

Implementasi Method selectStudent

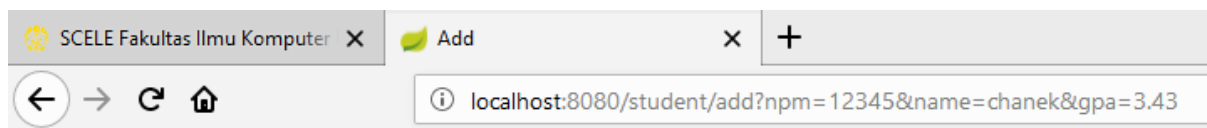
```
@Override
public StudentModel selectStudent(String npm) {
    for (StudentModel student : studentList) {
        if(student.getNpm().equals(npm)) {
            return student;
        }
    }

    return null;
}
```

Controller dan Fungsi Add

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

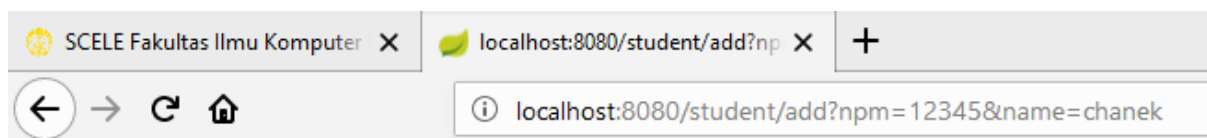
Pertanyaan 1: apakah hasilnya? Jika *error*, tuliskan penjelasan Anda.



Data berhasil ditambahkan

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

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



Whitelabel Error Page

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

Wed Feb 28 22:30:08 ICT 2018

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

Required double parameter 'gpa' is not present

Terjadi Whitelabel Error Page karena parameter 'gpa' tidak ditambahkan. Parameter 'gpa' dibutuhkan karena parameter tersebut bersifat *required*.

Method View by NPM

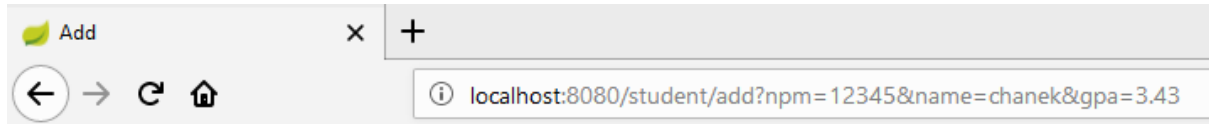
Jalankan program dan buka

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

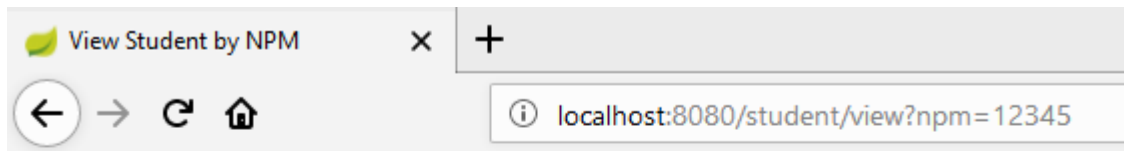
localhost:8080/student/view?npm=12345

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

Data student muncul.



Data berhasil ditambahkan



NPM = 12345

Name = chanek

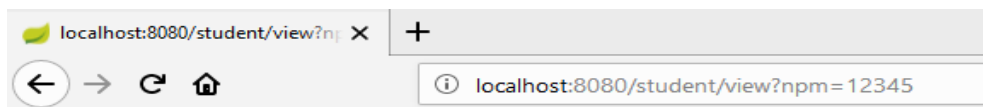
GPA = 3.43

4. Coba matikan program dan jalankan kembali serta buka

localhost:8080/student/view?npm=12345

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

Data student tidak muncul karena apabila session dimatikan dan dijalankan kembali, maka session diperbarui seluruhnya sehingga data student akan terhapus juga.



Whitelabel Error Page

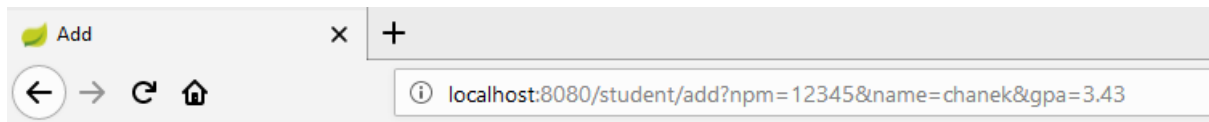
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri Mar 02 23:33:13 ICT 2018

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

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

5. Coba tambahkan data Student lainnya dengan NPM yang berbeda.



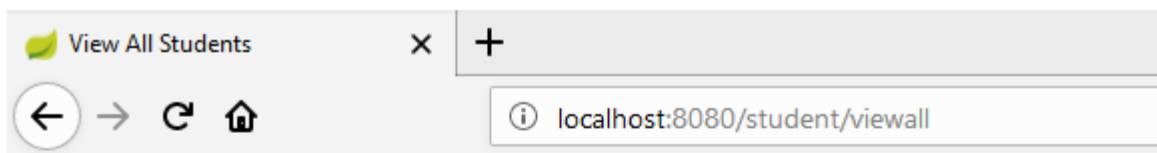
Data berhasil ditambahkan

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?

Data student mucul.



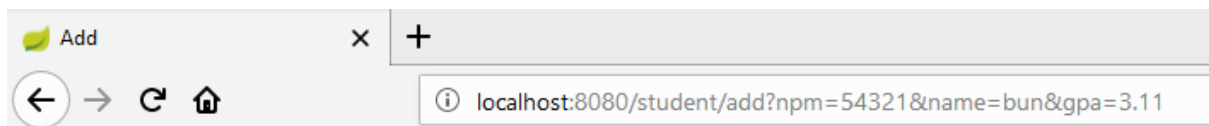
No. 1

NPM = 12345

Name = chanek

GPA = 3.43

Coba tambahkan data Student lainnya dengan NPM yang berbeda

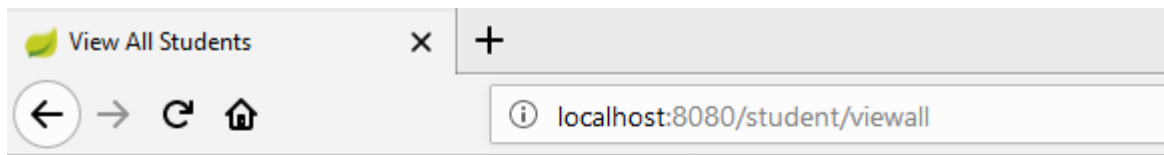


Data berhasil ditambahkan

localhost:8080/student/viewall,

Pertanyaan 6: Apakah semua data Student muncul?

Semua data student muncul.



No. 1

NPM = 12345

Name = chanek

GPA = 3.43

No. 2

NPM = 54321

Name = bun

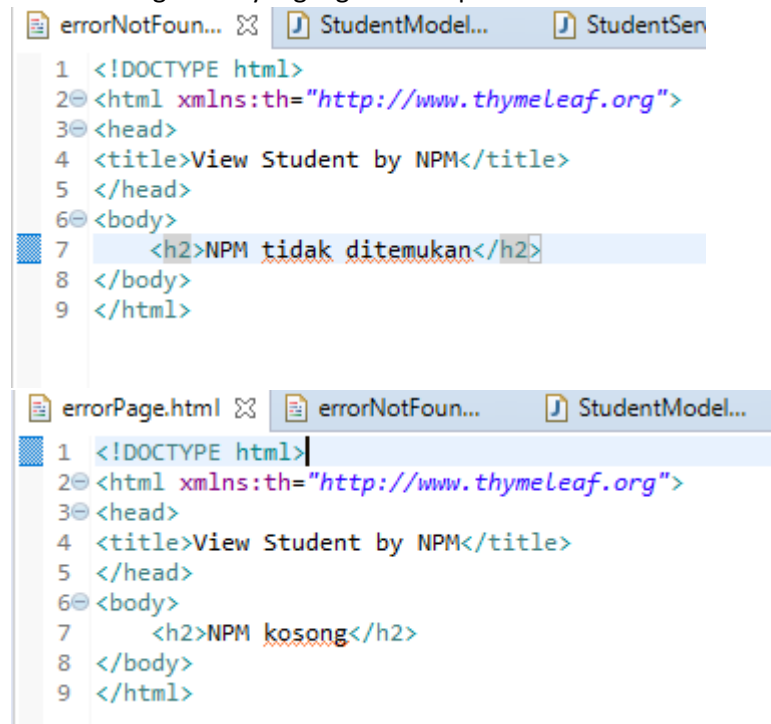
GPA = 3.11

Latihan

1. Pada studentController buat method view Student dengan menggunakan Path Variable.

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

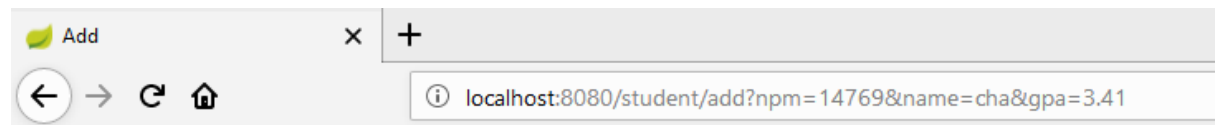
Buat halaman `errorNotFound.html` yang digunakan apabila informasi NPM tidak ditemukan dan `errorPage.html` yang digunakan apabila informasi NPM kosong.



```
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 <h2>NPM tidak ditemukan</h2>
8 </body>
9 </html>
```

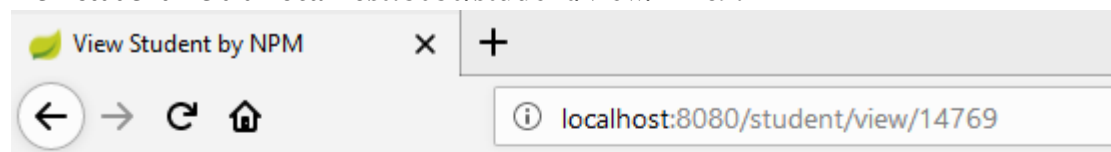
```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>View Student by NPM</title>
5 </head>
6 <body>
7 <h2>NPM kosong</h2>
8 </body>
9 </html>
```

Tambahkan data student.



Data berhasil ditambahkan

View student melalui **localhost:8080/student/view/14769**.

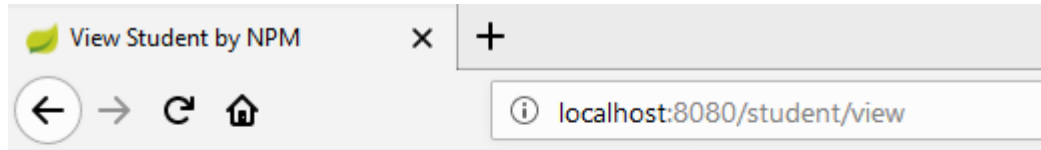


NPM = 14769

Name = cha

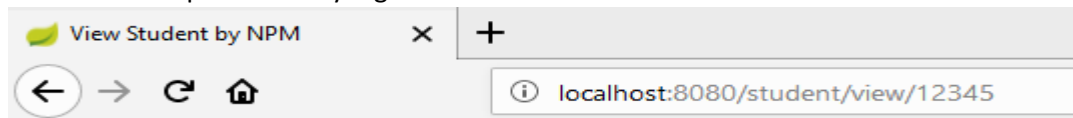
GPA = 3.41

View student apabila NPM tidak ada/kosong.



NPM kosong

View student apabila NPM yang dicari tidak ditemukan.



NPM tidak ditemukan

2. Fitur delete dapat menghapus data student. Hampir mirip seperti view, delete menghapus data melalui Path Variable dengan method deletePath dan method deleteStudent yang berada pada InMemoryStudentService.java.

Method delete student pada studentController.

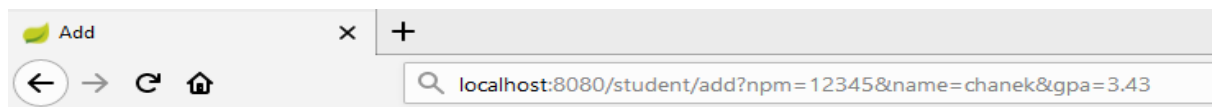
```
@RequestMapping(value = {"/student/delete", "/student/delete/{npm}"})
public String deletePath(Model model,@PathVariable Optional<String> npm) {
    if(npm.isPresent()) {
        StudentModel student = studentService.selectStudent(npm.get());
        if(student == null) {
            return "errorNotFound";
        }
        else {
            StudentModel studentDelete = studentService.deleteStudent(npm.get());
            //model.addAttribute("studentDelete", studentDelete);
            return "delete";
        }
    }
    else {
        return "errorPage";
    }
}
```

Tambahkan method delete InMemoryStudentService.java dan implementasikan pada interface studentService.

```
@Override
public StudentModel deleteStudent(String npm) {
    int counter = 0;
    StudentModel temp = null;
    for(StudentModel student : studentList) {
        if(student.getNpm().equals(npm)) {
            temp = studentList.get(counter);
            studentList.remove(counter);
        }
        counter++;
    }

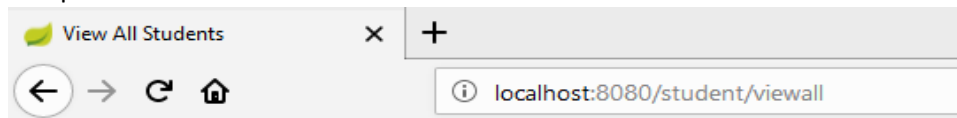
    return temp;
}
```

Menambahkan data student.



Data berhasil ditambahkan

Tampilkan semua data student.



No. 1

NPM = 14769

Name = cha

GPA = 3.41

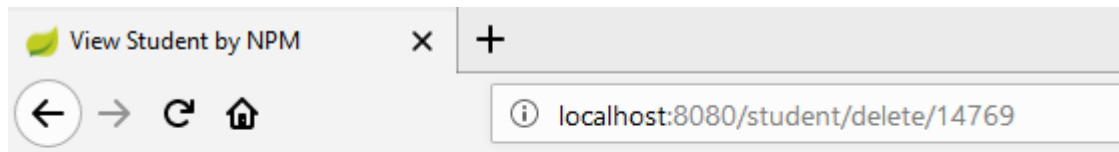
No. 2

NPM = 12345

Name = chanek

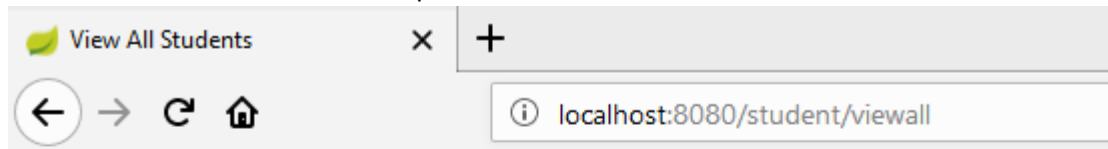
GPA = 3.43

Delete data student melalui **localhost:8080/student/view/14769**.



Data berhasil dihapus

Data student 14769 berhasil terhapus.



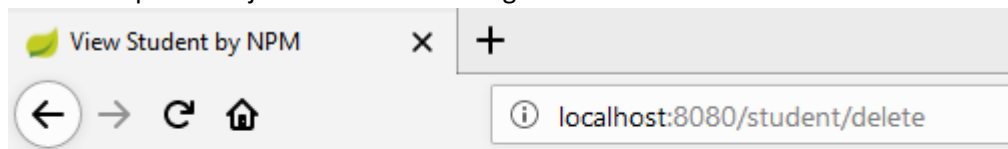
No. 1

NPM = 12345

Name = chanek

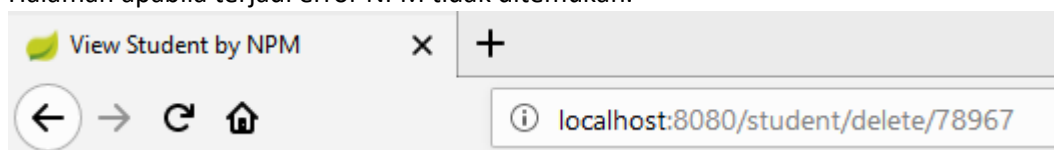
GPA = 3.43

Halaman apabila terjadi error NPM kosong.



NPM kosong

Halaman apabila terjadi error NPM tidak ditemukan.



NPM tidak ditemukan

Ringkasan Materi

Implementasi konsep *model-view-controller* (MVC) menggunakan Spring Boot. Komponen yang menjadi fokus dalam tutorial ini adalah *model* dan *service*. *Model* merupakan sebuah objek yang merepresentasikan dan menyimpan informasi terhadap suatu hal. Contoh dari model adalah objek mahasiswa. Objek mahasiswa ini memiliki nama, alamat, tanggal lahir,

nomor pokok mahasiswa, dsbnya. *Model* digunakan untuk merepresentasikan hal-hal tersebut dalam atribut yang dimiliki sebuah *model*.

Service adalah suatu *layer* yang menjadi mediator antara *controller* dan *database*. Pada *service layer*, disimpan *business logic* yang digunakan untuk mengolah data yang terdapat dalam *database*. Pengolahan ini meliputi kalkulasi data yang diambil dari *database*, manipulasi *userinput* kedalam *database*, dsbnya. Contohnya adalah melakukan kalkulasi IPK sebuah *model* mahasiswa.