A. Ringkasan dari materi yang Saya pelajari pada tutorial ini

Pada tutorial 3 ini saya belajar bagaimana menggunakan Model dan Service pada Project Spring Boot. Model adalah sebuah objek yang merepresentasikan dan menyimpan informasi terhadap suatu hal, yang pada modelny adalah student, dimana ia menggambarkan bahwa student memiliki nama, npm dan GPA.

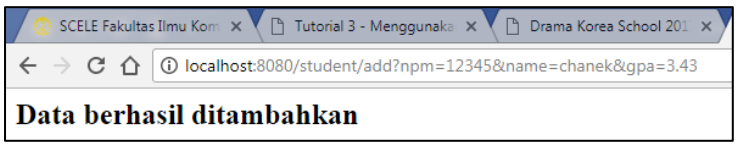
Service adalah suatu layer yang menjadi mediator antara controller dan database. Pada service layer, disimpan business logic yang digunakan untuk mengolah data yang terdapat dalam database. Service pada tutorial kali ini menggunakan interface StudentService.java yang diimplementasi oleh InMemoryStudentService.java. Namun pada tutorial kali ini, database belum digunakan, sehingga disini belum ada pengaturan untuk database, dan penyimpanan data dengan menggunakan arrayList.

B. Hasil Jawaban dari poin-poin Tutorial

Membuat Class Model

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

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

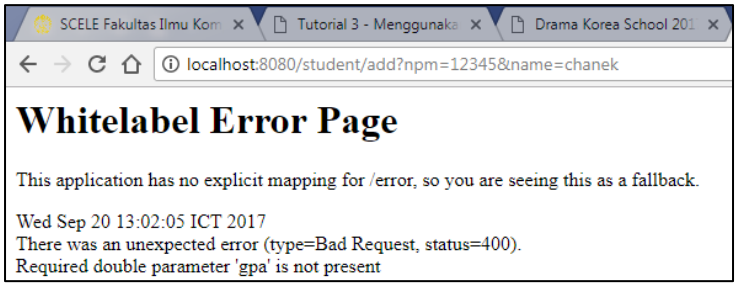


  Jawab :

  Tidak terjadi error. Data berhasil ditambahkan

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

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

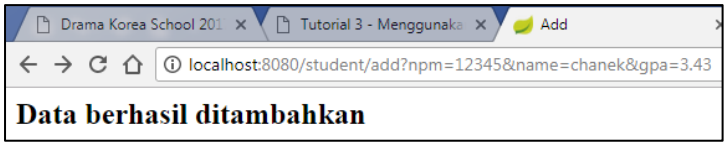


  Jawab :

  Ya, terjadi error yang disebabkan tidak diberinya parameter GPA pada link. Padahal parameter gpa required / harus untuk disertakan di link sebagai parameter, sehingga terjadilah error.

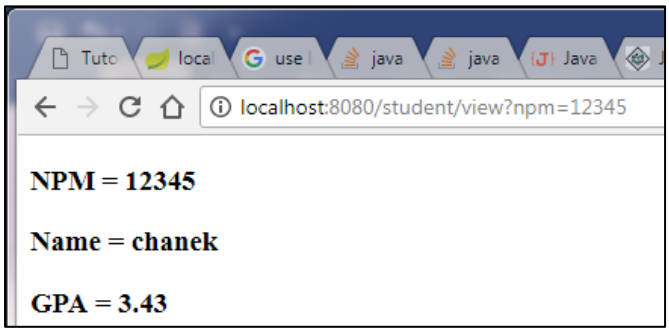
Method View by NPM

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



- localhost:8080/student/view?npm=12345

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

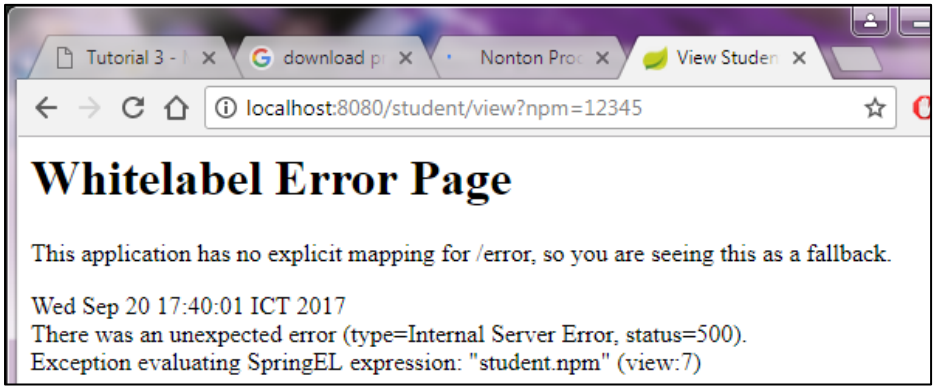


  Jawab :

  Data student tersebut muncul yaitu ada npm, nama dan gpa nya.

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

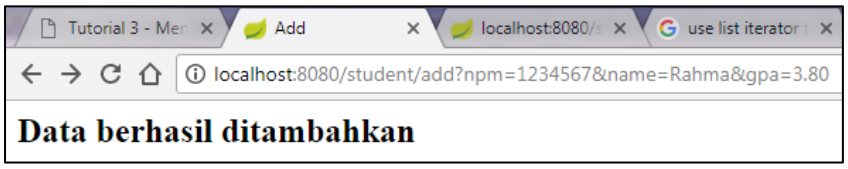
  Pertanyaan 4: apakah data Student tersebut muncul? Jika tidak, mengapa?



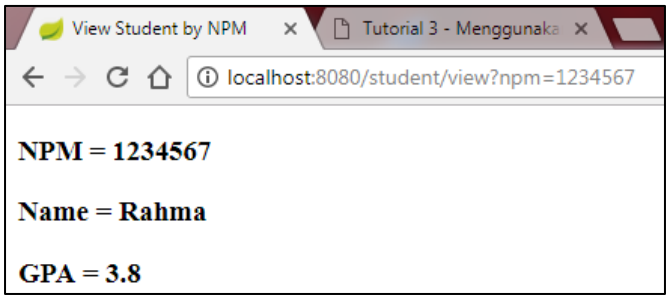
  Jawab :

  Tidak, data student tersebut tidak muncul. Karena ketika di start ulang maka program jalan dari awal, dan data List pun dari mulai kosong, sehingga ketika langsung diakses view?npm=12345 data tersebut belum ada karena belum di add dulu sebelumnya.

- Coba tambahkan data Student lainnya dengan NPM yang berbeda.

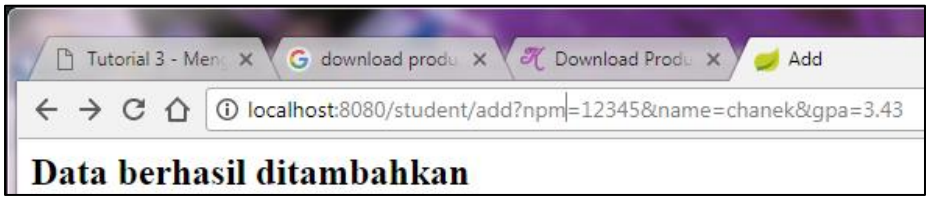


  View Data yang baru dimasukkan :



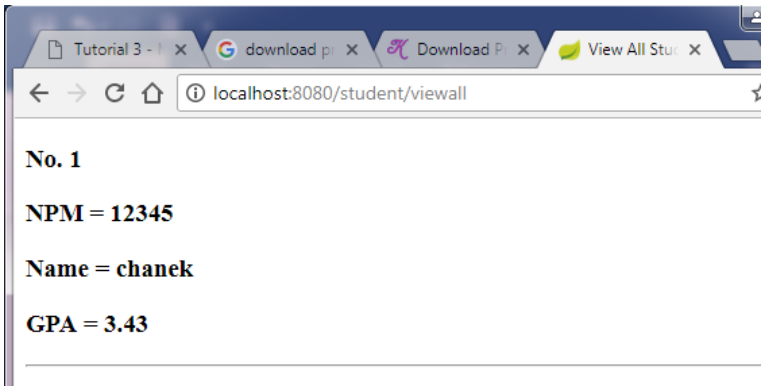
Method View All

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



lalu buka localhost:8080/student/viewall

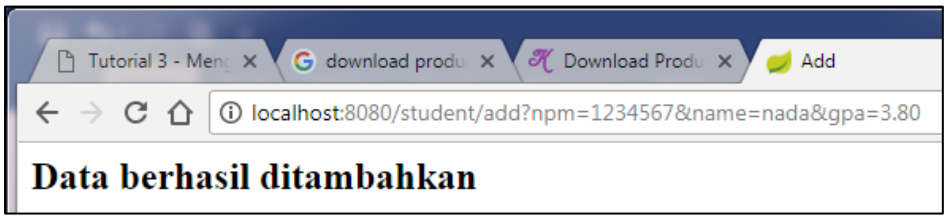
Pertanyaan 5: apakah data Student tersebut muncul?



Jawab :

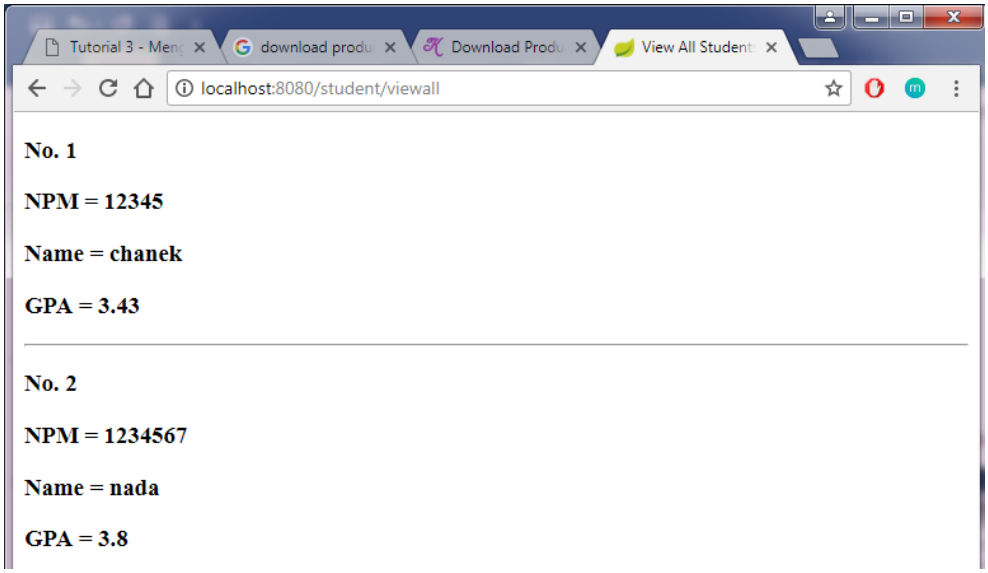
Ya, data student tersebut muncul.

- Coba tambahkan data Student lainnya dengan NPM yang berbeda



lalu buka localhost:8080/student/viewall

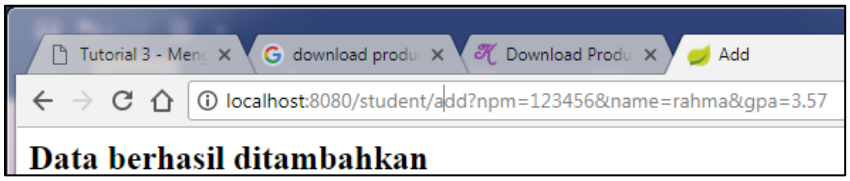
Pertanyaan 6: Apakah semua data Student muncul?

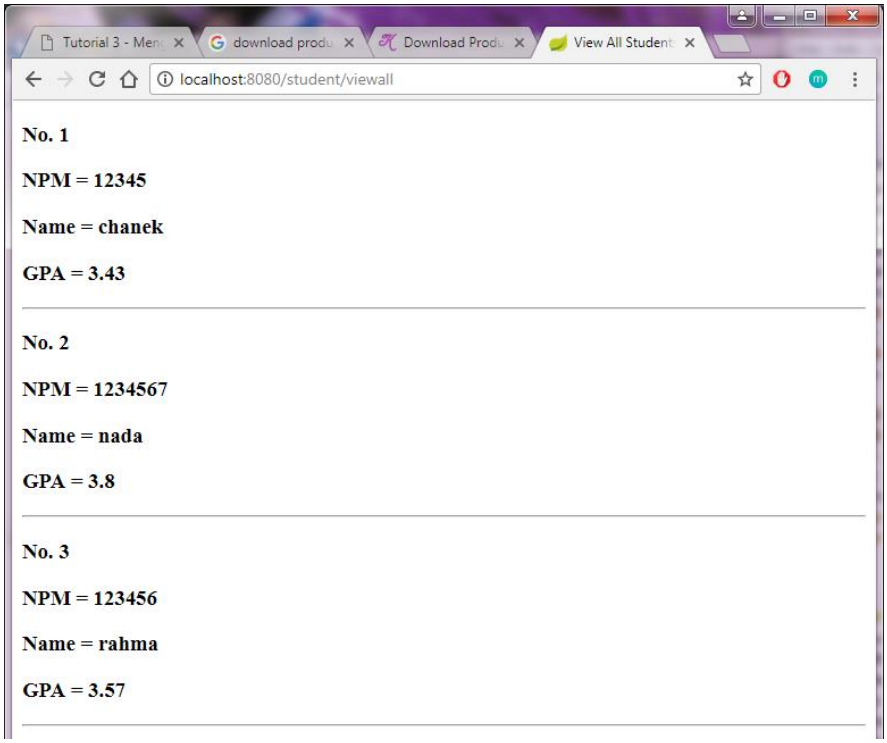


Jawab :

Ya, Semua data student muncul.

Berikut saya mencoba menambahkan beberapa lagi :





C. Method selectStudent yang Saya implementasikan

```
@Override
public StudentModel selectStudent(String npm) {
    // TODO Auto-generated method stub

    Iterator<StudentModel> itr = studentList.listIterator();
    while (itr.hasNext()) {
        StudentModel s = (StudentModel) itr.next();
       if (s.getNpm().equals(npm)) {
          return s;
       }
    }
    return null;
}
```

D. Penjelasan fitur delete yang Saya buat pada bagian latihan.

Untuk fitur delete ini saya mengedit pada beberapa file, yaitu pertama pada interface StudentService.java dilakukan penambahan method deleteStudent yang nantinya diimplementasikan pada InMemoryStudentService.

```
6
7  public interface StudentService {
8
9      StudentModel selectStudent(String npm);
10
11     List<StudentModel> selectAllStudents();
12
13     void addStudent(StudentModel student);
14
15     StudentModel deleteStudent(String npm);
16 }
17
```

Kedua, implementasi method deleteStudent pada file InMemoryStudentService.java :

```
@Override
public StudentModel deleteStudent(String npm) {
	// TODO Auto-generated method stub

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

Pada method deleteStudent ini, pertama akan dicari pada arrayList student yang memiliki npm seperti yg ada di parameter, lalu kalau ada akan di hapus dan di return array yang dihapus tsb untuk bisa ttp diakses datanya. Lalu kalau tidak ketemu maka akan dikembalikan null.

Ketiga yaitu mengatur routing pada StudentController.java

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

Pada routing tsb maka dilakukan pengecekan dahulu apakah npm diinput apa tidak pada link, kalau tidak dia akan menampilkan viewNoting, dan kalau iya maka akan di cek npm tsb sudah tersimpan atau blm, kalau tersimpan maka akan dilakukan pendeletan dan kalau ga ada maka menampilkan NPM tidak ditemukan.

E. Penjelasan Step-step pengerjaan latihan untuk Path Variabel yang Saya buat pada bagian latihan No. 1.

Untuk pengerjaan fitur view dengan menggunakan Path Variable saya mengedit pada file StudentController.java. yaitu seperti sebagai berikut :

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

Dimulai dengan mengecek apakah npm ada dalam link atau tidak, lalu npm ada di link dan dicari ternyata ada maka akan menuju halaman view yang menampilkan data-data student. Lalu jika npm tidak ada maka akan ke halaman viewGaKetemu dimana akan memberitahu bahwa NPM [no npm] tidak ditemukan. Dan ketika npm tidak dimasukkan kedalah link maka akan menuju halaman viewNothing yang menampilkan Bahwa NPM Kosong.

Berikut halaman viewNothing.html dan viewGaKetemu.html :