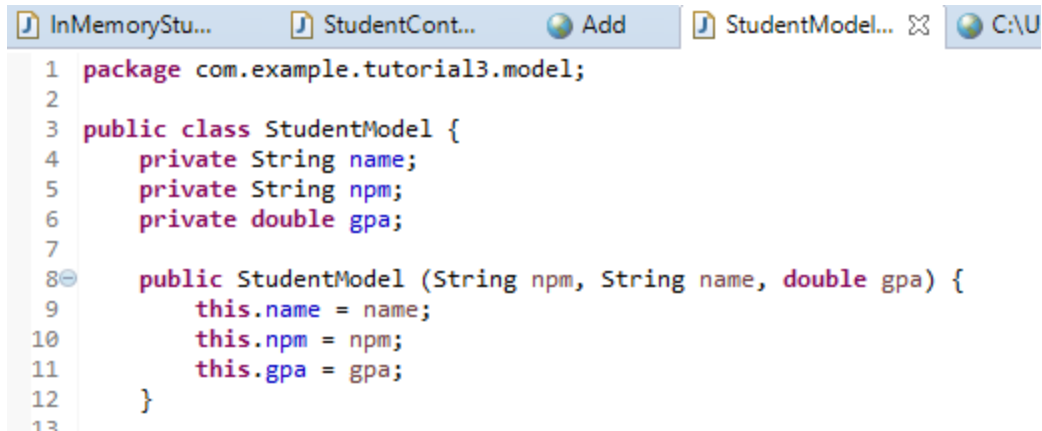


Membuat Class Model

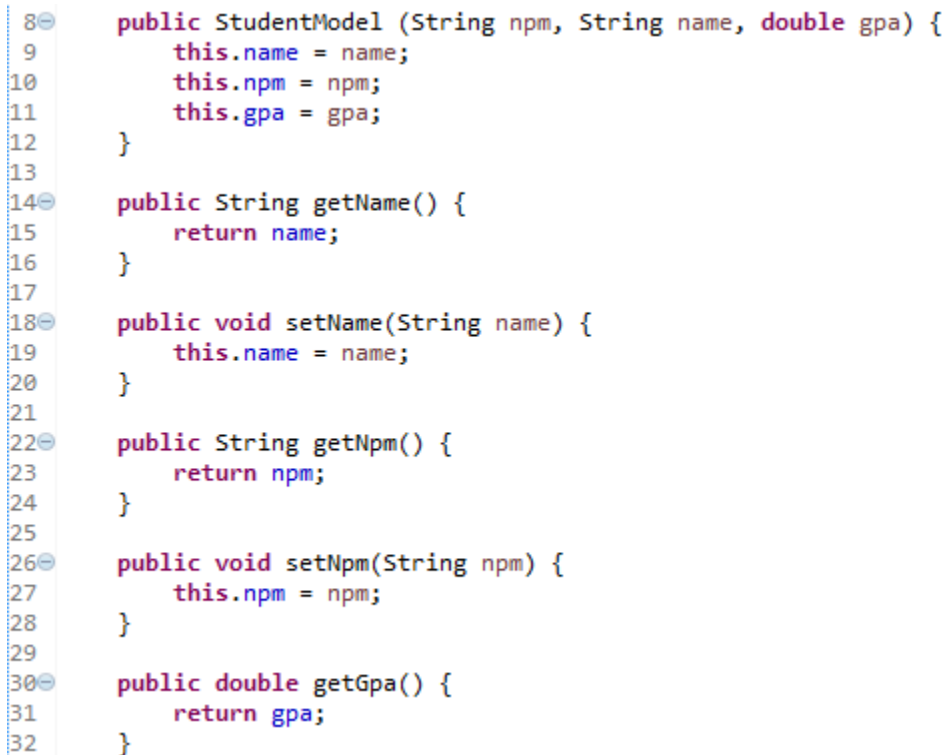
1. Klik kanan pada Project > New > Package. Buatlah package com.example.tutorial3.model
2. Buatlah class StudentModel dengan spesifikasi seperti di atas pada package tersebut



The screenshot shows an IDE with a project explorer on the left containing 'InMemoryStu...', 'StudentCont...', and 'StudentModel...'. The main editor displays the following Java code for the StudentModel class:

```
1 package com.example.tutorial3.model;
2
3 public class StudentModel {
4     private String name;
5     private String npm;
6     private double gpa;
7
8     public StudentModel (String npm, String name, double gpa) {
9         this.name = name;
10        this.npm = npm;
11        this.gpa = gpa;
12    }
13 }
```

3. Tambahkan method constructor, setter, dan getter.



The screenshot shows the same IDE with the StudentModel class now including getters and setters. The code is as follows:

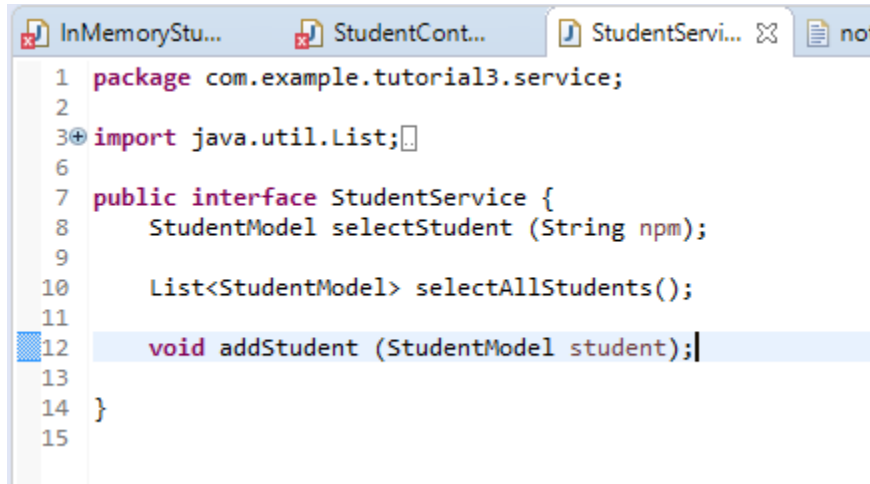
```
8     public StudentModel (String npm, String name, double gpa) {
9         this.name = name;
10        this.npm = npm;
11        this.gpa = gpa;
12    }
13
14    public String getName() {
15        return name;
16    }
17
18    public void setName(String name) {
19        this.name = name;
20    }
21
22    public String getNpm() {
23        return npm;
24    }
25
26    public void setNpm(String npm) {
27        this.npm = npm;
28    }
29
30    public double getGpa() {
31        return gpa;
32    }
33 }
```

Membuat Service

1. Klik kanan pada Project > New > Package. Buatlah package com.example.tutorial3.service

2. Buatlah interface StudentService.java pada package tersebut dengan isi sebagai berikut:

Method selectStudent:

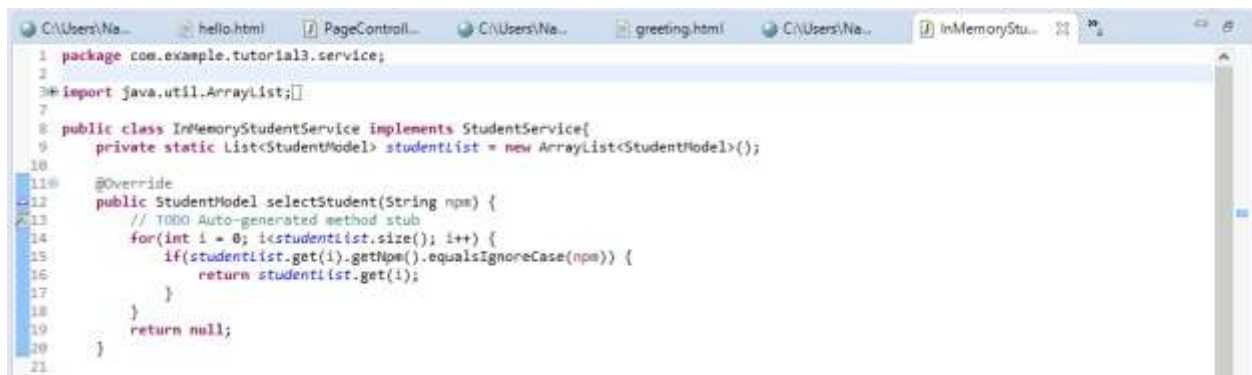


```
1 package com.example.tutorial3.service;
2
3 import java.util.List;
4
5
6
7 public interface StudentService {
8     StudentModel selectStudent (String npm);
9
10    List<StudentModel> selectAllStudents();
11
12    void addStudent (StudentModel student);
13
14 }
15
```

3. Pada package yang sama, buat class InMemoryStudentService yang meng-implements StudentService dengan isi sebagai berikut:

```
public class InMemoryStudentService implements StudentService{
    private static List<StudentModel> studentList = new ArrayList<StudentModel>();

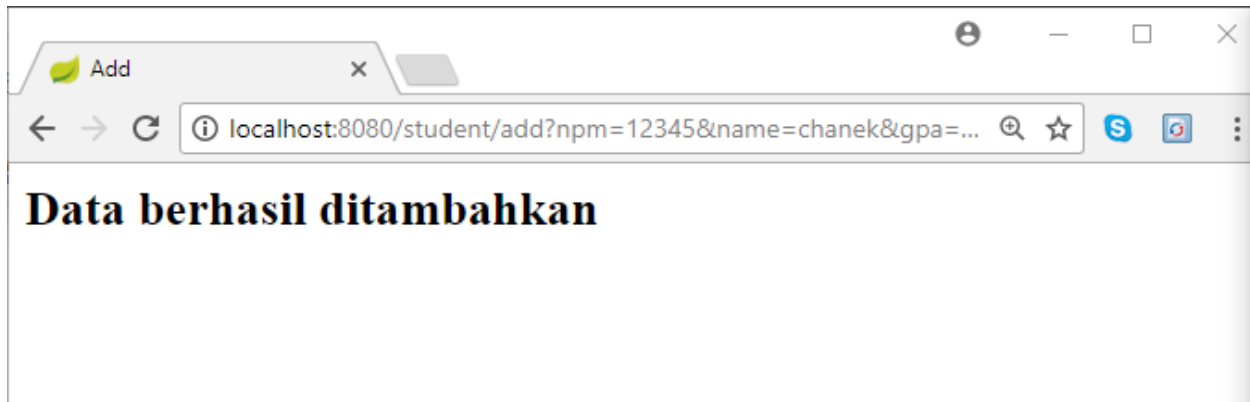
    @Override
    public StudentModel selectStudent(String npm) {
        // TODO Auto-generated method stub
        for(int i = 0; i<studentList.size(); i++) {
            if(studentList.get(i).getNpm().equalsIgnoreCase(npm)) {
                return studentList.get(i);
            }
        }
        return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



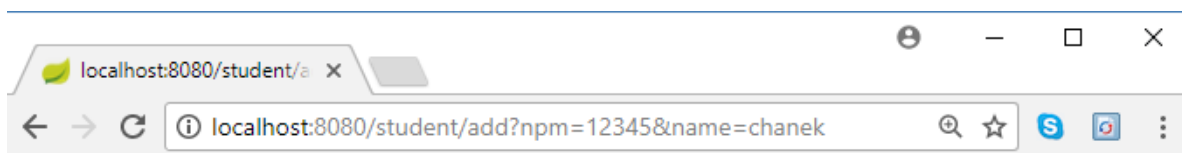
```
1 package com.example.tutorial3.service;
2
3 import java.util.ArrayList;
4
5
6
7 public class InMemoryStudentService implements StudentService{
8     private static List<StudentModel> studentList = new ArrayList<StudentModel>();
9
10
11     @Override
12     public StudentModel selectStudent(String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

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

Pertanyaan 1: apakah hasilnya?



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



Whitelabel Error Page

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

Fri Sep 22 22:15:02 ICT 2017

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

Required double parameter 'gpa' is not present

Pertanyaan 2: apakah hasilnya?

- ➔ Terdapat 'Whitelabel Error Page'. Hal ini terjadi karena tidak terdapat parameter "gpa", sedangkan parameter tersebut dibutuhkan.

Membuat Controller dan Fungsi Add

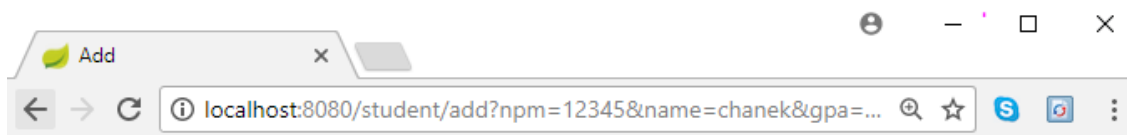
1. Pada class StudentController tambahkan method berikut:

```
28  
29 @RequestMapping("/student/view")  
30 public String view(Model model, @RequestParam(value = "npm", required = true) String npm) {  
31     StudentModel student = studentService.selectStudent(npm);  
32     model.addAttribute("student", student);  
33     return "view";  
34  
35 }
```

2. Selanjutnya pada directory resources/templates tambahkan view.html dengan isi sebagai berikut:

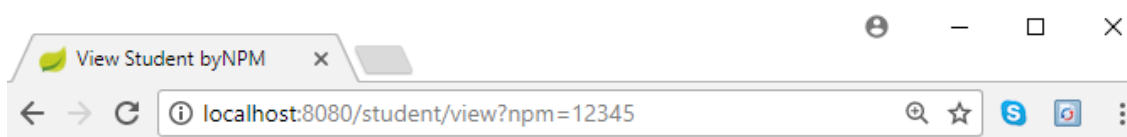
```
1 <!DOCTYPE html>  
2 <html xmlns:th="http://www.thymeleaf.org">  
3 <head>  
4 <title>View Student byNPM</title>  
5 </head>  
6 <body>  
7 <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>  
8 <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>  
9 <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>  
10 </body>  
11 </html>
```

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



Data berhasil ditambahkan

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?

➔ Ya, data student tersebut muncul.

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



Whitelabel Error Page

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

Fri Sep 22 23:07:44 ICT 2017

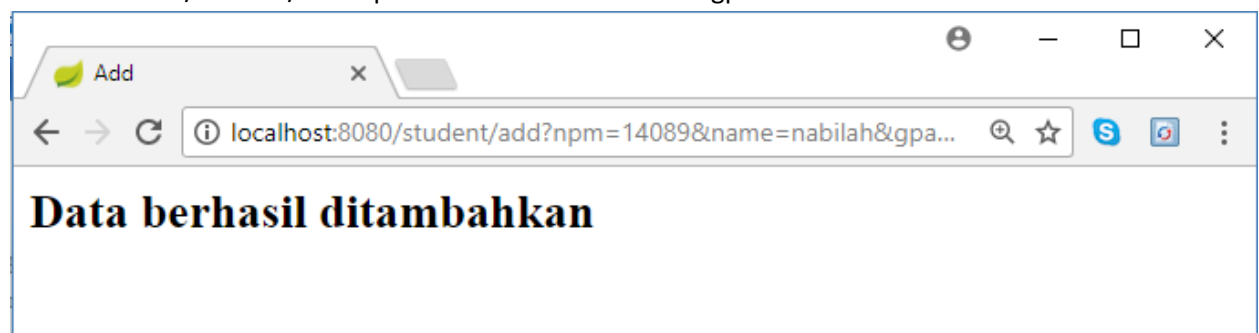
There was an unexpected error (type=Internal Server Error, status=500).

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

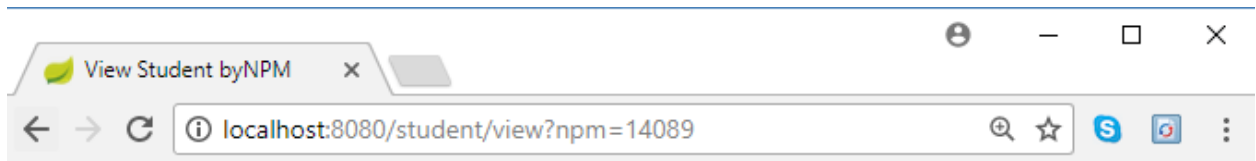
Pertanyaan 4: apakah data Student tersebut muncul? Jika tidak, mengapa?

➔ Tidak muncul. Hal ini disebabkan karena data dari Student disimpan pada setiap *session*, sedangkan *session* akan setiap kali *springboot* dijalankan. Sehingga data yang telah dimasukkan belum disave pada *session* yang baru tersebut.

5. Coba tambahkan data Student lainnya dengan NPM yang berbeda.
`localhost:8080/student/add?npm=14089&name=nabilah&gpa=3.90`



`http://localhost:8080/student/view?npm=14089`



NPM = 14089

Name = nabilah

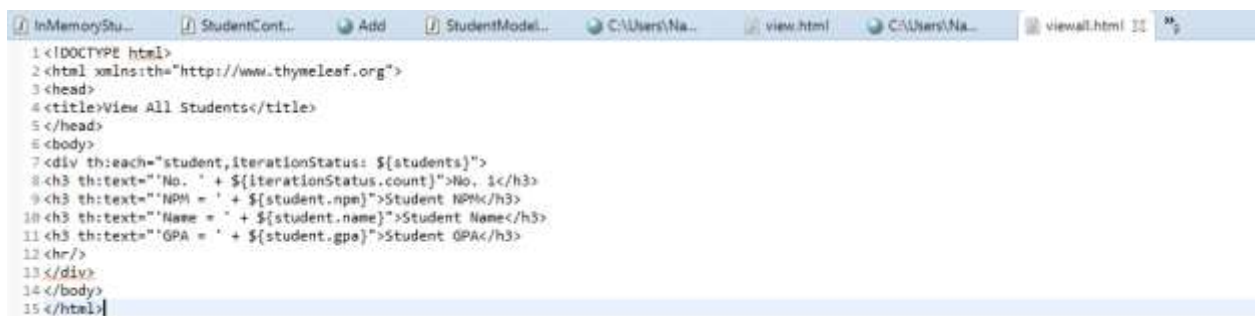
GPA = 3.9

Method View All

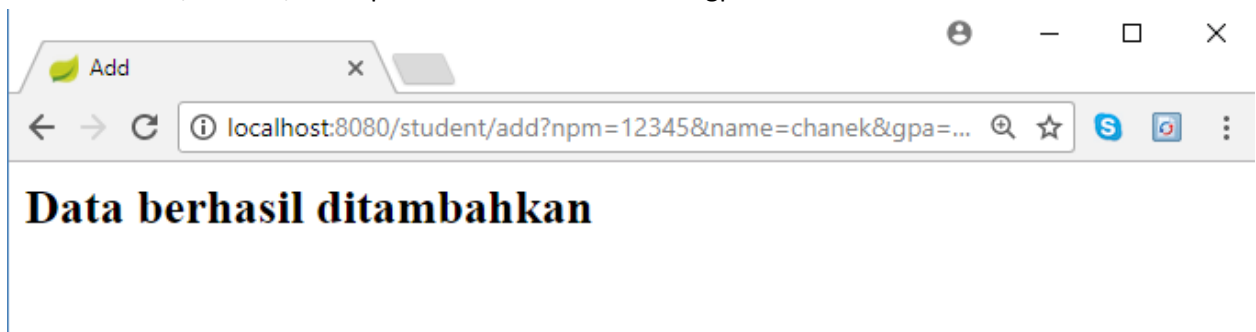
1. Pada class StudentController tambahkan method berikut

```
@RequestMapping("/student/viewall")
public String viewAll (Model model) {
    List<StudentModel> students = studentService.selectAllStudents();
    model.addAttribute("students", students);
    return "viewall";
}
```

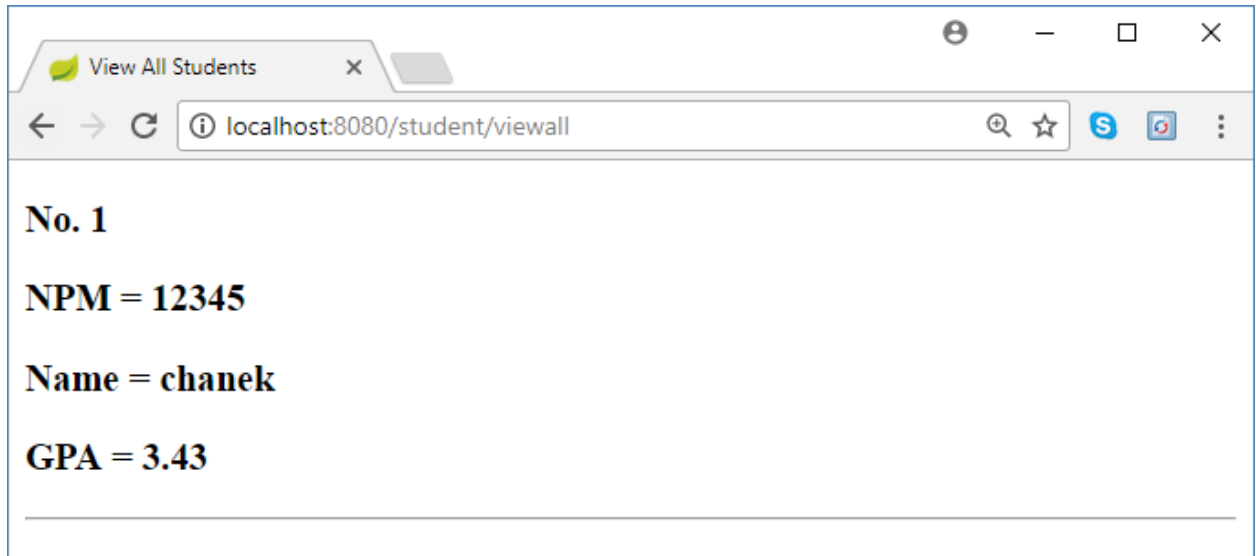
2. Selanjutnya pada directory resources/templates tambahkan viewall.html dengan isi sebagai berikut:



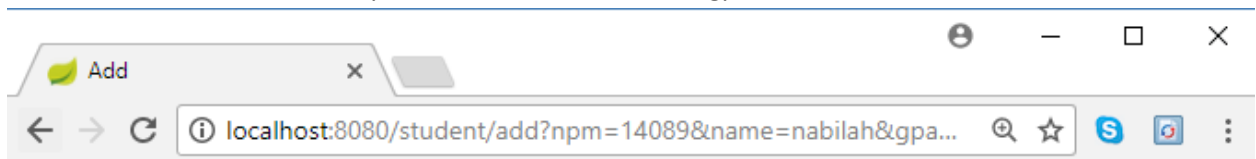
3. Jalankan program dan buka
localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43



lalu buka localhost:8080/student/viewall,
Pertanyaan 5: apakah data Student tersebut muncul?
➔ Ya, muncul.

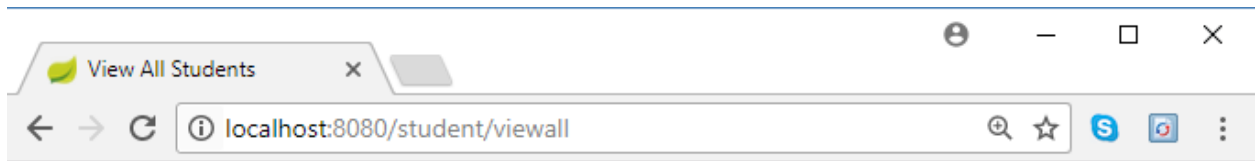


4. Coba tambahkan data Student lainnya dengan NPM yang berbeda,
localhost:8080/student/add?npm=14089&name=nabilah&gpa=3.97



Data berhasil ditambahkan

lalu buka localhost:8080/student/viewall,



No. 1

NPM = 12345

Name = chaneK

GPA = 3.43

No. 2

NPM = 14089

Name = nabilah

GPA = 3.97

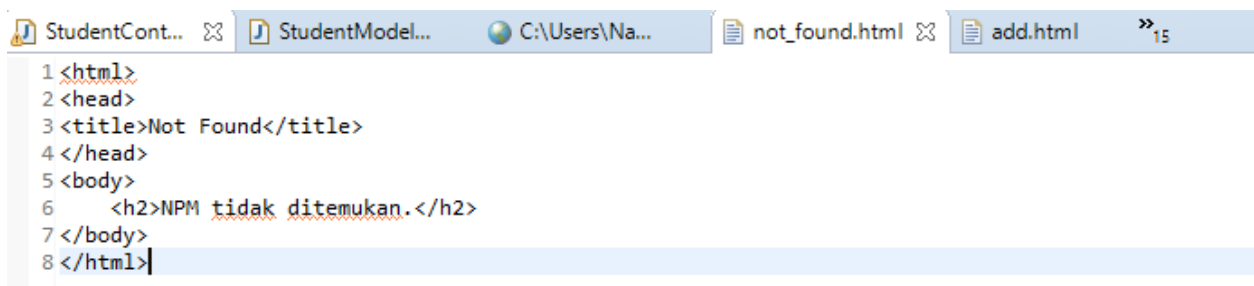
Pertanyaan 6: Apakah semua data Student muncul?

➔ Ya, muncul.

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.

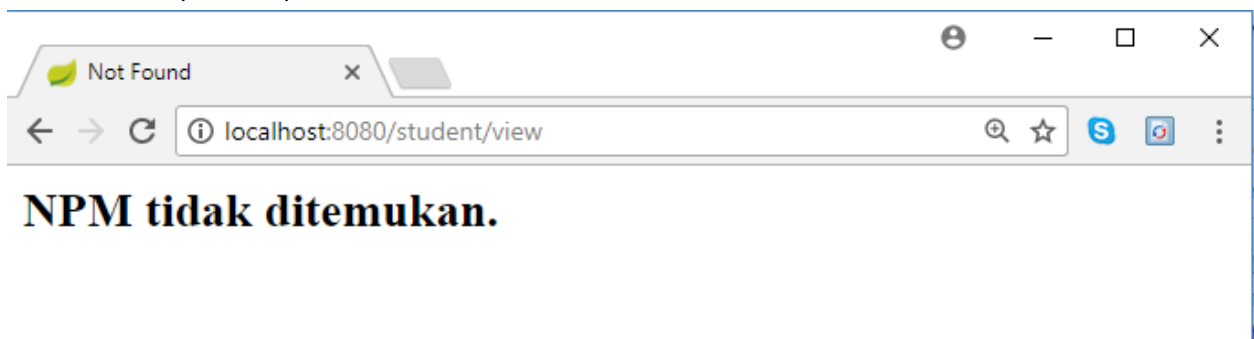

```
@RequestMapping(value = {"/student/view" ,"/student/view/{npm}"})  
public String view(Model model, @PathVariable Optional<String> npm) {  
  
    if(npm.isPresent()) {  
        StudentModel student = studentService.selectStudent(npm.get());  
  
        if(student == null) {  
            return "not_found";  
        } else {  
            model.addAttribute("name", student.getName());  
            model.addAttribute("npm", student.getNpm());  
            model.addAttribute("gpa", student.getGpa());  
            return "view";  
        }  
    } else {  
        return "not_found";  
    }  
}
```



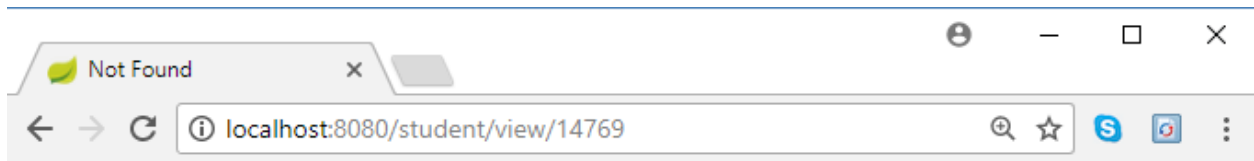
The screenshot shows an IDE window with several tabs: 'StudentCont...', 'StudentModel...', 'C:\Users\Na...', 'not_found.html', and 'add.html'. The 'not_found.html' tab is active, displaying the following HTML code:

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

localhost:8080/student/view

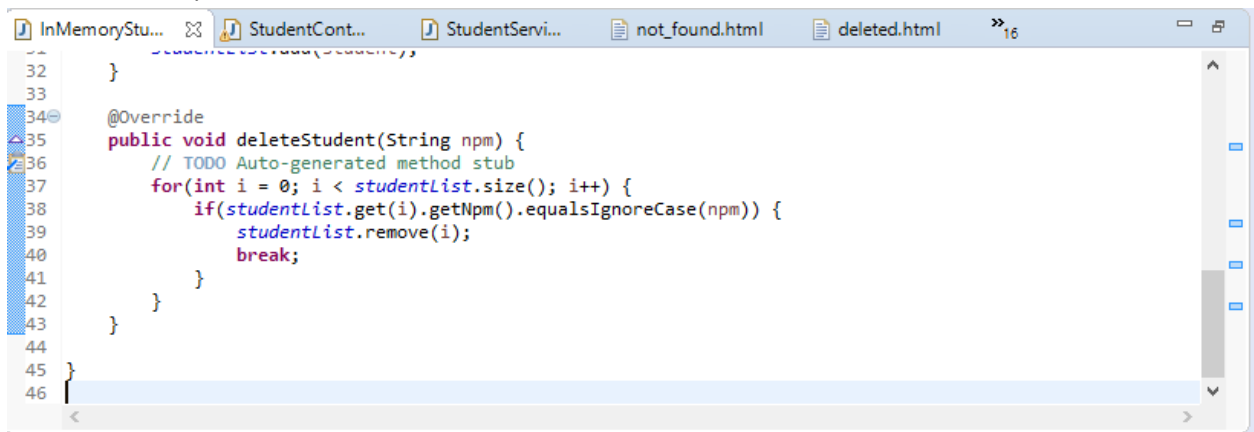


localhost:8080/student/view/14769



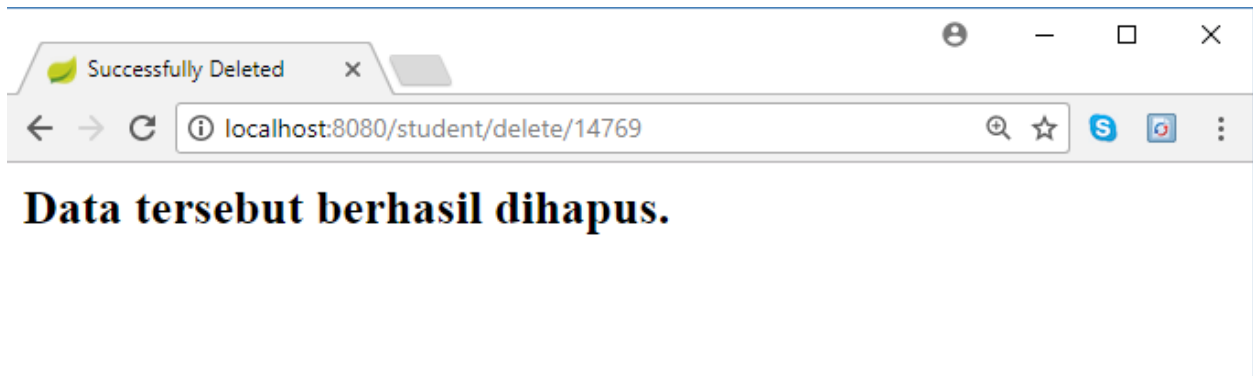
NPM tidak ditemukan.

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.

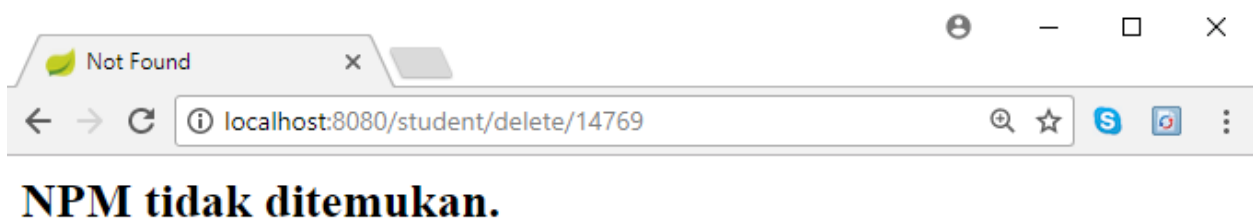


```
@RequestMapping(value = {"/student/delete", "/student/delete/{npm}"})
public String delete(Model model, @PathVariable Optional<String> npm) {
    if(npm.isPresent()) {
        StudentModel s = studentService.selectStudent(npm.get());
        if(s == null) return "not_found";
        studentService.deleteStudent(npm.get());
        return "deleted";
    }

    return "not_found";
}
```



Setelah tidak ada data lagi, saat melakukan delete akan keluar:



TAHAP-TAHAP YANG DILAKUKAN:

Method View Student

Pertama-pertama saya perlu membuat path variable untuk menyimpan parameter NPM yang nantinya akan disimpan didalam suatu variable dengan tipe data `optional<String>` agar dapat membedakan parameter NPM ada atau tidak ada. Setelah itu saya mengecek `isPresent()` dari variable `npm`, jika ada maka objek `student` akan dicari berdasarkan `npm`, jika tidak maka akan mengembalikan page `not_found`, jika dalam proses pencarian tidak ditemukan `npm` yang diminta, maka akan mengembalikan page `not_found` juga

Method Delete Student

Pertama-pertama saya perlu membuat path variable untuk menyimpan parameter NPM yang nantinya akan disimpan didalam suatu variable dengan tipe data `optional<String>` agar dapat membedakan parameter NPM ada atau tidak ada. Setelah itu saya mengecek `isPresent()` dari variable `npm`, jika ada maka objek `student` akan dicari berdasarkan `npm`, jika tidak ada maka akan mengembalikan `not_found`, jika ada maka objek tersebut akan di hapus dari arraylist `studentList`.