

## TUTORIAL 3

### Menggunakan Model dan Service pada Project Spring Boot

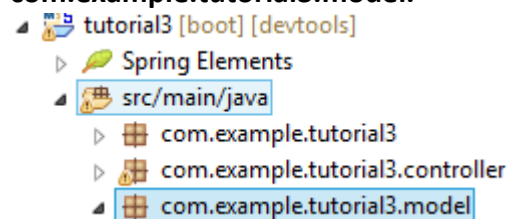
#### Ringkasan

Model merupakan sebuah objek yang merepresentasikan dan menyimpan informasi terhadap suatu hal. Contoh dari model pada tutorial kali ini adalah Mahasiswa yang memiliki nama, npm, dan nilai gpa (direpresentasikan sebagai atribut). Service merupakan suatu layer yang menjadi mediator antara controller dan database. Service layer berguna untuk mengolah data pada database seperti kalkulasi data, input data oleh user, remove data, dan sebagainya. Data pada database juga dapat ditampilkan dengan method view pada controller yang menggunakan teknologi Thymeleaf dan Spring Boot.

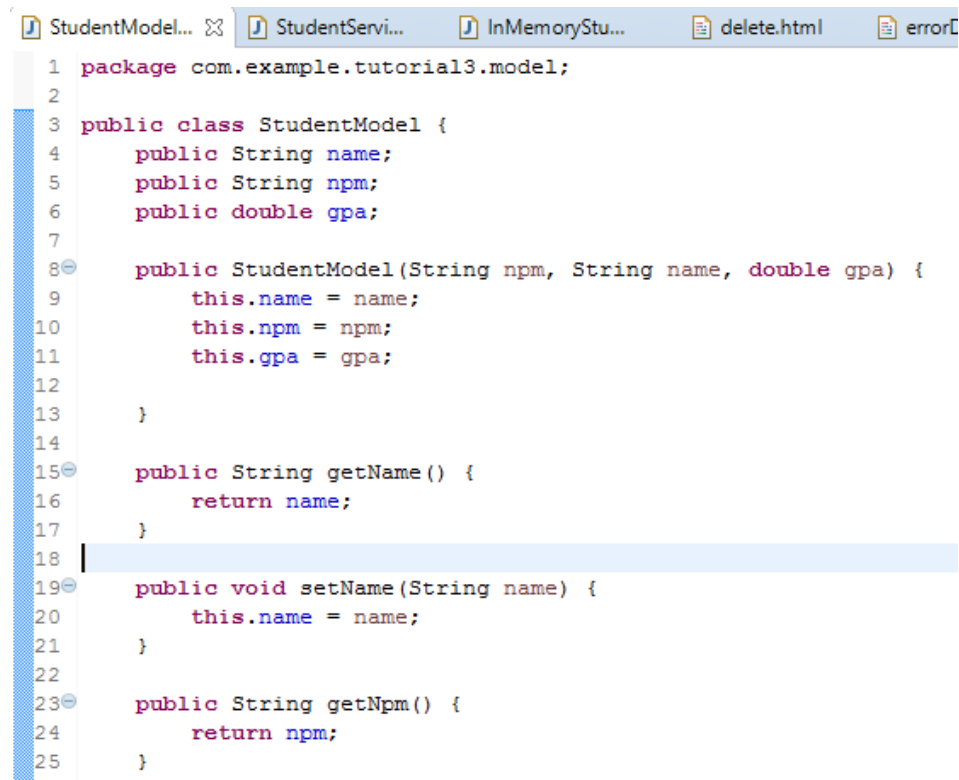
#### Membuat Class Model

1. Klik kanan pada Project > New > Package. Buat package

**com.example.tutorial3.model.**




2. Buatlah class **StudentModel** dengan spesifikasi seperti di atas pada package tersebut.

A screenshot of an IDE showing the code for the 'StudentModel' class. The code is as follows:

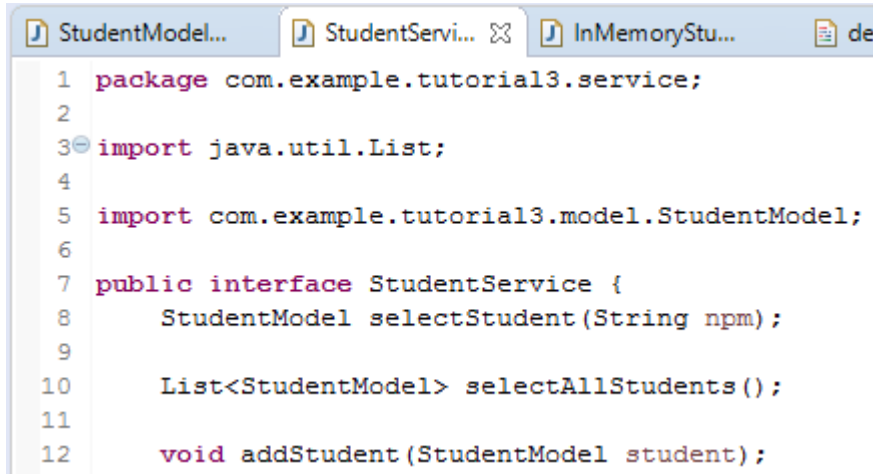
```
1 package com.example.tutorial3.model;
2
3 public class StudentModel {
4     public String name;
5     public String npm;
6     public double gpa;
7
8     public StudentModel(String npm, String name, double gpa) {
9         this.name = name;
10        this.npm = npm;
11        this.gpa = gpa;
12    }
13
14
15    public String getName() {
16        return name;
17    }
18
19    public void setName(String name) {
20        this.name = name;
21    }
22
23    public String getNpm() {
24        return npm;
25    }
26 }
```

### Membuat Service

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

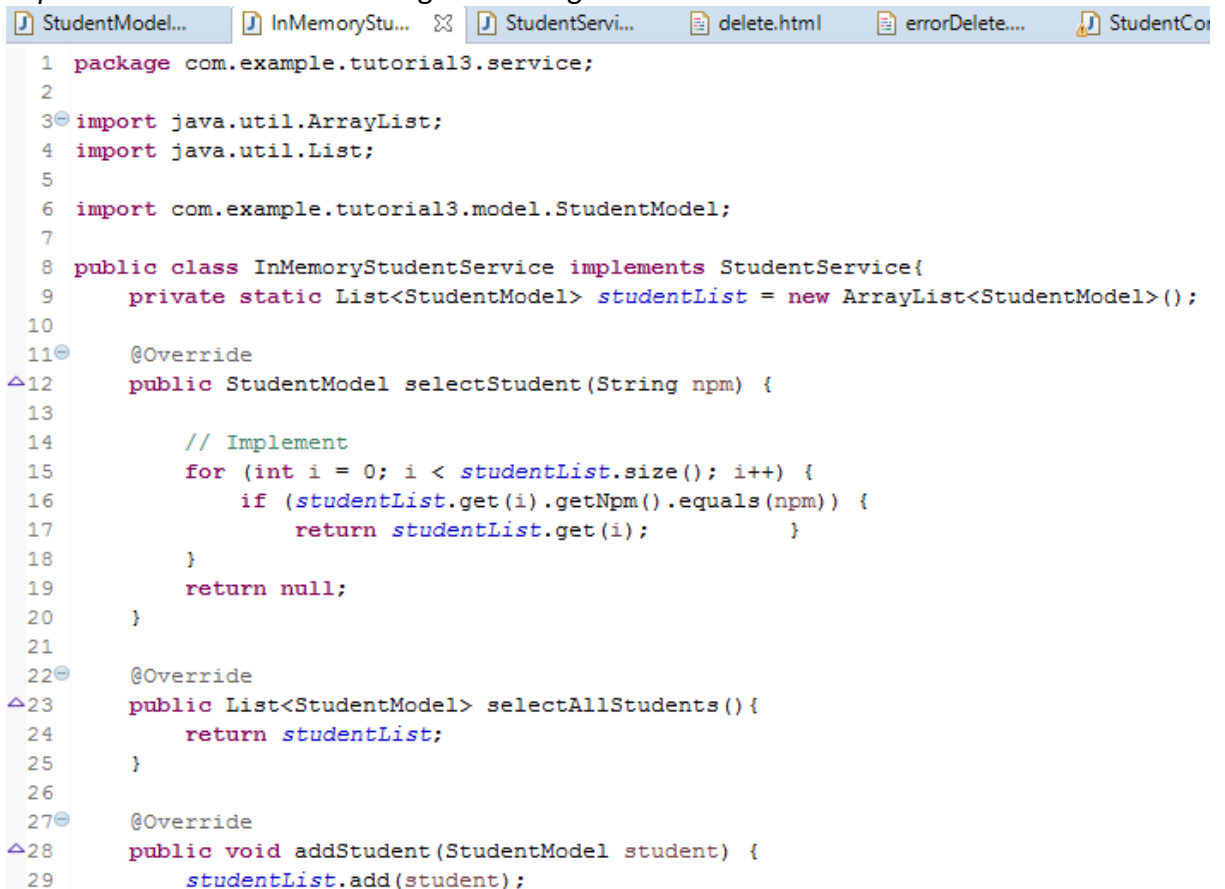
▶  com.example.tutorial3.service

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



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

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



```
1 package com.example.tutorial3.service;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import com.example.tutorial3.model.StudentModel;
7
8 public class InMemoryStudentService implements StudentService{
9     private static List<StudentModel> studentList = new ArrayList<StudentModel>();
10
11     @Override
12     public StudentModel selectStudent(String npm) {
13
14         // Implement
15         for (int i = 0; i < studentList.size(); i++) {
16             if (studentList.get(i).getNpm().equals(npm)) {
17                 return studentList.get(i);
18             }
19             return null;
20         }
21
22     @Override
23     public List<StudentModel> selectAllStudents() {
24         return studentList;
25     }
26
27     @Override
28     public void addStudent(StudentModel student) {
29         studentList.add(student);
```

- Implementasikan *method* `selectStudent`! *Method* ini menerima NPM mahasiswa dan mengembalikan *object* `Student` dengan NPM tersebut. Return null jika tidak ditemukan.


Method `selectStudent`

```
@Override
public StudentModel selectStudent(String npm) {

    // Implement
    for (int i = 0; i < studentList.size(); i++) {
        if (studentList.get(i).getNpm().equals(npm)) {
            return studentList.get(i);
        }
    }
    return null;
}
```

### Membuat Controller dan Fungsi Add

- Klik kanan pada Project > New > Package. Buatlah *package* **com.example.tutorial3.controller**

▷  com.example.tutorial3.controller

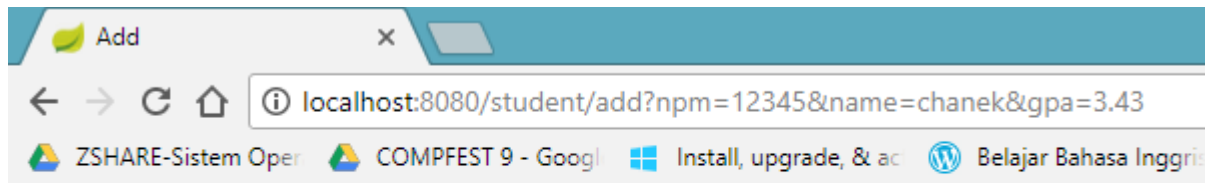
- Tambahkan method *add* yang menerima parameter `name`, `npm`, dan `gpa` dengan menggunakan request method GET. Buatlah *class* **StudentController** dengan isi sebagai berikut:

```
@RequestMapping("/student/add")
public String add(@RequestParam(value = "npm", required = true) String npm,
                 @RequestParam(value = "name", required = true) String name,
                 @RequestParam(value = "gpa", required = true) double gpa) {
    StudentModel student = new StudentModel(npm, name, gpa);
    studentService.addStudent(student);
    return "add";
}
```

- Selanjutnya pada *directory* `resources/templates` tambahkan `add.html` dengan isi sebagai berikut:

```
1 <html>
2   <head>
3     <title>Add</title>
4   </head>
5   <body>
6     <h2>Data berhasil ditambahkan</h2>
7   </body>
8 </html>
```

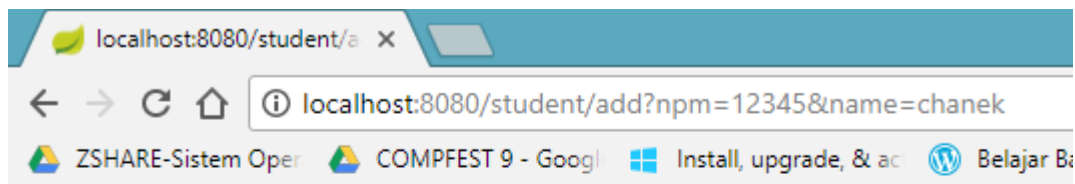
- Jalankan program dan buka  
Localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43



## Data berhasil ditambahkan

\*output di atas menandakan bahwa model mahasiswa dengan npm 12345, name chanek, dan gpa 3.43 berhasil ditambahkan ke database.

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.

Sat Sep 23 20:26:45 ICT 2017

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

Required double parameter 'gpa' is not present

\*Terjadi *error*, karena gpa tidak dimasukkan oleh *user* sementara nilai gpa required=true (required dan tidak bernilai *null* jika tidak di-input oleh *user*).

### Method View by NPM

1. Tambahkan method view pada StudentController

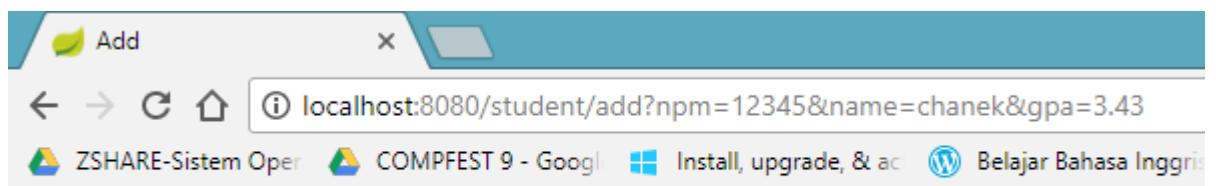
```
@RequestMapping("/student/view")
public String view(Model model, @RequestParam(value = "npm", required = true) String npm) {
    StudentModel student = studentService.selectStudent(npm);
    model.addAttribute("student", student);
    return "view";
}
```

2. Buat 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>
```

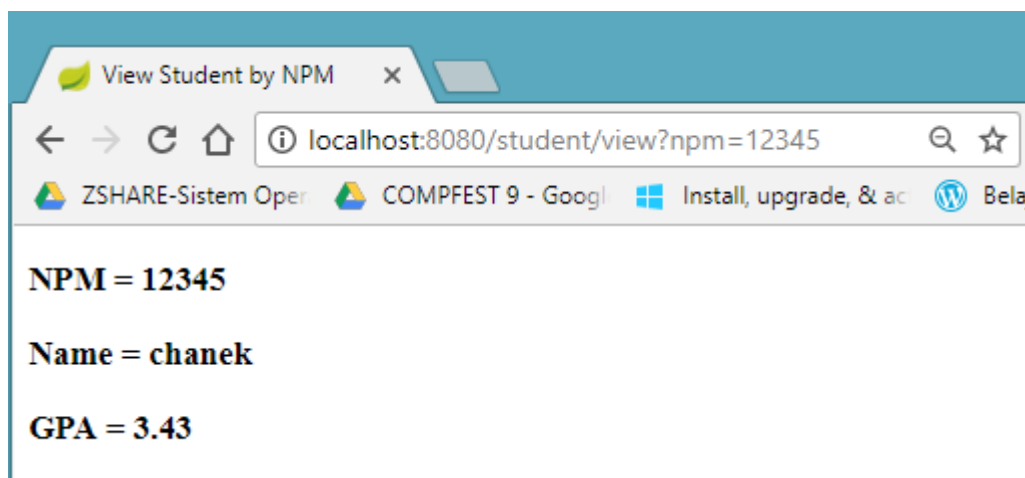
3. Jalankan program dan buka

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



## Data berhasil ditambahkan

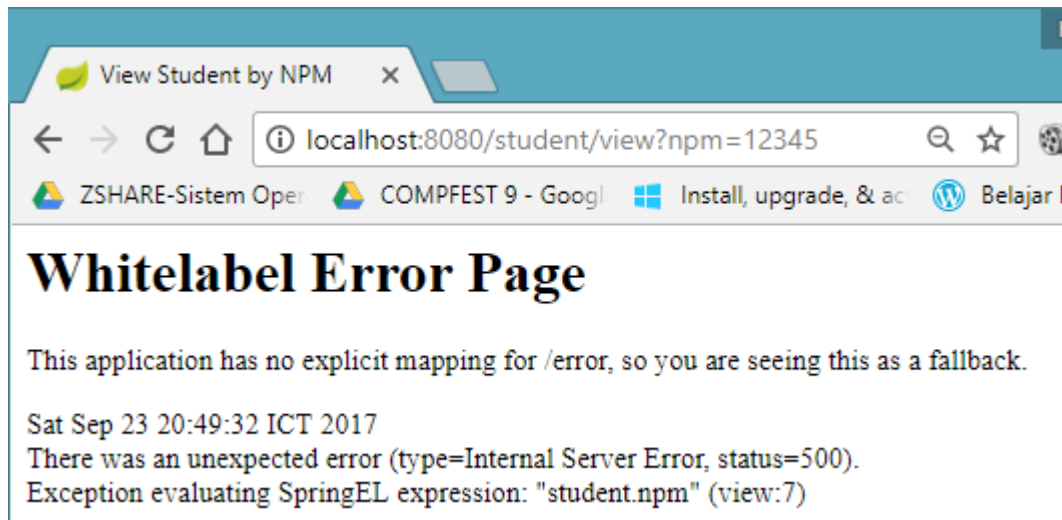
Lalu buka localhost:8080/student/view?npm=12345



\*data mahasiswa berhasil ditampilkan

4. Coba matikan program dan jalankan kembali serta buka

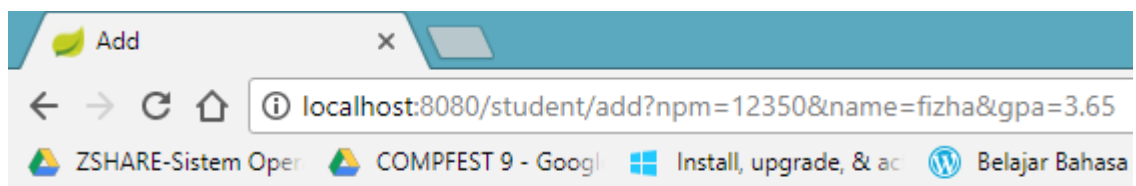
localhost:8080/student/view?npm=12345



\*terjadi *error* program tidak menemukan *object student* karena saat program dimatikan data yang tadinya disimpan terhapus sehingga saat di-*run* kembali data tidak dapat ditampilkan.

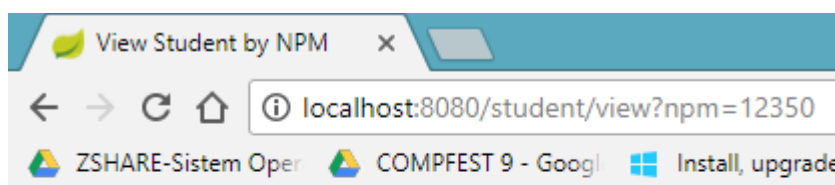
5. Coba tambahkan data Student lainnya dengan NPM yang berbeda

Data lain ditambahkan:



## Data berhasil ditambahkan

View data:



**NPM = 12350**

**Name = fizha**

**GPA = 3.65**

### Method View All

1. Buat method `viewAll` pada `StudentController`

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

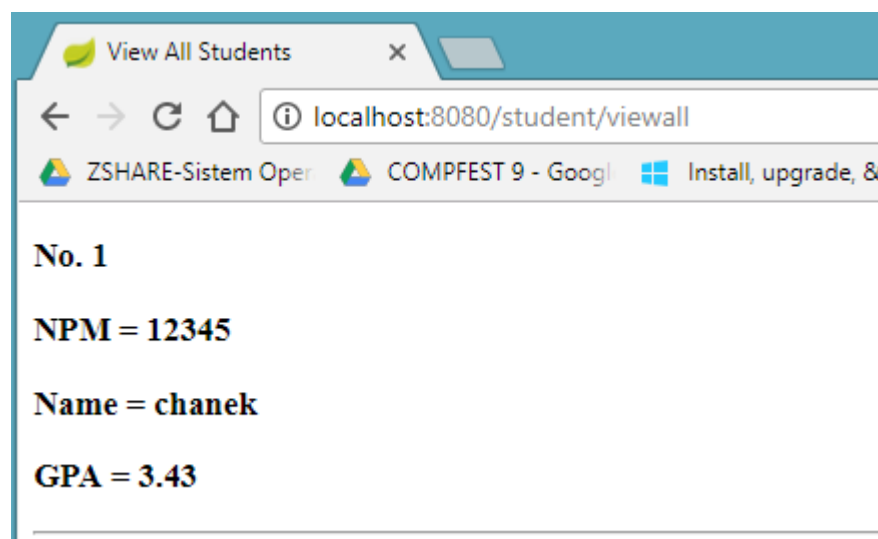
2. Buat viewAll.html

```
<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
    <head>  
        <title>View All Students</title>  
    </head>  
    <body>  
        <div th:each="student, iterationStatus: ${students}">  
            <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>  
            <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>  
            <hr/>  
        </div>  
    </body>  
</html>
```

3. Jalankan program dan buka

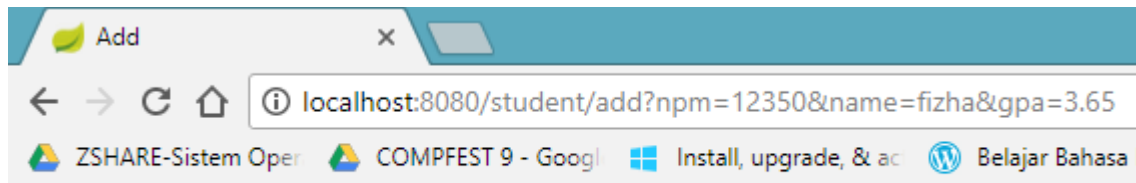
localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43

lalu buka localhost:8080/student/viewall



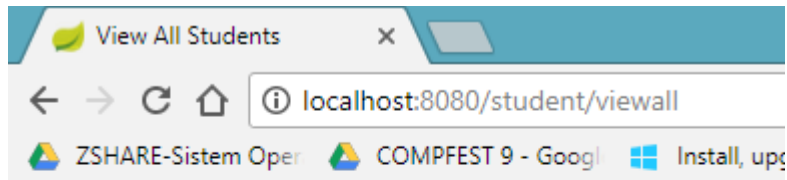
\*data Student berhasil ditampilkan

4. Coba tambahkan data Student lainnya dengan NPM yang berbeda,  
Tambah data mahasiswa



## Data berhasil ditambahkan

lalu buka localhost:8080/student/viewall



No. 1

**NPM = 12345**

**Name = chanek**

**GPA = 3.43**

---

No. 2

**NPM = 12350**

**Name = fizha**

**GPA = 3.65**

---

\*semua data Student berhasil ditampilkan

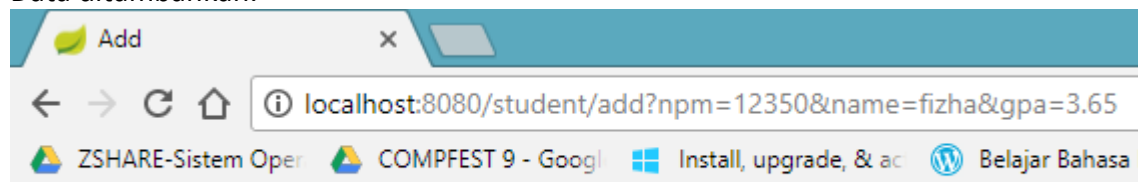
### Latihan

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**.  
Tambahkan *method view*:



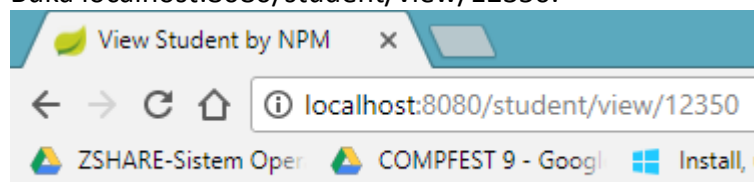
```
@RequestMapping(value = {"/student/view/{npm}"})  
public String viewPath(@PathVariable Optional<String> npm, Model model) {  
    if (npm.isPresent()) {  
        StudentModel student = studentService.selectStudent(npm.get());  
        model.addAttribute("student", student);  
        return "view";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

Data ditambahkan.



## Data berhasil ditambahkan

Buka localhost:8080/student/view/12350.



**NPM = 12350**

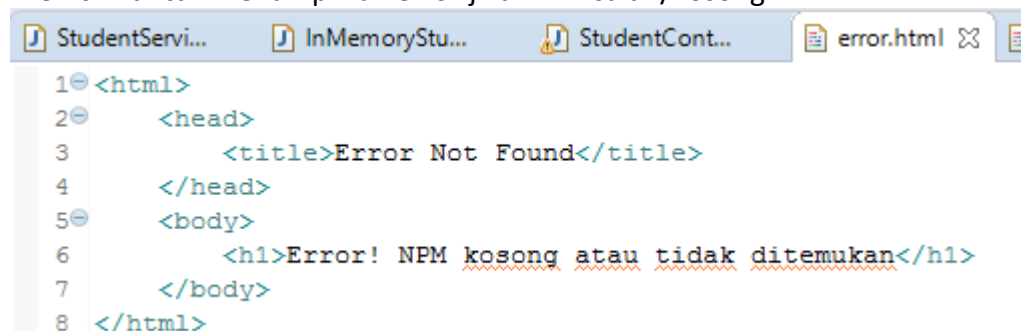
**Name = fizha**

**GPA = 3.65**

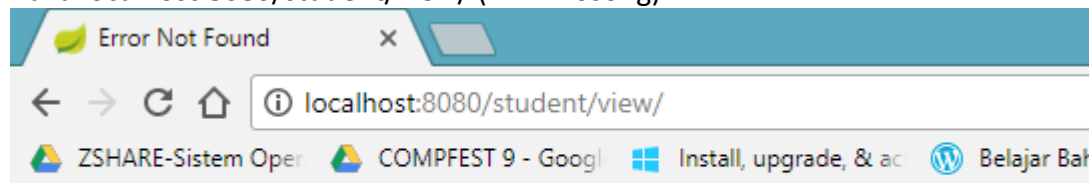
\*data berhasil ditampilkan menggunakan Path Variable

Jika nomor NPM tidak diberikan atau tidak ditemukan kembalikan halaman *error* yang berisi informasi bahwa nomor NPM kosong atau tidak ditemukan.

File html untuk menampilkan *error* jika NPM salah/kosong:

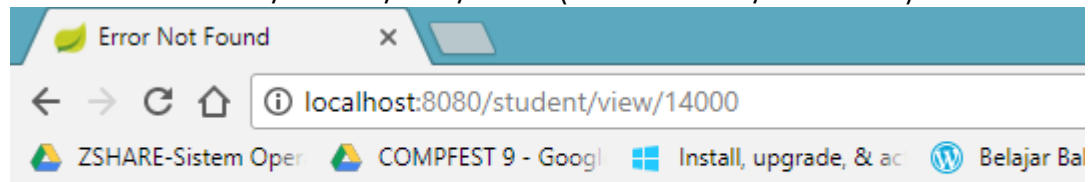


Buka localhost:8080/student/view/ (NPM kosong)



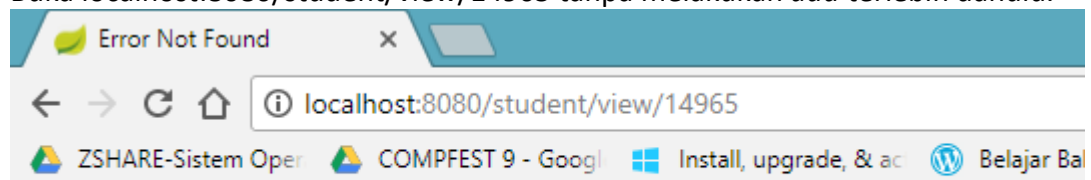
## Error! NPM kosong atau tidak ditemukan

Buka localhost:8080/student/view/14000 (data tidak ada/NPM salah)



## Error! NPM kosong atau tidak ditemukan

Buka localhost:8080/student/view/14965 tanpa melakukan *add* terlebih dahulu.



## Error! NPM kosong atau 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.

Tambah *interface* `removeStudent`:

```
void removeStudent(StudentModel student);
```

Tambah *method* dan implementasikan `removeStudent`:

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

Pada *student controller*, tambah *method delete* yang mengimplementasikan Path Variable:

```

@RequestMapping(value = {"/student/delete/{npm}"})
public String delete(@PathVariable Optional<String> npm, Model model) {
    if (npm.isPresent() && studentService.selectStudent(npm.get()) != null) {
        String npm2 = npm.get();

        StudentModel student = studentService.selectStudent(npm.get());
        studentService.removeStudent(student);

        model.addAttribute("npm", npm2);
        return "delete";
    } else {
        return "errorDelete";
    }
}

```

\*npm2 berfungsi untuk menyimpan nilai npm yang dihapus yang akan diinformasikan di delete.html

Delete.html:

```

1 <!DOCTYPE html>
2 <html th:xmlns="http://www.thymeleaf.org">
3     <head>
4         <title>Delete Student</title>
5     </head>
6     <body>
7         <h3 th:text="'Mahasiswa dengan NPM ' + ${npm} + ' telah berhasil dihapus'">Delete student</h3>
8     </body>
9 </html>

```

Buat file errorDelete:

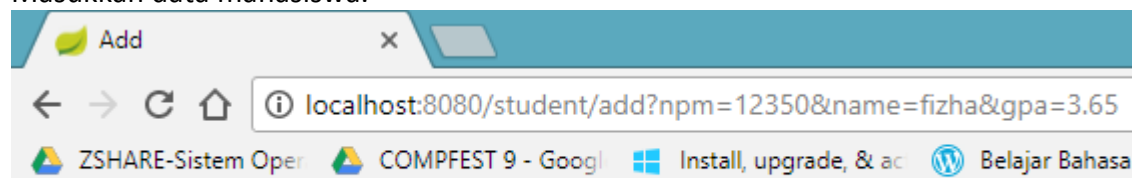
```

1 <html>
2     <head>
3         <title>Error Not Found | Delete canceled</title>
4     </head>
5     <body>
6         <h1>Error! NPM kosong atau tidak ditemukan</h1>
7         <h3>Proses delete mahasiswa dibatalkan</h3>
8     </body>
9 </html>

```

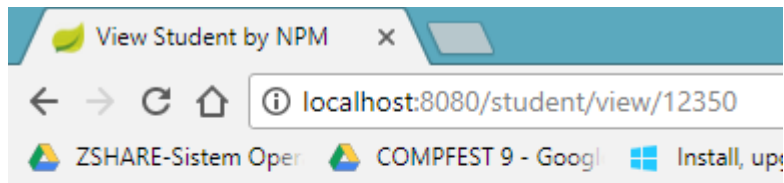
Jalankan program:

Masukkan data mahasiswa.



## Data berhasil ditambahkan

Tampilkan data mahasiswa yang dimasukkan.

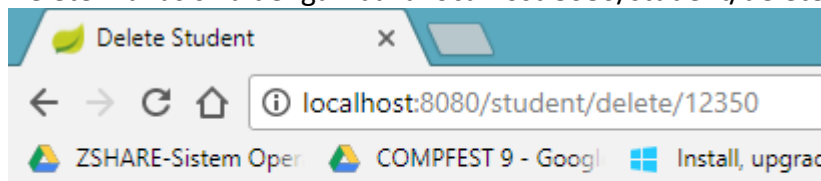


**NPM = 12350**

**Name = fizha**

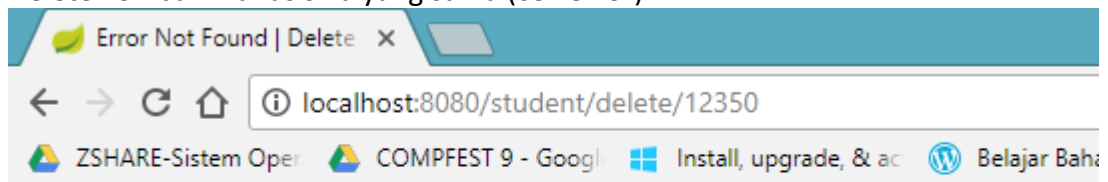
**GPA = 3.65**

Delete mahasiswa dengan buka localhost:8080/student/delete/12350



**Mahasiswa dengan NPM 12350 telah berhasil dihapus**

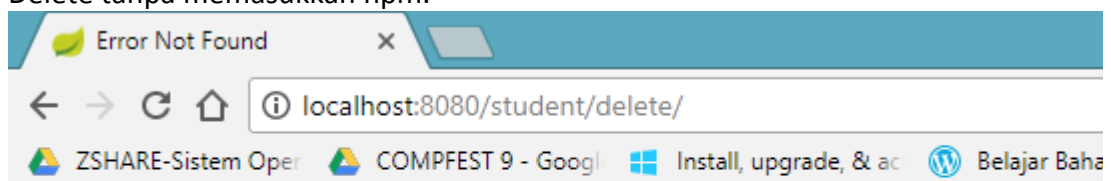
Delete kembali mahasiswa yang sama (cek error).



**Error! NPM kosong atau tidak ditemukan**

**Proses delete mahasiswa dibatalkan**

Delete tanpa memasukkan npm.



**Error! NPM kosong atau tidak ditemukan**