

Tutorial 3

Menggunakan Model dan Service pada Project Spring Boot

1. Ringkasan materi

Pada tutorial ini, dibahas mengenai *model* dan *service* pada *project* Spring Boot serta kaitannya dengan *view* dan *controller* yang sudah dibahas di tutorial 2. *Model* merupakan sebuah objek yang merepresentasikan suatu hal, misalnya dalam tutorial ini adalah *model student* yang merepresentasikan objek mahasiswa yang memiliki atribut nama, npm, dan gpa. *Model* tersebut kemudian dapat dilakukan manipulasi dengan *service layer*.

Service adalah *layer* yang menjadi perantara antara *controller* dan *database*. Pada *service layer* tersebut, kita dapat melakukan pengolahan data yang disimpan di *database*, misalnya pada tutorial ini, kita memiliki *service* *StudentService* yang merupakan *interface* yang mendefinisikan *method-method* apa saja yang ada untuk memanipulasi kelas *Student*. Kita juga memiliki *service* *InMemoryStudentService* yang mengimplementasikan *interface* *StudentService* dan melakukan *override method-method* yang ada di *StudentService* sehingga dapat dilakukan pengolahan data seperti menambah data mahasiswa dengan mengimplementasikan *method add* atau men-select data mahasiswa dengan mengimplementasikan *method select*. *Method-method* tersebut kemudian dapat dipanggil di kelas *controller* jika diperlukan pengolahan data terlebih dahulu sebelum ditampilkan di *view*.

2. Hasil jawaban dari setiap poin pada bagian tutorial

a. Membuat Controller dan Fungsi Add

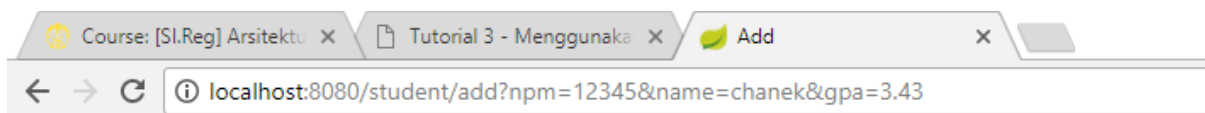
Jalankan program dan buka:

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

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

Jawab: Ditampilkan *page add* yang memberikan informasi bahwa data berhasil ditambahkan.

Screenshot:



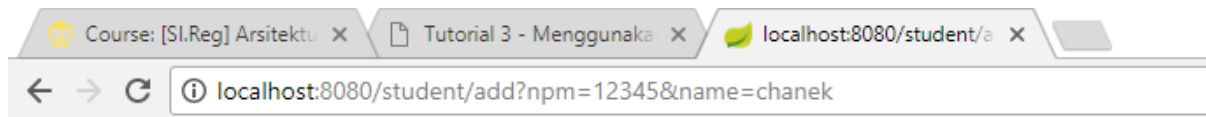
Data berhasil ditambahkan

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

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

Jawab: Terjadi *error bad request* karena tidak terdapat parameter gpa pada url (parameter gpa didefinisikan sebagai *required* parameter) yang harus diambil oleh *method* yang menangani RequestMapping /student/add

Screenshot:



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Sep 20 23:50:33 ICT 2017

There was an unexpected error (type=Bad Request, status=400).

Required double parameter 'gpa' is not present

b. Method View by NPM

Jalankan program dan buka:

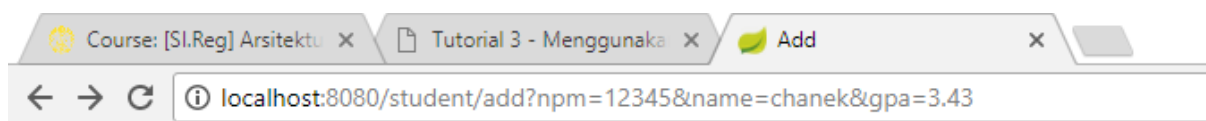
localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka

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

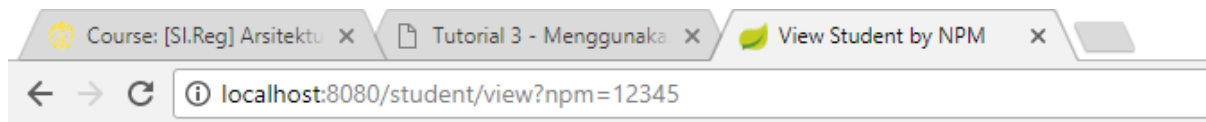
Pertanyaan 3: apakah data Student tersebut muncul? Jika tidak, mengapa?

Jawab: Data student tersebut dapat dimunculkan.

Screenshot:



Data berhasil ditambahkan



NPM = 12345

Name = chanek

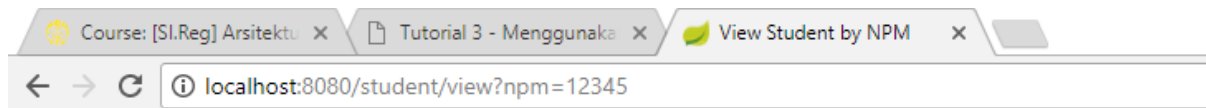
GPA = 3.43

Coba matikan program dan jalankan kembali serta buka

localhost:8080/student/view?npm=12345

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

Jawab: Data Student tidak dapat dimunculkan. Terjadi *error internal server* karena template `view.html` yang mengeksekusi perintah `student.npm`, `student.name`, `student.gpa` tidak mendapatkan *object student* untuk dilihat data-datanya. Karena ketika program di-run kembali maka List yang menampung data *students* yang sebelumnya sudah di-add jadi kembali kosong.



Whitelabel Error Page

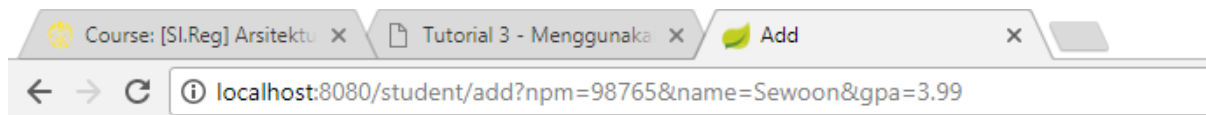
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Thu Sep 21 01:06:55 ICT 2017

There was an unexpected error (type=Internal Server Error, status=500).

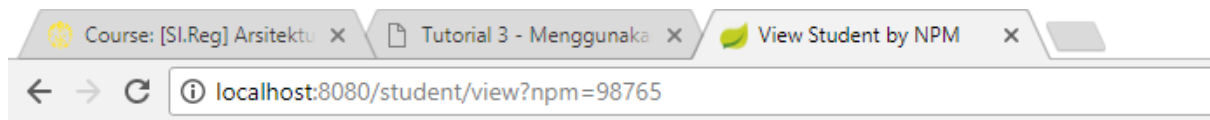
Exception evaluating SpringEL expression: "student.npm" (view:7)

Coba tambahkan data Student lainnya dengan NPM yang berbeda.



Data berhasil ditambahkan

Ricca Fitriani
1506689616
APAP – B



NPM = 98765

Name = Sewoon

GPA = 3.99

c. Method View All

Jalankan program dan buka

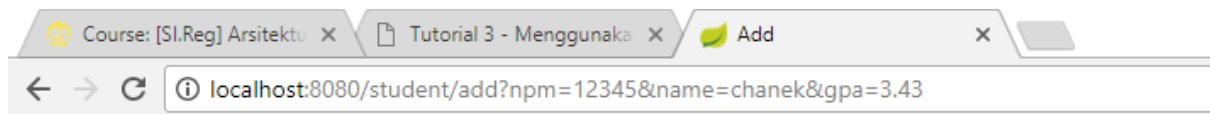
localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka

localhost:8080/student/viewall,

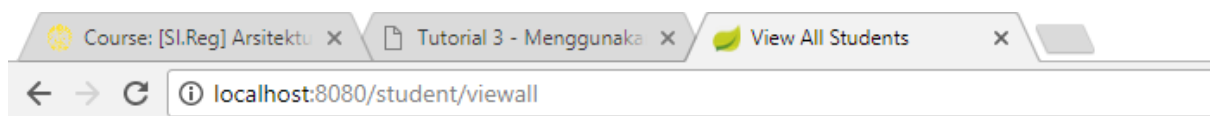
Pertanyaan 5: apakah data Student tersebut muncul?

Jawab: Ya, data Student dapat dimunculkan.

Screenshot:



Data berhasil ditambahkan



No. 1

NPM = 12345

Name = chanek

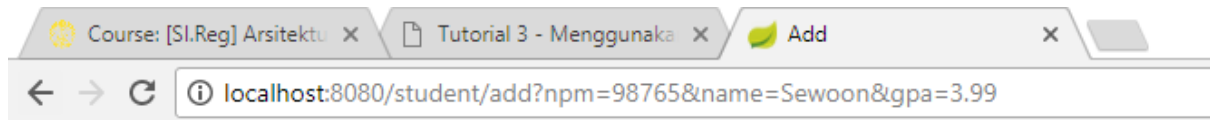
GPA = 3.43

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

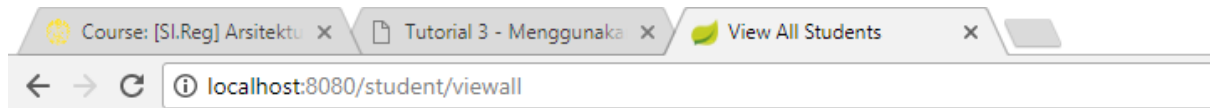
Pertanyaan 6: Apakah semua data Student muncul?

Jawab: Ya, semua data Student dapat dimunculkan.

Screenshot:



Data berhasil ditambahkan



No. 1

NPM = 12345

Name = chanek

GPA = 3.43

No. 2

NPM = 98765

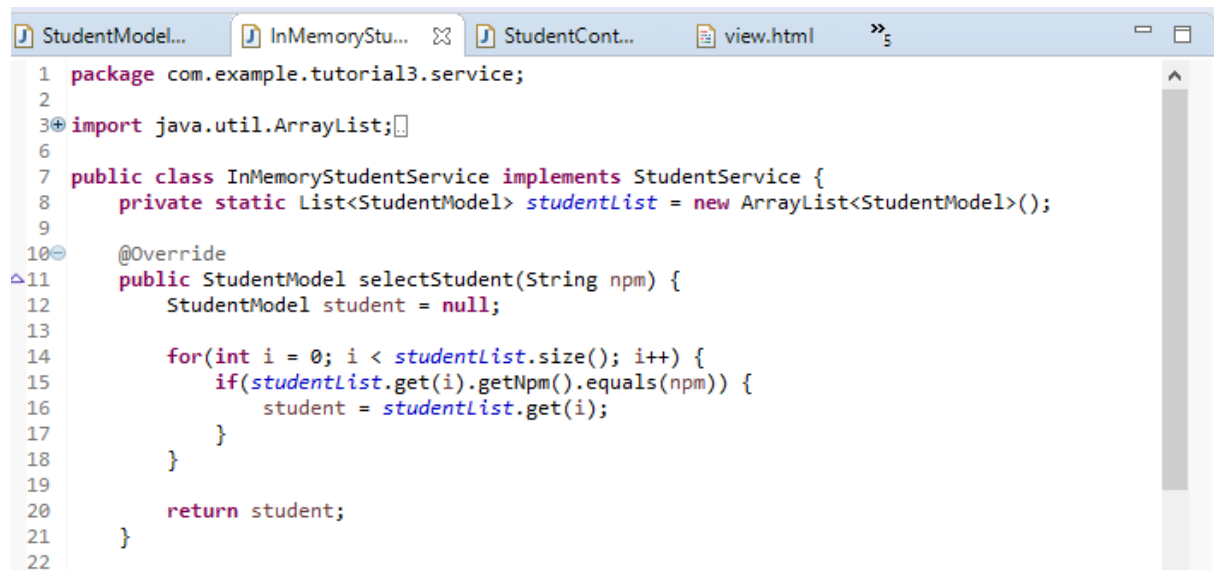
Name = Sewoon

GPA = 3.99

3. Method selectStudent

Method selectStudent tersebut akan melakukan *looping* List yang menampung objek student yang sudah ditambahkan, kemudian *method* akan mencari objek *student* berdasarkan NPM yang menjadi parameter selectStudent tersebut dan mengembalikan objek *student* tersebut jika ada data yang sesuai dengan npm yang diberikan.

Screenshot:



```
1 package com.example.tutorial3.service;
2
3 import java.util.ArrayList;
4
5
6
7 public class InMemoryStudentService implements StudentService {
8     private static List<StudentModel> studentList = new ArrayList<StudentModel>();
9
10    @Override
11    public StudentModel selectStudent(String npm) {
12        StudentModel student = null;
13
14        for(int i = 0; i < studentList.size(); i++) {
15            if(studentList.get(i).getNpm().equals(npm)) {
16                student = studentList.get(i);
17            }
18        }
19
20        return student;
21    }
22 }
```

4. Penjelasan soal latihan (fitur view dan delete)

a. Method view

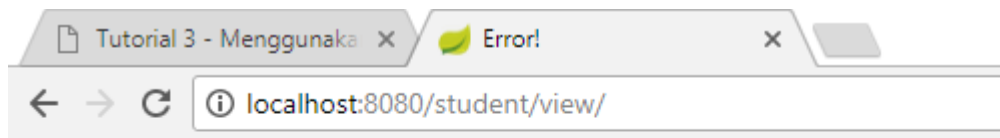
Sebelum menambahkan *method view*, saya menghapus/meng-komen *method view* yang lama (yang dibuat ketika melakukan tutorial di atas) agar sistem tidak rancu (terjadi *error* karena tidak dapat menentukan *method* mana yang menangani requestmapping /student/view).

Kemudian saya menambahkan *method view* pada StudentController dengan RequestMapping student/view dan student/view/{npm}. student/view berfungsi untuk meng-handle akses dengan url yang tidak menuliskan *path variable* npm.

```
StudentModel... InMemoryStu... StudentCont... view.html
43 @RequestMapping(value = {"/student/view", "/student/view/{npm}"})
44 public String view(@PathVariable Optional<String> npm, Model model) {
45     String hasil = "";
46     if (npm.isPresent()) {
47         StudentModel student = studentService.selectStudent(npm.get());
48         if (student == null) {
49             hasil = "nullerror";
50         } else {
51             model.addAttribute("student", student);
52             hasil = "view";
53         }
54     } else {
55         hasil = "unknownerror";
56     }
57     return hasil;
58 }
59
```

Kemudian pada *method* tersebut akan dicek apakah pada url setelah /view diikuti *path variable* lain berupa angka (npm). Jika tidak, maka sistem akan mengembalikan string unknownerror, yang mana merupakan halaman *template* yang menyatakan bahwa npm kosong.

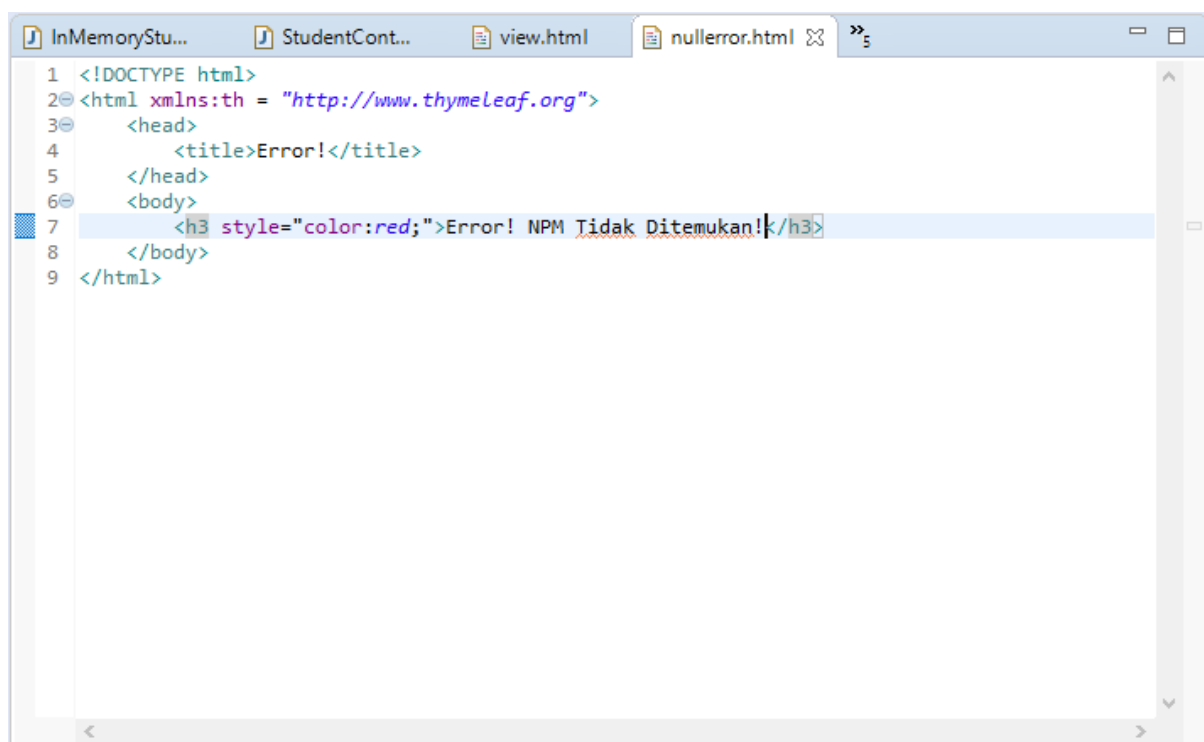
```
InMemoryStu... StudentCont... nullerror.html unknownerro...
1 <!DOCTYPE html>
2 <html xmlns:th = "http://www.thymeleaf.org">
3 <head>
4 <title>Error!</title>
5 </head>
6 <body>
7 <h3 style="color:red;">Error! NPM Kosong!</h3>
8 </body>
9 </html>
```

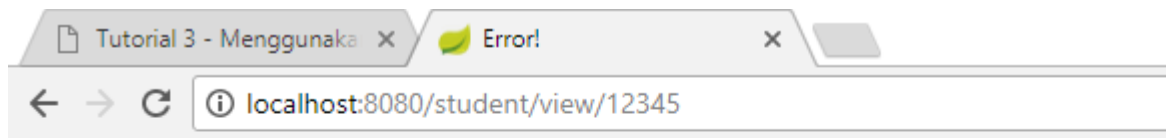


Error! NPM Kosong!

Jika url `/view` diikuti *path variable* yang berupa angka untuk mencari npm, sistem akan mengecek kembali apakah objek student sudah ada di list atau belum dengan memanggil *method* `selectStudent` pada `studentService`. Untuk dapat memanggil *method* `selectStudent` tersebut, sebelumnya harus dibuat objek `InMemoryStudentService` pada *constructor* `StudentController`.

Jika pada list yang menampung objek student tidak ditemukan npm yang dicari, maka *method* `view` akan mengembalikan string `nullerror` yang mana merupakan halaman *template* yang memberikan informasi bahwa npm tidak ditemukan.



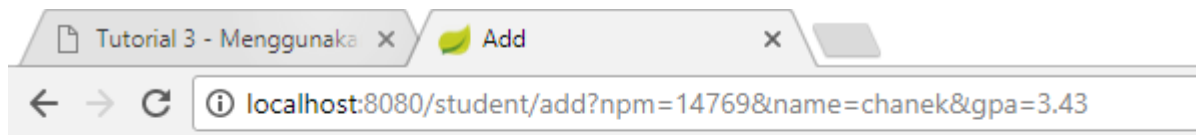


Error! NPM Tidak Ditemukan!

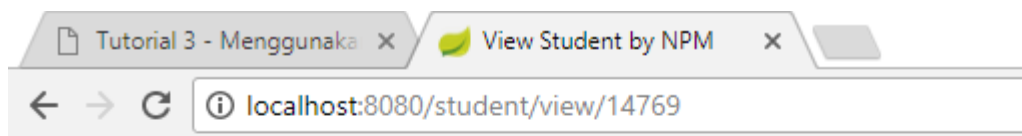
Jika objek student ditemukan, maka sistem akan mengembalikan string view dan *passing* objek student tersebut ke halaman *template* view agar dapat ditampilkan data-data yang bersangkutan.

A screenshot of a code editor with several tabs: 'StudentModel...', 'InMemoryStu...', 'StudentCont...', and 'view.html'. The 'view.html' tab is active, showing the following HTML code:

```
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 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
8     <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
9     <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
10  </body>
11 </html>
```



Data berhasil ditambahkan



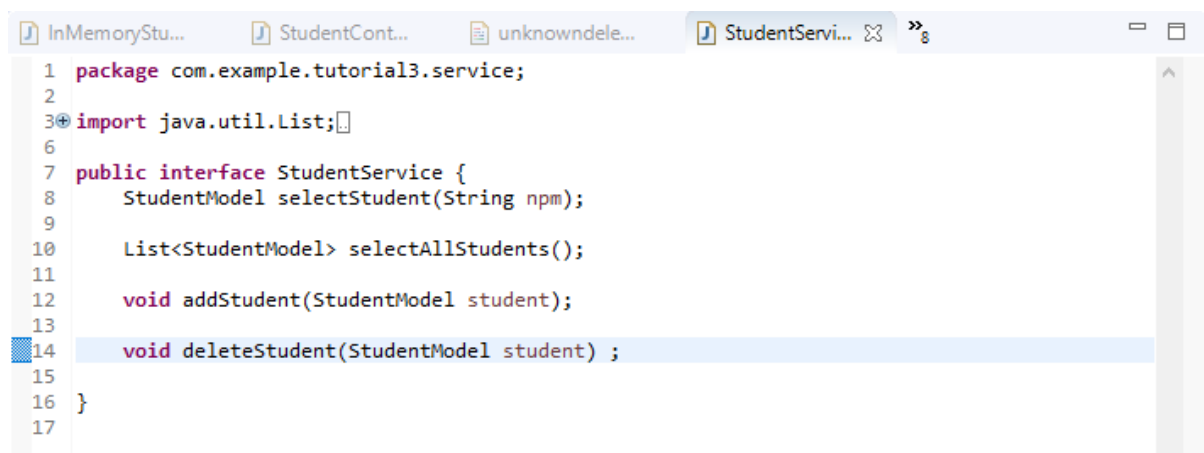
NPM = 14769

Name = chanek

GPA = 3.43

b. Method delete

Pertama, dibuat *method* deleteStudent dengan paramater objek student pada interface StudentService.

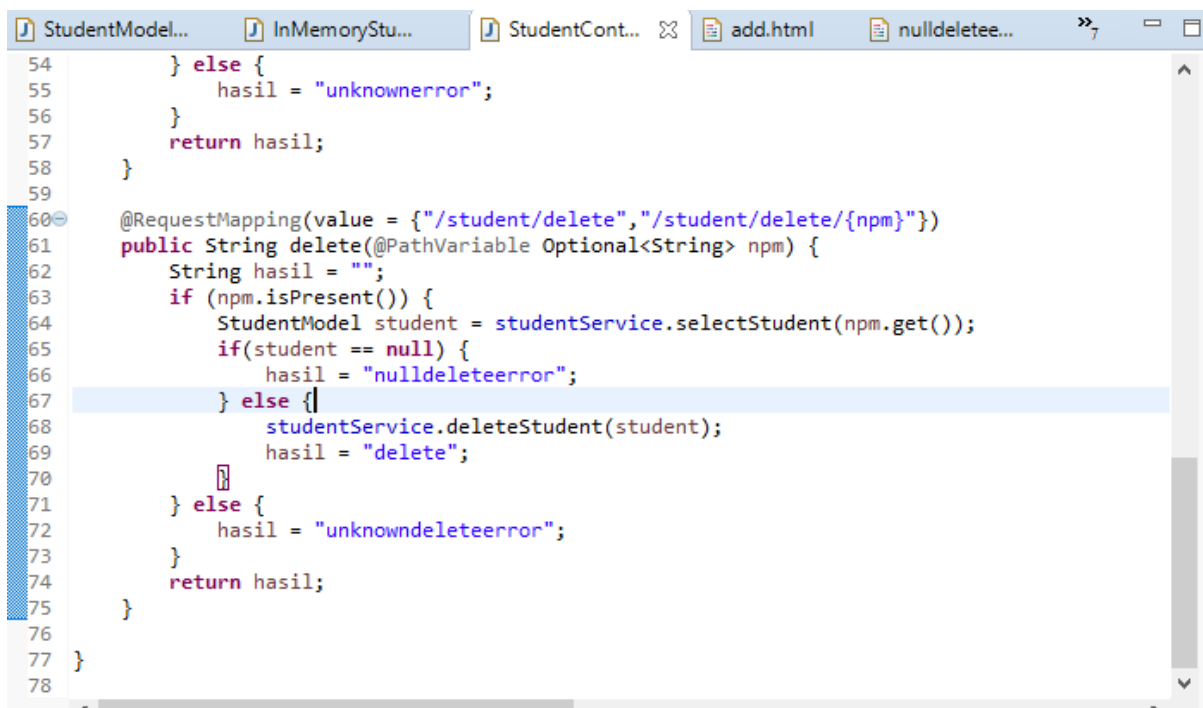


Kemudian, *method* `deleteStudent` tersebut di-override di kelas `InMemoryStudentService` untuk menghapus data student yang diinginkan pada list student.



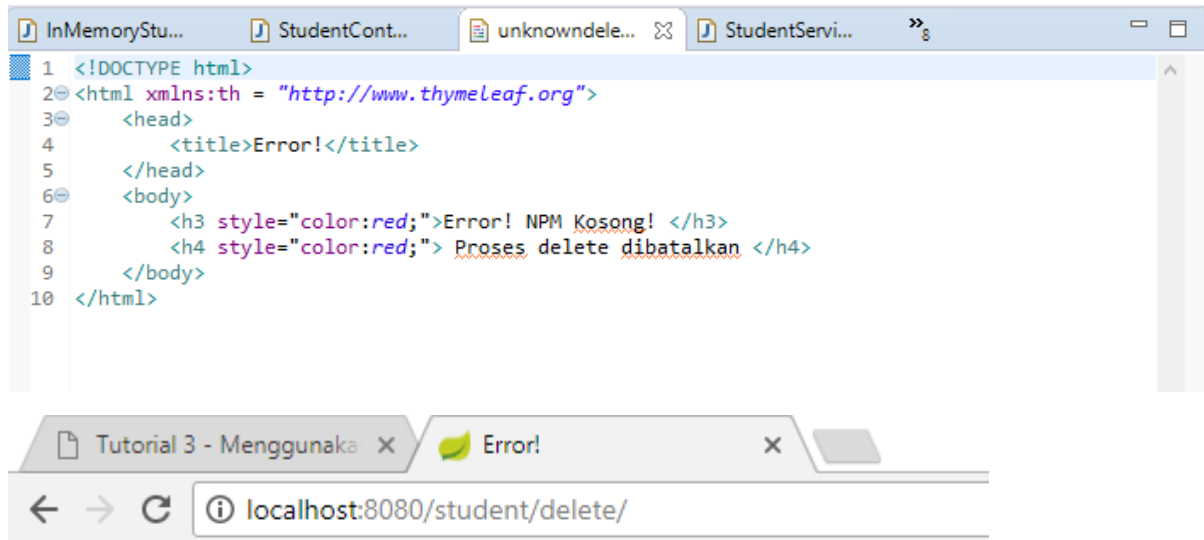
```
14     for(int i = 0; i < studentList.size(); i++) {  
15         if(studentList.get(i).getNpm().equals(npm)) {  
16             student = studentList.get(i);  
17         }  
18     }  
19  
20     return student;  
21 }  
22  
23 @Override  
24 public List<StudentModel> selectAllStudents() {  
25     return studentList;  
26 }  
27  
28 @Override  
29 public void addStudent(StudentModel student) {  
30     studentList.add(student);  
31 }  
32  
33 @Override  
34 public void deleteStudent(StudentModel student) {  
35     studentList.remove(student);  
36 }  
37 }  
38
```

Kemudian, pada kelas `StudentController` ditambahkan *method* `delete` dengan `RequestMapping` `student/delete/{npm}` dan `student/delete`. `student/delete` berfungsi untuk meng-handle url yang tidak menuliskan *path variable* npm.



```
54     } else {  
55         hasil = "unknownerror";  
56     }  
57     return hasil;  
58 }  
59  
60 @RequestMapping(value = {"/student/delete", "/student/delete/{npm}"})  
61 public String delete(@PathVariable Optional<String> npm) {  
62     String hasil = "";  
63     if (npm.isPresent()) {  
64         StudentModel student = studentService.selectStudent(npm.get());  
65         if(student == null) {  
66             hasil = "nulldeleteerror";  
67         } else {  
68             studentService.deleteStudent(student);  
69             hasil = "delete";  
70         }  
71     } else {  
72         hasil = "unknowndeleteerror";  
73     }  
74     return hasil;  
75 }  
76  
77 }  
78
```

Pada *method* tersebut akan dicek apakah pada url setelah `/delete` diikuti *path variable* lain berupa angka (*npm*). Jika tidak, maka sistem akan mengembalikan string `unknowndeleteerror`, yang mana merupakan halaman *template* yang menyatakan bahwa *npm* kosong dan proses *delete* dibatalkan.



The image shows a screenshot of a development environment. The top part is an IDE window with a file explorer on the left showing files like `InMemoryStu...`, `StudentCont...`, `unknownnde...`, and `StudentServi...`. The main editor displays an HTML file with the following content:

```
1 <!DOCTYPE html>
2 <html xmlns:th = "http://www.thymeleaf.org">
3   <head>
4     <title>Error!</title>
5   </head>
6   <body>
7     <h3 style="color:red;">Error! NPM Kosong! </h3>
8     <h4 style="color:red;">Proses delete dibatalkan </h4>
9   </body>
10 </html>
```

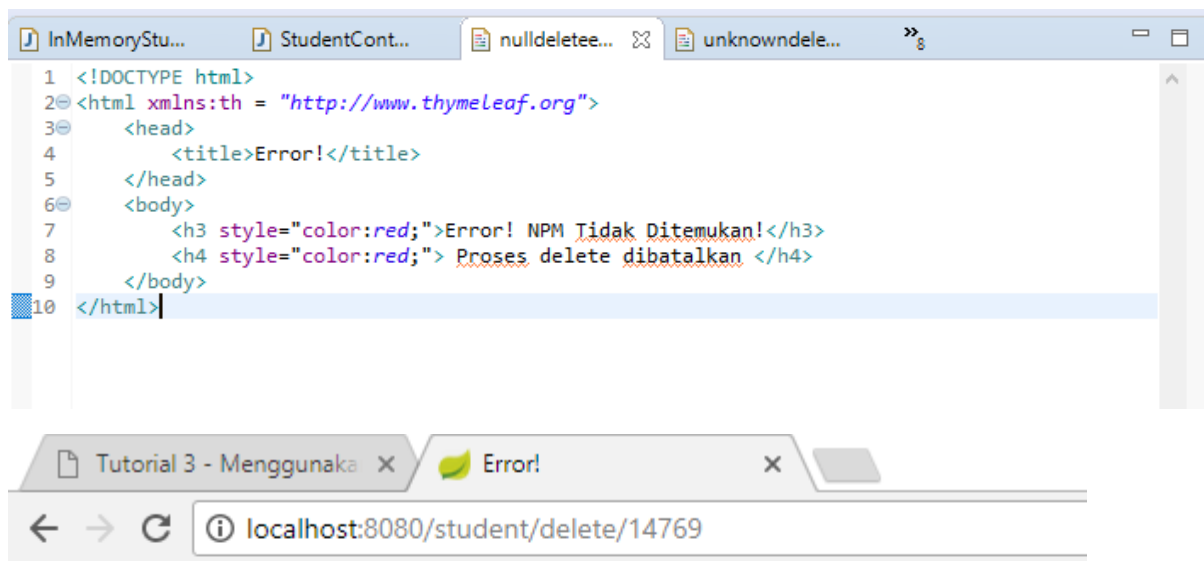
Below the IDE, a web browser window is open with the title "Tutorial 3 - Menggunakan x" and a sub-tab "Error!". The address bar shows the URL `localhost:8080/student/delete/`.

Error! NPM Kosong!

Proses delete dibatalkan

Jika url `/delete` diikuti *path variable* yang berupa angka untuk mencari *npm*, sistem akan mengecek kembali apakah objek *student* dengan *npm* tersebut sudah ada di list atau belum dengan memanggil *method* `selectStudent` pada `studentService`. Untuk dapat memanggil *method* `selectStudent` tersebut, sebelumnya harus dibuat objek `InMemoryStudentService` pada *constructor* `StudentController`.

Jika pada list yang menampung objek *student* tidak ditemukan *npm* yang dicari, maka *method* `delete` akan mengembalikan string `nulldeleteerror` yang mana merupakan halaman *template* yang memberikan informasi bahwa *npm* tidak ditemukan dan proses *delete* dibatalkan.

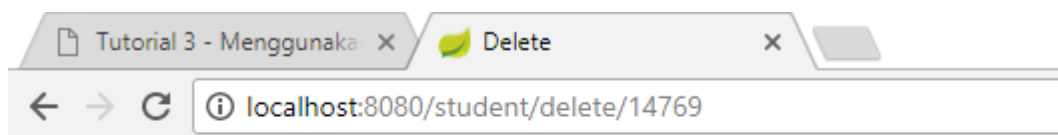


Error! NPM Tidak Ditemukan!

Proses delete dibatalkan

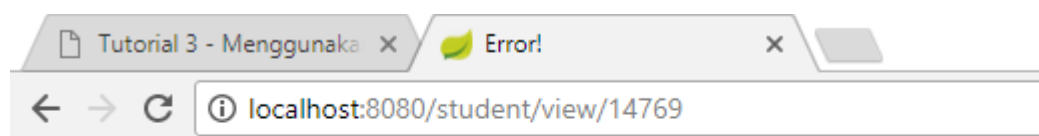
(Ket: screenshot di atas diambil setelah menghapus npm 14769 dari list, yang dijelaskan di bawah)

Jika objek student ditemukan, maka sistem akan mengembalikan string delete yang mana merupakan halaman *template* dengan informasi bahwa data berhasil dihapus dan kemudian memanggil *method* deleteStudent pada studentService yang sudah dibuat untuk menghapus objek student tersebut dari list.



Data berhasil dihapus

Jika student tersebut sudah dihapus, maka ketika dipanggil url view dengan npm student tersebut akan dikembalikan *page error* npm tidak ditemukan.



Error! NPM Tidak Ditemukan!