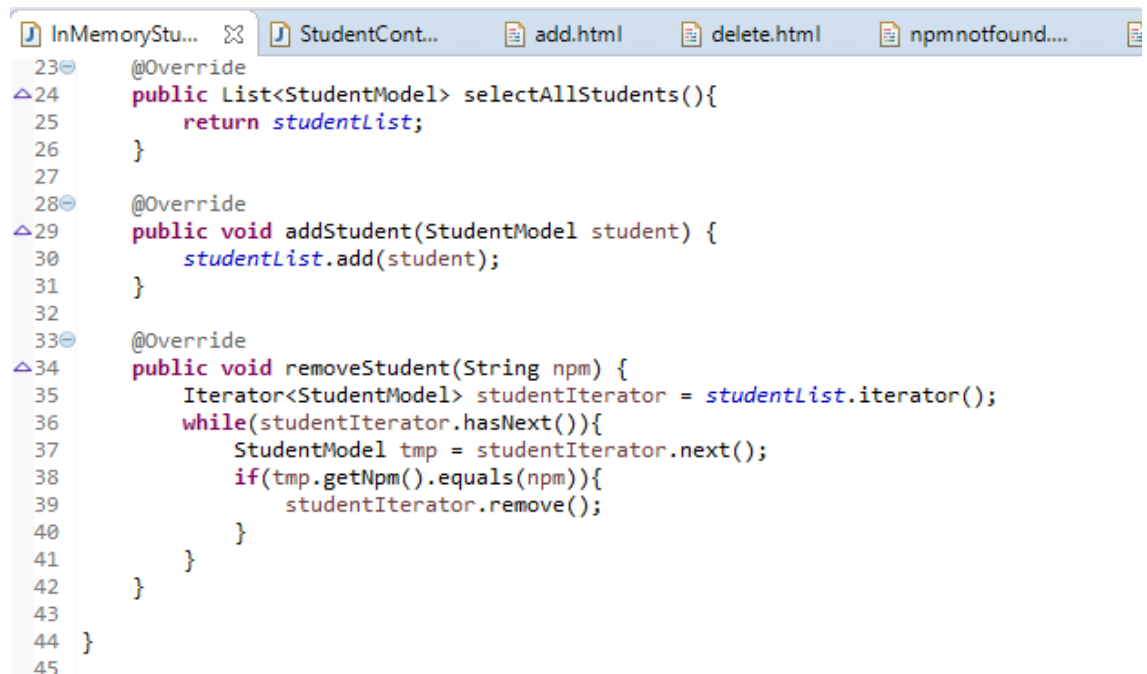
## 2. Delete student

- Membuat abstract method `removeStudent` pada *interface* `studentService`



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

- Mengimplementasikan method `removeStudent` pada `InMemoryStudentService`



```
23 @Override
24 public List<StudentModel> selectAllStudents(){
25     return studentList;
26 }
27
28 @Override
29 public void addStudent(StudentModel student) {
30     studentList.add(student);
31 }
32
33 @Override
34 public void removeStudent(String npm) {
35     Iterator<StudentModel> studentIterator = studentList.iterator();
36     while(studentIterator.hasNext()){
37         StudentModel tmp = studentIterator.next();
38         if(tmp.getNpm().equals(npm)){
39             studentIterator.remove();
40         }
41     }
42 }
43
44 }
45
```

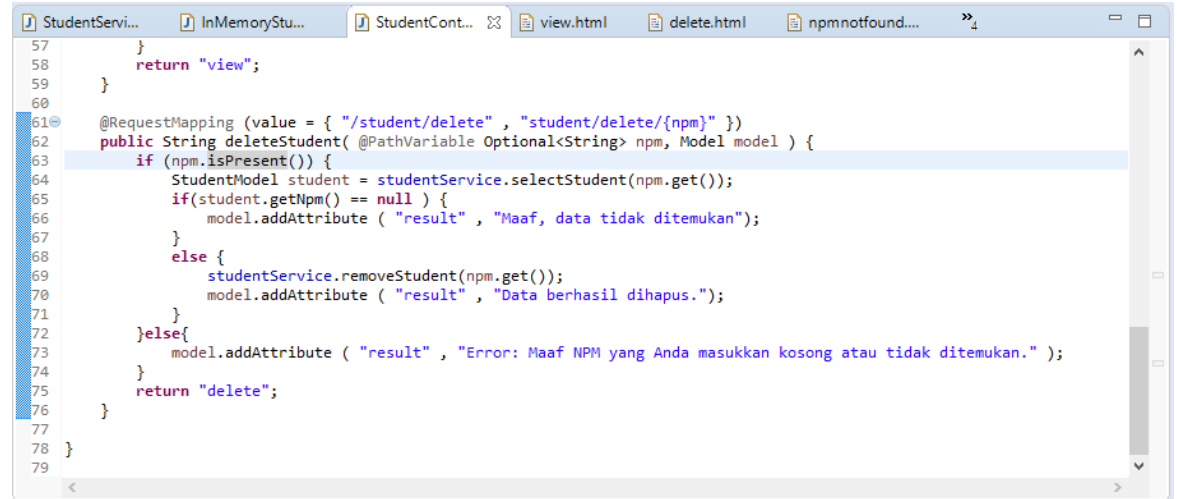
Method ini memerlukan import `iterator`. `Iterator` disini digunakan untuk mencari *student* berdasarkan `npm` yang dimasukkan. Ketika `iterator` menemukan *student* yang dimaksud, kemudian dilakukan penghapusan *student*.

Atikah Luthfiana

1506689250

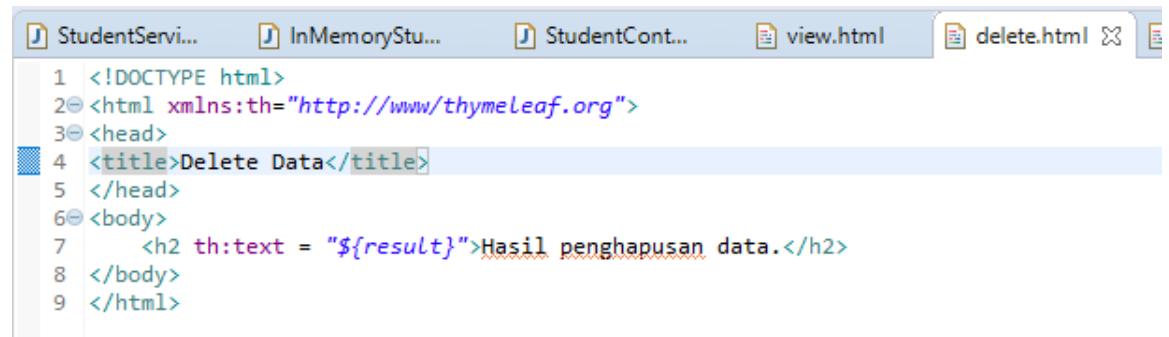
ADPAP B

- Membuat RequestMapping pada class StudentController



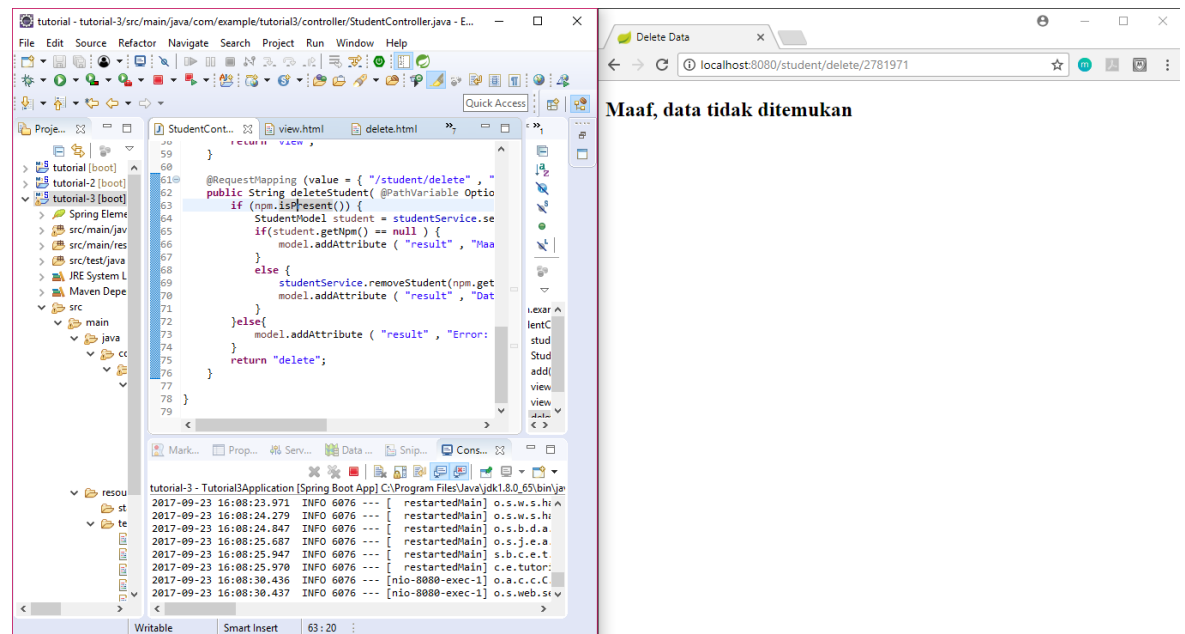
```
57     }
58     return "view";
59 }
60
61 @RequestMapping (value = { "/student/delete", "student/delete/{npm}" })
62 public String deleteStudent( @PathVariable Optional<String> npm, Model model ) {
63     if (npm.isPresent()) {
64         StudentModel student = studentService.selectStudent(npm.get());
65         if(student.getNpm() == null ) {
66             model.addAttribute ( "result" , "Maaf, data tidak ditemukan");
67         }
68         else {
69             studentService.removeStudent(npm.get());
70             model.addAttribute ( "result" , "Data berhasil dihapus.");
71         }
72     }else{
73         model.addAttribute ( "result" , "Error: Maaf NPM yang Anda masukkan kosong atau tidak ditemukan." );
74     }
75     return "delete";
76 }
77 }
78
79 }
```

- Membuat html untuk menampilkan proses delete berhasil



```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>Delete Data</title>
5 </head>
6 <body>
7 <h2 th:text = "${result}">Hasil penghapusan data.</h2>
8 </body>
9 </html>
```

- NPM diberikan tetapi tidak ada dalam List

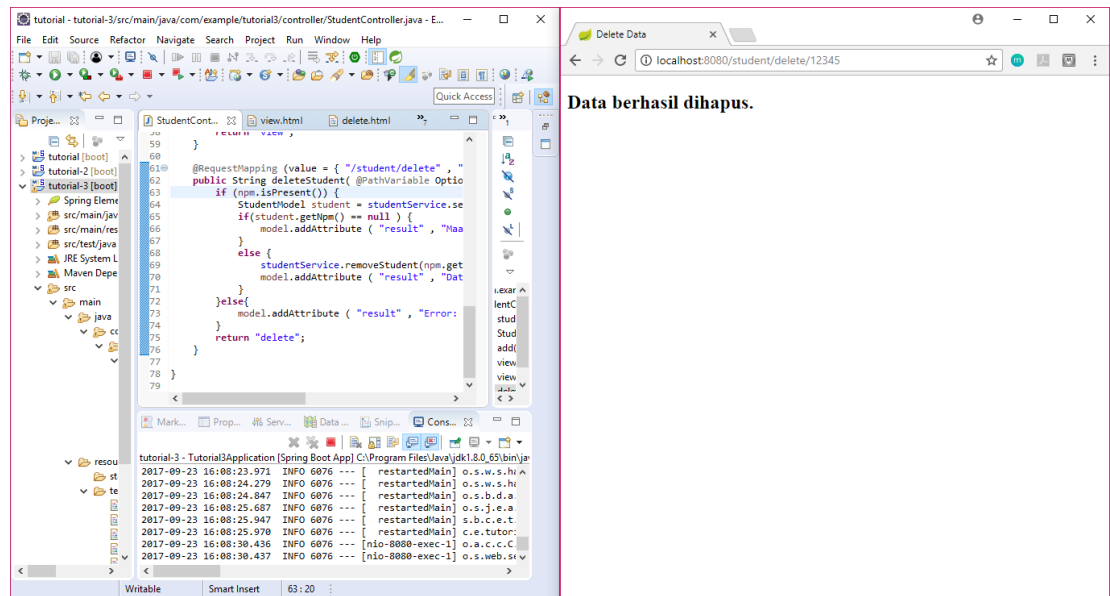
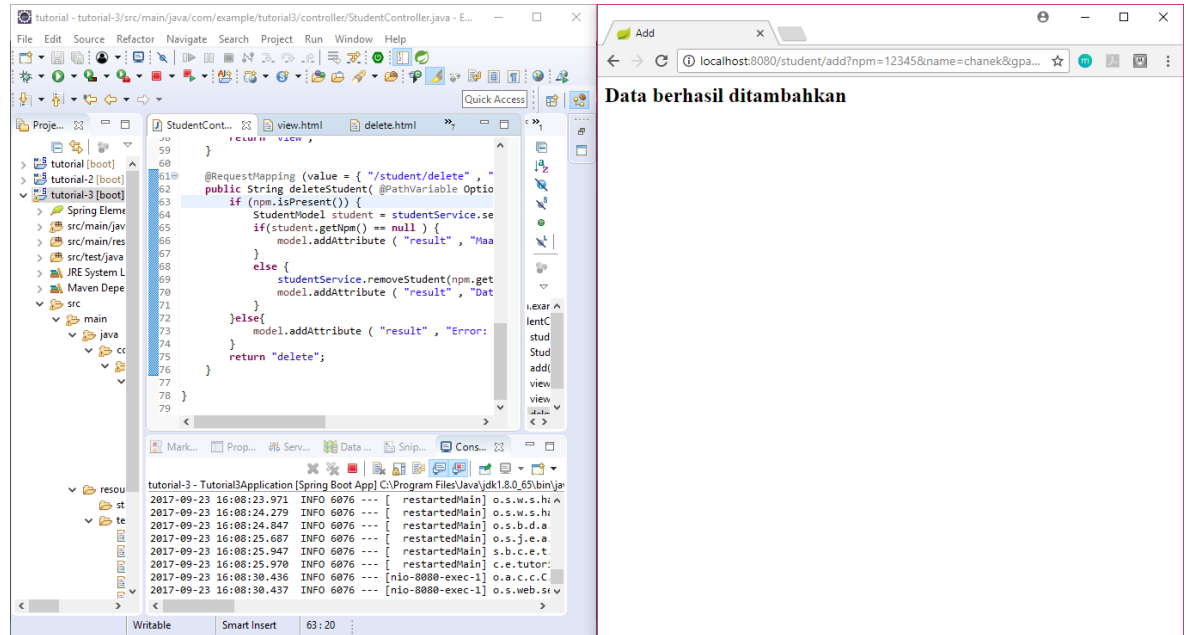


Atikah Luthfiana

1506689250

ADPAP B

- NPM ada dan berhasil dihapus

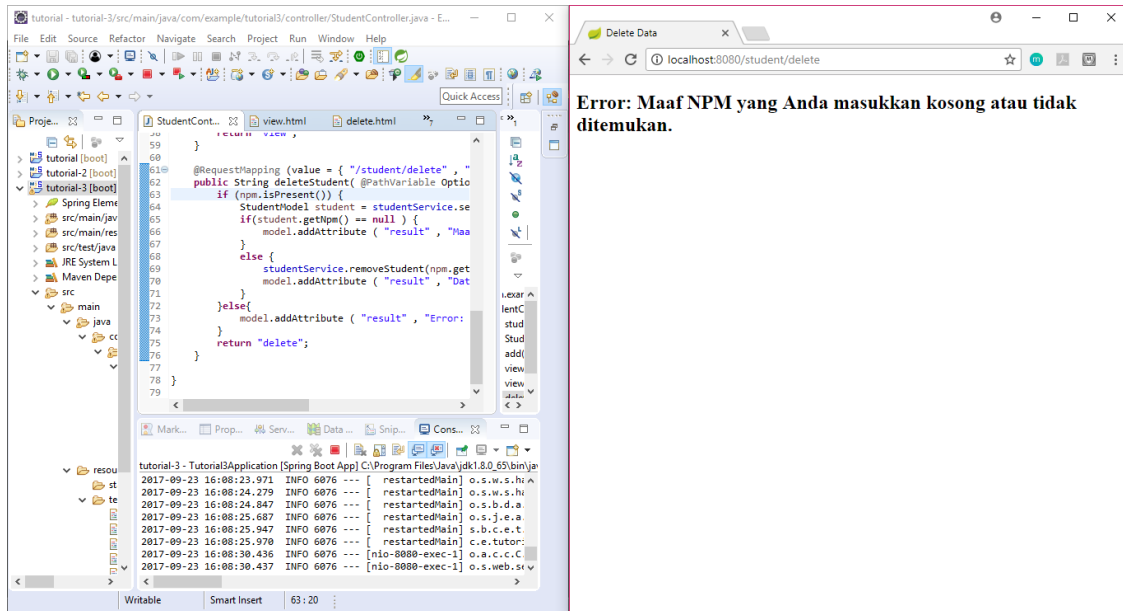


- NPM tidak diberikan user

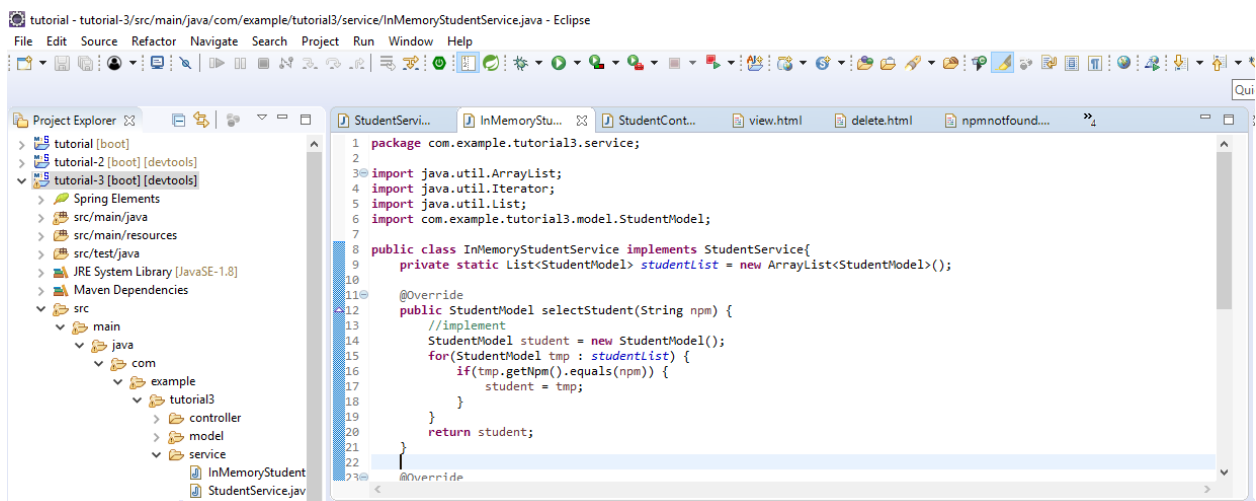
Atikah Luthfiana

1506689250

ADPAP B



### c. Implementasi method selectStudent



Source code:

```
public StudentModel selectStudent(String npm) {  
    //implement  
    StudentModel student = new StudentModel();  
    for(StudentModel tmp : studentList) {  
        if(tmp.getNpm().equals(npm)) {  
            student = tmp;  
        }  
    }  
    return student;  
}
```

d. Penjelasan fitur delete telah saya jelaskan pada bagian b di atas (b → Latihan → Delete Student)