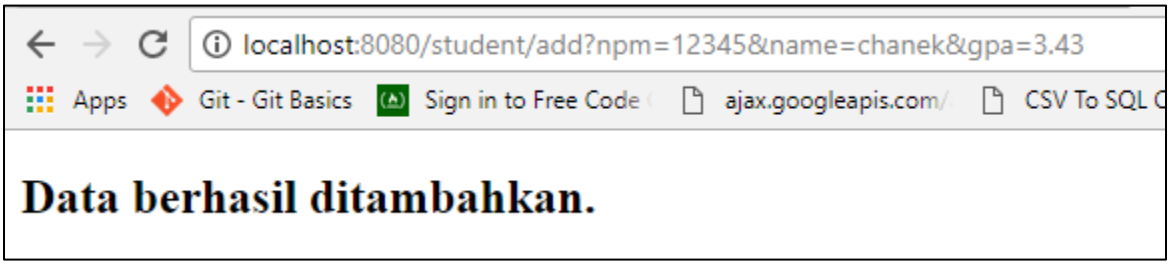
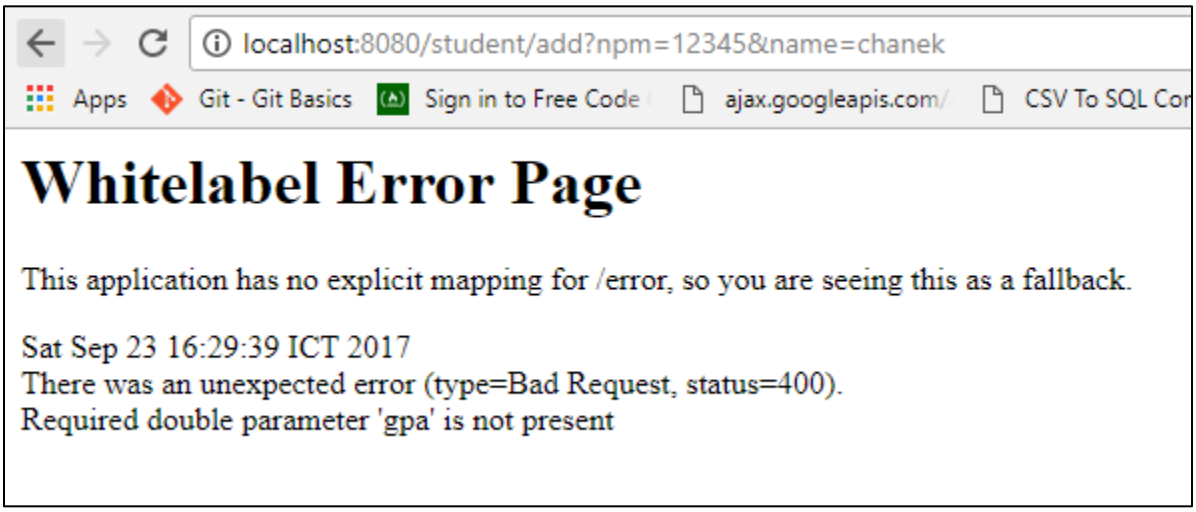
TUTORIAL 2 WRITE-UP

Pada tutorial ini saya mempelajari penyimpanan data dalam menggunakan Spring Boot yang secara tidak langsung juga menjelaskan kepada saya terkait implementasi konsep MVC lebih lanjut. Meski saya masih kurang paham terkait *service layer*¸ setidaknya tutorial kali ini memberikan penjelasan kepada saya bahwa salah satu fungsi pemisahan layer adalah untuk mempermudah *maintenance* program. Pemahaman saya terkait komponen *model* adalah sebuah abstraksi dari objek yang akan terus kita gunakan pada program, sementara *service layer* berisi operasi-operasi yang dilakukan terhadap objek dan media penyimpanannya (dalam tutorial ini misalnya *List*).

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

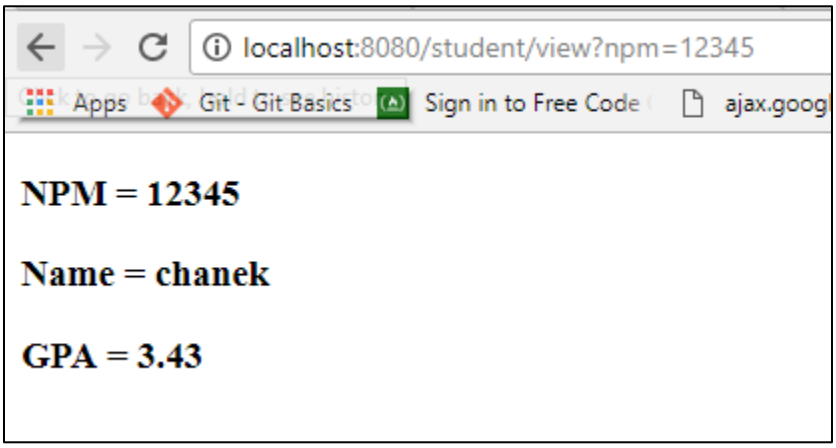


2. Jalankan program dan buka **localhost:8080/student/add?npm=12345&name=chanek.** Apakah hasilnya?



Tidak dapat menambahkan karena parameter gpa yang dibutuhkan untuk meng-*construct* objek Student tidak ada.

3. Jalankan program dan buka **localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43.** Lalu buka **localhost:8080/student/view?npm=12345.** Apakah data Student muncul?

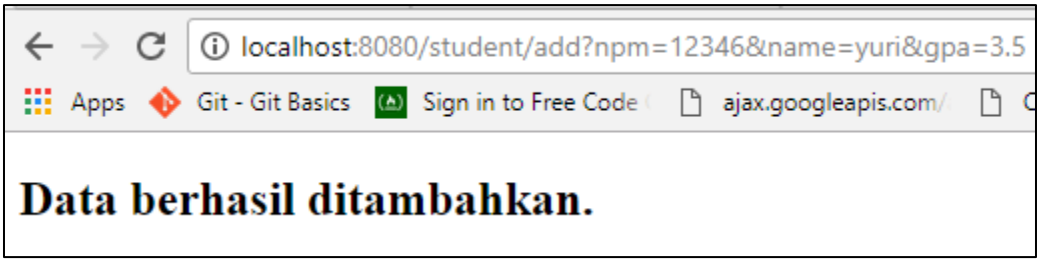


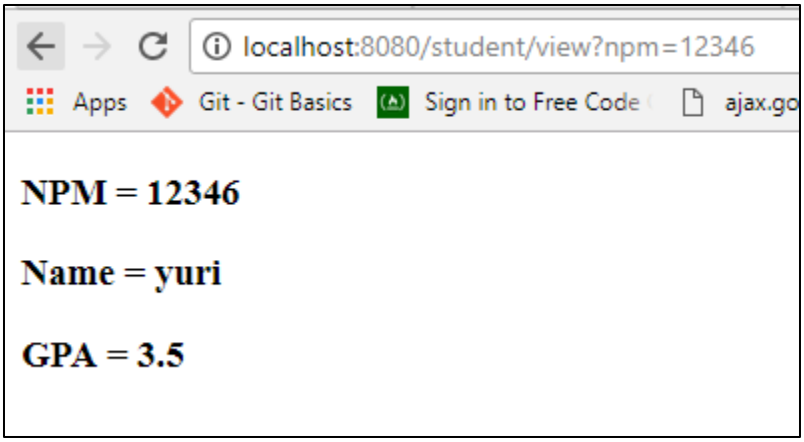
Ya, muncul.

4. Coba matikan program dan jalankan kembali serta buka **localhost:8080/student/view?npm=12345.** Apakah data Student muncul?

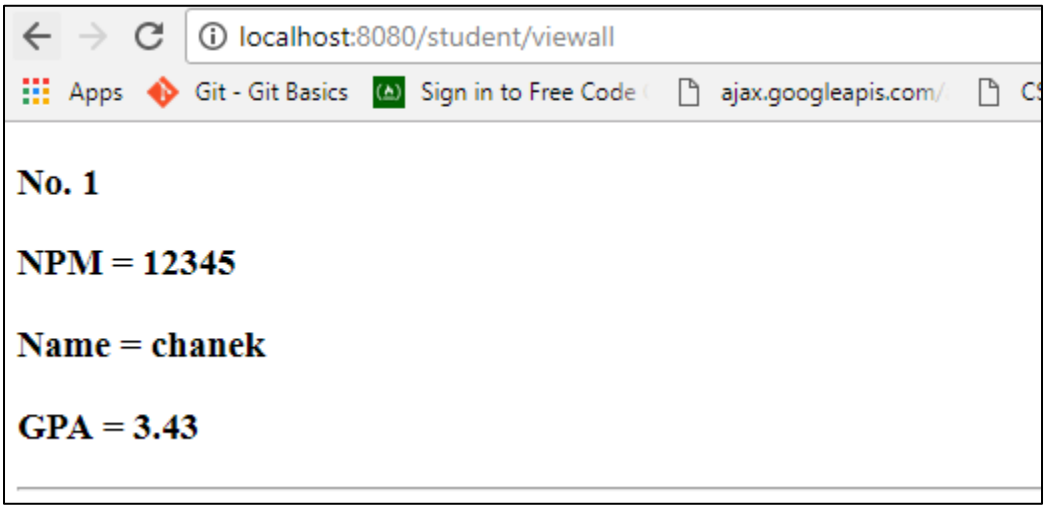
Data Student tidak muncul karena ketika program dimatikan, data yang sebelumnya ada dihapus. Mungkin kalau menggunakan database maka data tidak akan hilang.

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

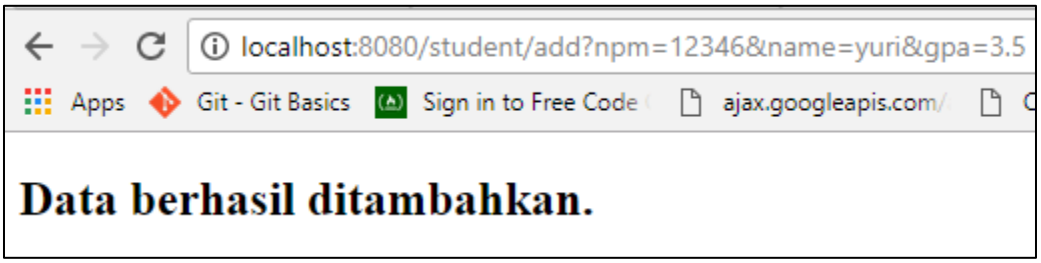


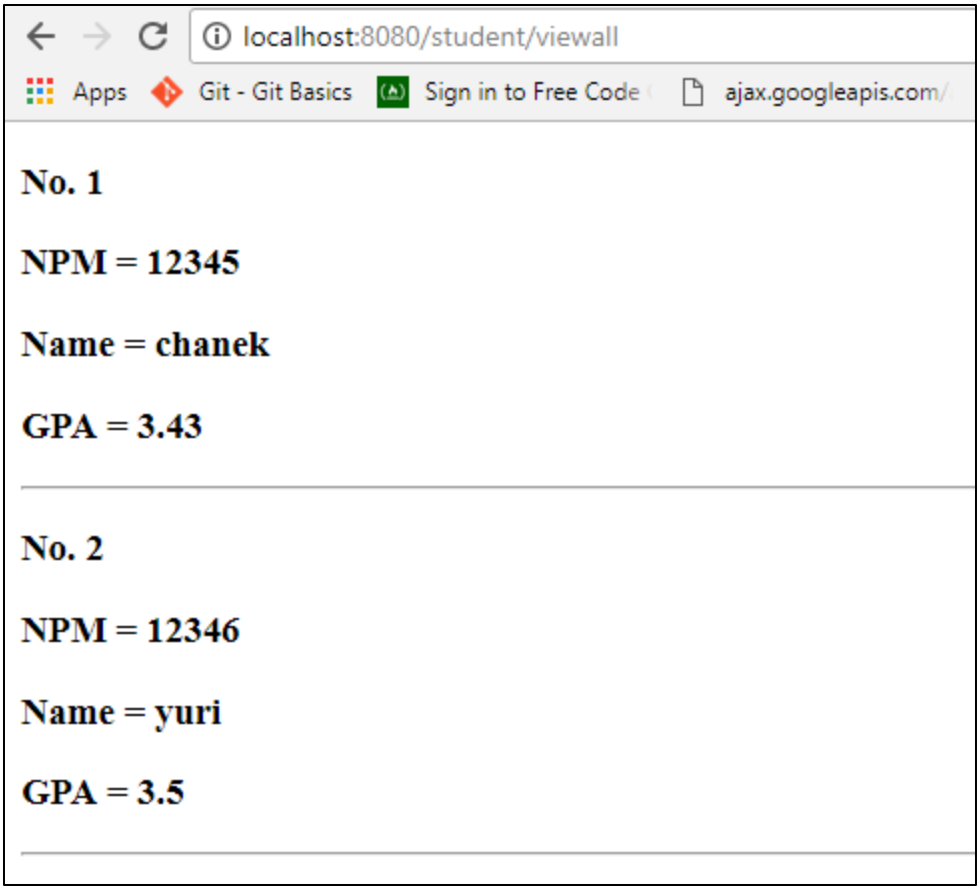


6. Jalankan program dan buka **localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43.** Lalu buka **localhost:8080/student/viewall.** Apakah data Student muncul?



7. Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka Lalu buka **localhost:8080/student/viewall.** Apakah semua data Student muncul?





Ya, data semua Student muncul.

Method selectStudent.

```
@Override
public StudentModel selectStudent(String npm) {
    int i = 0;
    while(i < studentList.size()) {
        if(studentList.get(i).getNpm().equals(npm)) {
            return studentList.get(i);
        }
        i++;
    }
    return null;
}
```

Method deleteStudent.

```
@Override
public StudentModel deleteStudent(String npm) {
    int i = 0;
    while(i < studentList.size()) {
        if(studentList.get(i).getNpm().equals(npm)) {
            studentList.remove(i);
        }
        i++;
    }
    return null;
}
```

Tahapan penulisan method deleteStudent

1. Membuat *interface*.

```
public interface StudentService {
    StudentModel selectStudent(String npm);
    List<StudentModel> selectAllStudents();
    void addStudent(StudentModel student);
    StudentModel deleteStudent(String npm);
}
```

2. Membuat implementasi *interface*

```
@Override
public StudentModel deleteStudent(String npm) {
    int i = 0;
    while(i < studentList.size()) {
        if(studentList.get(i).getNpm().equals(npm)) {
            studentList.remove(i);
        }
        i++;
    }
    return null;
}
```

3. Mendefiniskan *controller*

```
@RequestMapping("/student/delete/{npm}")
public String deletePath(@PathVariable String npm) {
    studentService.deleteStudent(npm);
    return "delete";
}
```

4. Membuat *view*

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Deletion</title>
5 </head>
6 <body>
7     <h1>Student deleted</h1>
8 </body>
9 </html>
```