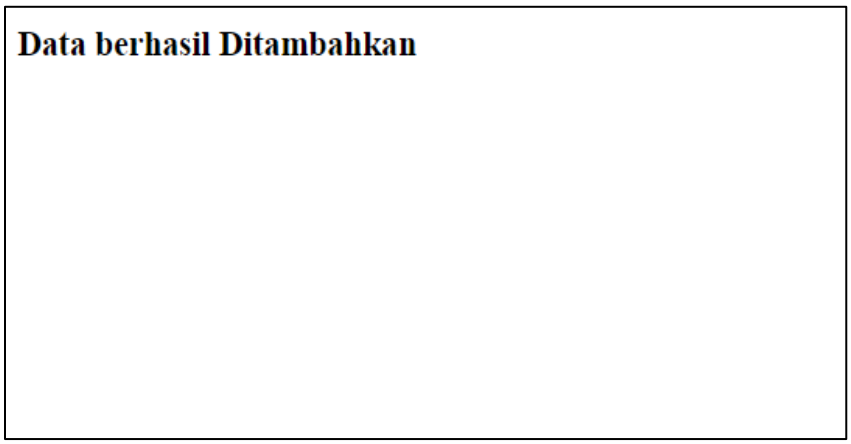
Membuat Controller dan Fungsi Add

Jalankan program dan buka

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

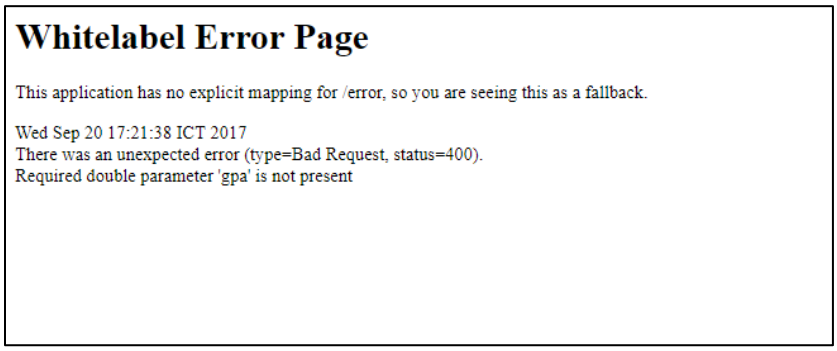
Pertanyaan 1 : apakah hasilnya? Jika error , tuliskan penjelasan Anda.



Tidak. Program menghasilkan sesuai pada file html add.html pada template

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

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



Error karena tidak terdapat parameter gpa (double) sementara pada controller diatur untuk harus memasukan parameter tersebut

Implementasikan method selectStudent!

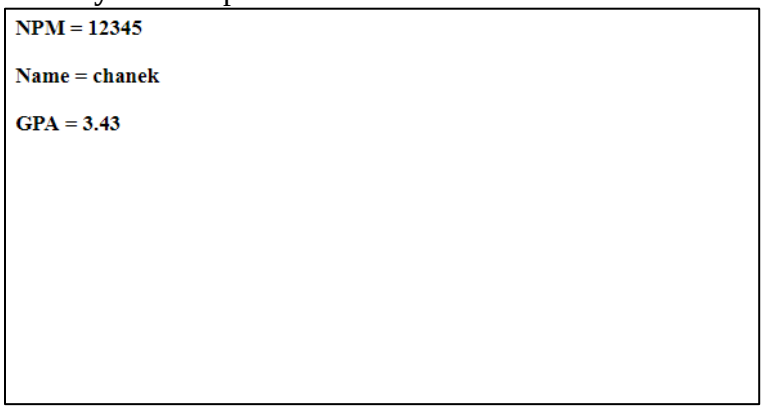
```
public StudentModel selectStudent(String npm) {

	int index=-1;
	for(int i=0;i<studentList.size();i++) {
		if(studentList.get(i).getNpm().equals(npm)) {
			index=i;
		}
	}
	if(index>-1) {
		return studentList.get(index);
	}else {
		return null;
	}
}
```

Method View By NPM

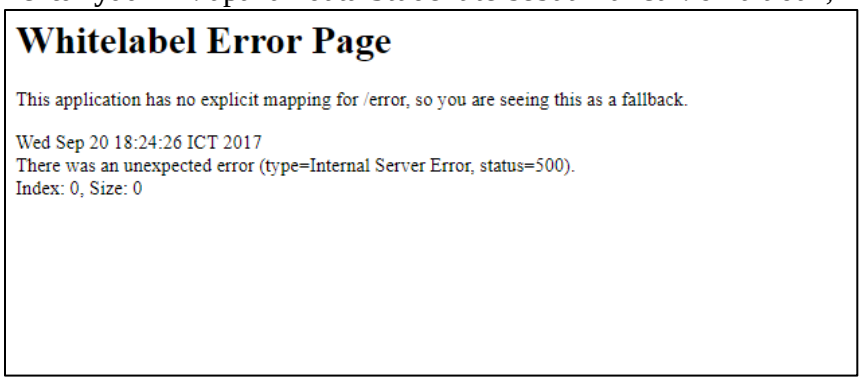
Jalankan program dan buka localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/view?npm=12345,

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



Data student berhasil ditampilkan

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 array list yang berisi data mahasiswa telah kosong setelah program diberhentikan

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

Method ViewAll

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?

Ya data student muncul

Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka localhost:8080/student/viewall,

Pertanyaan 6 : Apakah semua data Student muncul?

No. 1

NPM = 12345

Name = chanek

GPA = 3.43

No. 2

NPM = 12245

Name = messi

GPA = 4.0

Ya semua data yang dimasukan berhasil ditampilkan.

LATIHAN

1) **Membuat view menggunakan path variable**

   Langkah 1: Membuat fungsi di StudentController.java dengan request mapping sesuai soal

```
@RequestMapping(value={"student/view" , "student/view/{npm}" })
```

   Langkah 2 : Mengimplementasikan fungsi yang menerima parameter NPM (sebagai path variable). Kemudian memasukan kondisi jika parameter valid dan data ditemukan maka akan menambahkan atribut model berupa StudentModel yang dihasilkan dari method selectStudent. Kemudian akan me return template view. Jika NPM tidak ditemukan atau parameter kosong maka akan menambahkan model atribut berupa pesan error dan me return template 404

```
public String viewPath(@PathVariable Optional<String> npm , Model model ) {
	if(npm.isPresent ()&&studentService.isExist(npm.get())) {
		StudentModel student = studentService.selectStudent(npm.get());
		model.addAttribute("student", student);
		return "view";
	}else{
		model.addAttribute ( "message" , "NPM Kosong atau tidak
	ditemukan" );
		return "404";
	}
}
```

2) **Membuat fitur delete**

Langkah 1:

Menambah method isExist() dan deleteStudent() pada interface StudentService.java

Langkah 2:

Mengimplementasikan method isExist() dan deleteStudent() pada inMemoryStudentService.java .

IsExit() method

```
public boolean isExist(String npm) {
	boolean exist=false;
	for(int i=0;i<studentList.size();i++) {
		if(studentList.get(i).getNpm().equals(npm)) {
			exist=true;
			break;
		}
	}
	return exist;
}
```

DeleteStudent() method

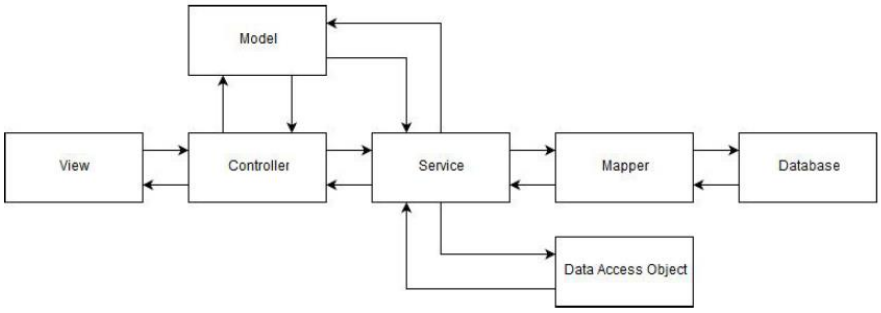
```
@Override
public void deleteStudent(String npm) {
	for(int i=0;i<studentList.size();i++) {
		if(studentList.get(i).getNpm().equals(npm)) {
			studentList.remove(i);
			break;
		}
	}
}
```

Langkah 3: Implementasi fungsi pada controller dengan request mapping menggunakan path variable

```
@RequestMapping(value={"student/delete" , "student/delete/{npm}" })
public String deletePath(@PathVariable Optional<String> npm , Model model )
{
	if(npm.isPresent()&&studentService.isExist(npm.get())) {
		studentService.deleteStudent(npm.get());
		model.addAttribute("message","Data berhasil dihapus");
	}else {
		model.addAttribute("message","Data gagal dihapus. NPM kosong
	atau tidak ditemukan");
	}
	return "delete";
}
```

Lesson Learned:

Membuat aplikasi menggunakan Java Springboot menerapkan konsep layering. Layering tersebut membantu programmer untuk membagi masing-masing fungsi. Dalam tutorial ini saya mempelajari layer model, service, controller dan view. Layer view berfungsi untuk melakukan *rendering* tampilan, layer *controller* berfungsi melakukan *handling request.* Model dan service berkaitan dengan data yang akan diolah.



Gambar alur interaksi data antar komponen pada layer Springboot MVC