1. **Ringkasan Materi**

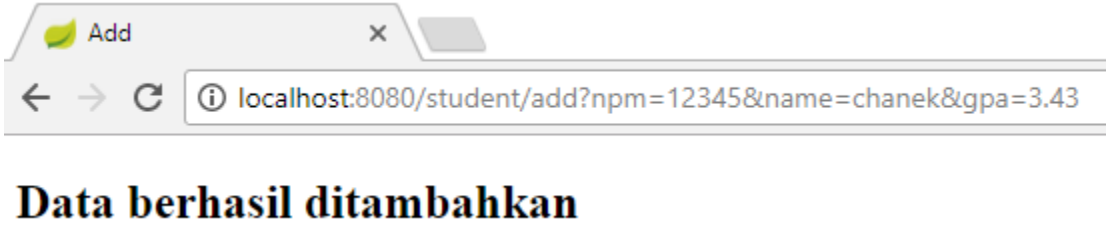
   Pada tutorial kali ini menggunakan konsep MVC (Model-View-Controller). Pada tutorial 2 hanya digunakan View yang akan ditampilkan pada browser dan Controller yang berisi method-method yang dapat diakses melalui url pada browser. Model digunakan untuk mendeklarasikan object yang akan digunakan beserta attribute-nya. Pada tutorial ini digunakan model StudentModel yang mempunyai attribute nama, npm, gpa. Selain tiga komponen di atas juga digunakan komponen service.

   Pada komponen service ini terdapat interface dan class yang mengimplementasikan interface. Pada interface di deklarasikan method-method yang akan digunakan. Sedangkan implementasi method-nya diletakkan pada class tersendiri, pada tutorial ini implementasi method pada class InMemoryStudentServices. Untuk dapat mengakses method-method yang sudah diimplementasikan maka Controller perlu untuk membuat definisi service yang akan digunakan. Selain itu, pada setiap running semua data akan diputihkan sehingga jika memang belum diberikan nilai maka variable akan bersifat kosong atau null.

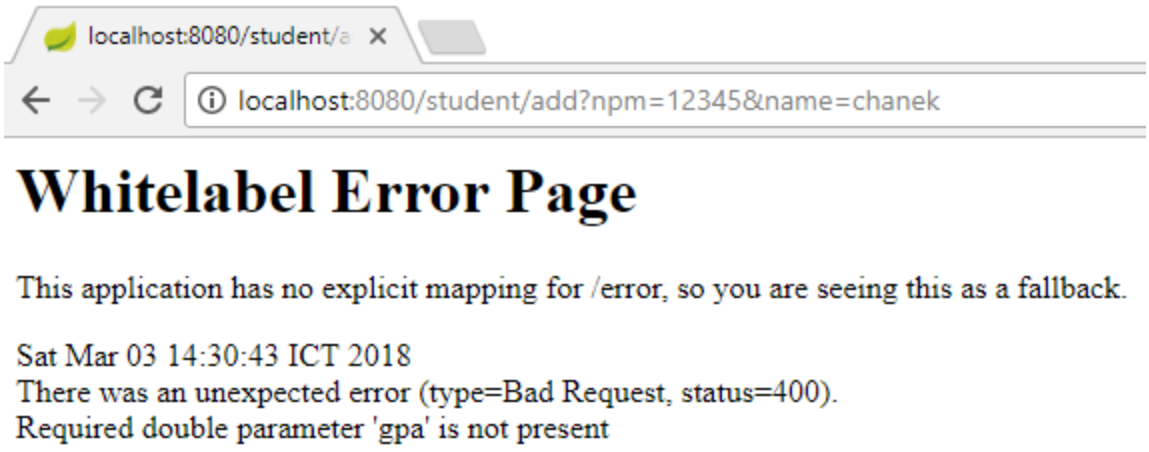
2. **Jawaban Tutorial**

   a. Fungsi Add

      1) 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: Berikut capture hasilnya.

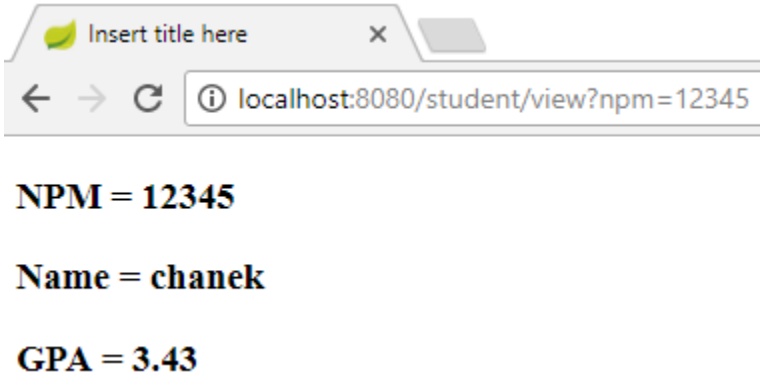


      2) Jalankan program dan buka
         localhost:8080/student/add?npm=12345&name=chanek
         Pertanyaan 2: apakah hasilnya? Jika error, tuliskan penjelasan Anda.
         Jawab: Berikut capture hasilnya.
         Terjadi error karena ketiga parameter yang diperlukan untuk dapat mengakses method add bersifat required (harus ada). Sedangkan pada url yang diakses hanya mengirimkan dua paremeter saja yaitu npm dan name, sedangkan parameter gpa tidak ada sehingga otomatis terjadi error.

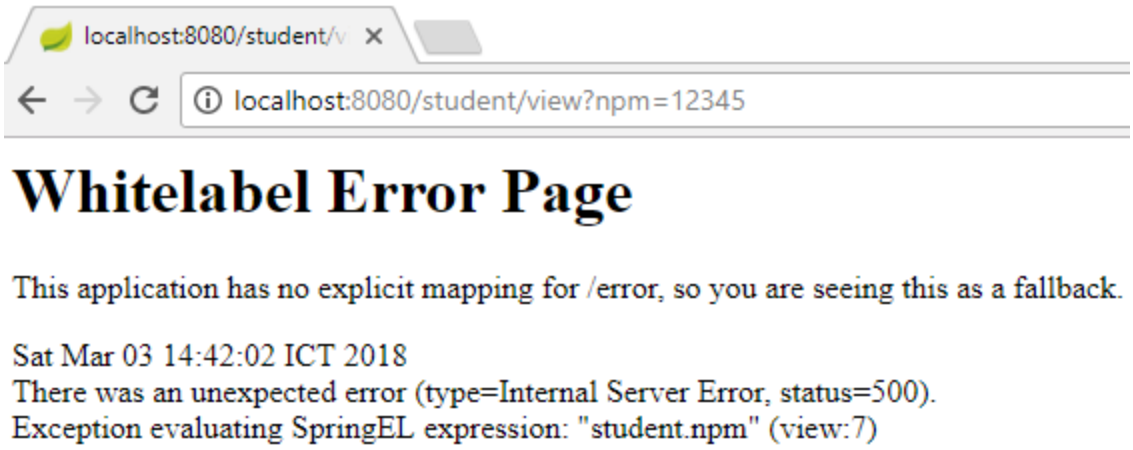


b. Fungsi View

1) 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 muncul. Berikut capture hasilnya.

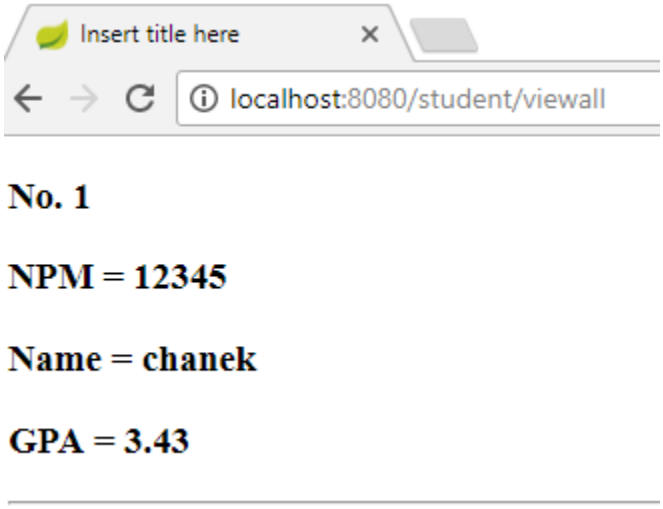


2) Coba matikan program dan jalankan kembali serta buka localhost:8080/student/view?npm=12345
Pertanyaan 4: apakah data Student tersebut muncul? Jika tidak, mengapa?
Jawab: Data student tidak muncul.
Hal ini dikarenakan memang tidak ada data student yang disimpan.
Ketika melakukan running code maka kondisi list student adalah kosing. Sehingga jika setelah running code dilakukan pemanggilan npm langsung maka data akan eror karena memang tidak ada data student yang dapat dikembalikan. Sebelum melakukan pemanggilan npm, seharusnya dilakukan proses add student tepat setelah klik running code.
Berikut capture hasilnya.

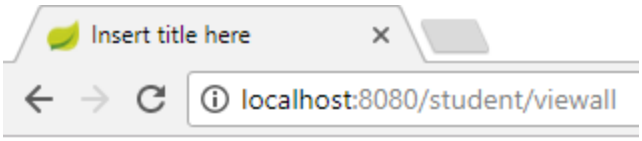


c. Fungsi ViewAll

1) 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: Data student muncul. Berikut adalah capture hasilnya.



2) Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka localhost:8080/student/viewall,
Pertanyaan 6: Apakah semua data Student muncul?
Jawab: Semua data student muncul. Berikut capture hasilnya.



No. 1

NPM = 12345

Name = chanek

GPA = 3.43

No. 2

NPM = 12745

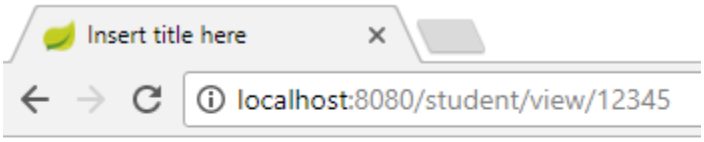
Name = selina

GPA = 3.6

---

d. Hasil Latihan

1) http://localhost:8080/student/view/12345

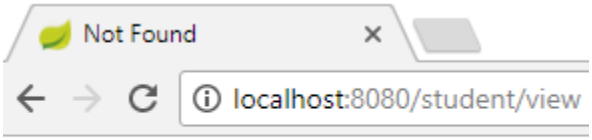


NPM = 12345

Name = chanek

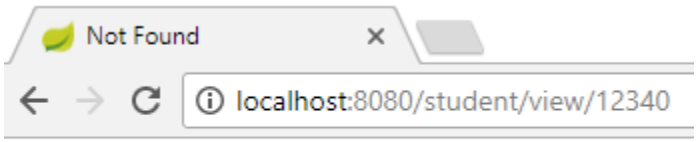
GPA = 3.43

2) http://localhost:8080/student/view



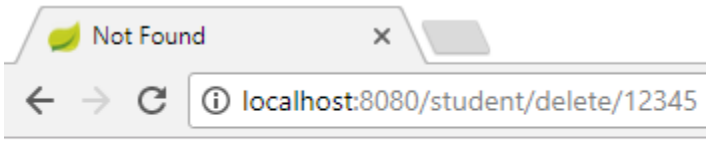
NPM Kosong

3) http://localhost:8080/student/view/12340



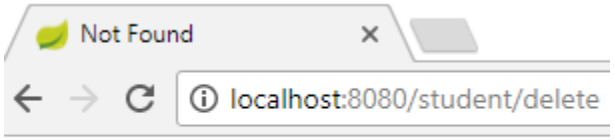
Data tidak ditemukan

4) http://localhost:8080/student/delete/12345



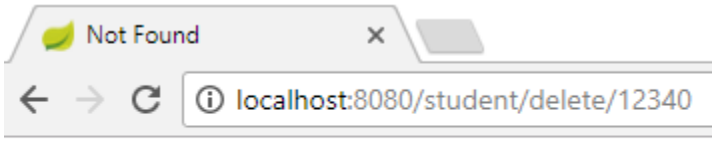
Data Berhasl Dihapus

5) http://localhost:8080/student/delete



NPM Kosong

6) http://localhost:8080/student/delete/12340



Data tidak ditemukan

3. Penjelasan method selectStudent

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

Pada tutorial terdapat fungsi untuk view student berdasarkan NPM. Oleh karen itu dibuatlah method select student pada InMemoryStudentService untuk melakukan pengecekan student.

Pada class InMemoryStudentService terdapat instance List yang berisi object StudentModel. List tersebut akan bertambah jika method addStudent dipanggil.

Kemudian List yang berisi object StudentModel tersebut dapat dipanggil untuk melakukan pengecekan apakah student dengan parameter npm ada pada dilist tersebut atau tidak. Pengecekannya adalah dengan menggunakan for dan dilakukan pengecekan pada setiap index List apakah parameter npm tersebut ada dalam list atau tidak. Jika parameter nim yang diinputkan ada pada List maka akan mengembalikan data student (object StudentModel) dan jika tidak ada pada list maka mengembalikan object null.
Parameter nim mempunyai tipe data String (sama seperti tipe data nim pada object StudentModel).

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

Method selectStudent digunakan untuk melakukan view dengan menggunakan RequestParam maupun PathVariabel dan untuk melakukan delete. Kedua fitur tersebut sama-sama menggunakan nim sebagai parameter yang dikirimkan sehingga dapat menggunakan method selectStudent ini. Untuk yang menggunakan PathVariabel digunakan tipe data Optional String sehingga perlu dibuatkan pula method yang select student dengan tipe data Optional String juga demikian pula pada interfacenya agar dapat diakses. Isi dari method selectStudent dengan tipe data optional string sama denga isi dari method selectStudent dengan tipe data string(telah dijelaskan di atas).

4. Penjelasan fitur delete
   StudentService :

```
String deleteStudent(StudentModel student);
```

Pertama-pertama pada interface StudentService dilakukan penambahan method deleteStudent dengan parameter object StudentModel yang akan mengambalikan nilai String.

InMemoryStudentService :

```
@Override
public String deleteStudent(StudentModel student){
    int index = studentList.indexOf(student);
    studentList.remove(index);
    return "Data Berhasl Dihapus";
}
```

Kemudian untuk mengimplementasikan method yang sudah dideklarasikan pada interface StudentService maka pada class InMemoryStudentService dilakukan penambahan method

deleteStudent dengan parameter dan nilai yang dikembalikan sama dengan yang ada pada interface.

Pada class inilah dilakukan penghapusan object studentModel dari List. Pertama-tama dilakukan pencarian index dari object studentModel. Setelah mendapatkan index-nya maka dilakukan penghapus object studentModel tersebut dari List. Method mengembalikan string bahwa proses penghapusan data berhasil dilakukan.

StudentController.

```
@RequestMapping(value = {"/student/delete", "student/delete/{npm}"})
public String delete(@PathVariable Optional<String> npm, Model model)
{
    if (npm.isPresent()){
        StudentModel student = studentService.selectStudent(npm.get());
        if (student == null) {
            model.addAttribute("message", "Data tidak ditemukan");
        } else {
            String message = studentService.deleteStudent(student);
            model.addAttribute("message", message);
        }
    } else{
        model.addAttribute("message", "NPM Kosong");
    }
    return "notfound";
}
```

Kemudian untuk dapat menjalankan methode deleteStudent maka dibuat method delete yang dapat diakses oleh user menggunakan url. Pada method ini digunakan PathVariabel dan bersifat optional sehingga parameter npm dapat diisikan atau tidak. Jika npm tidak diisikan maka akan mengembalikan message NPM Kosong pada halaman notfound.html. Jika npm diinputkan maka akan dilakukan pengecekan student dengan menggunakan method select student. Jika object student bernilai null maka akan mengembalikan message Data tidak ditemukan. Jika student ada maka akan mengembalikan message Data Berhasil Dihapus.