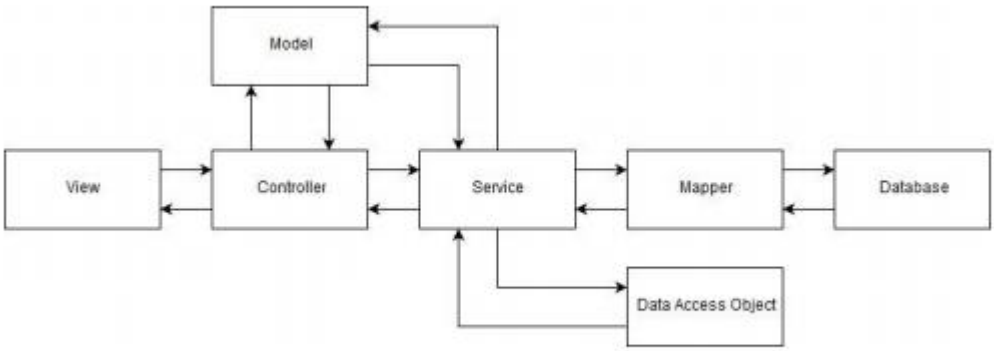
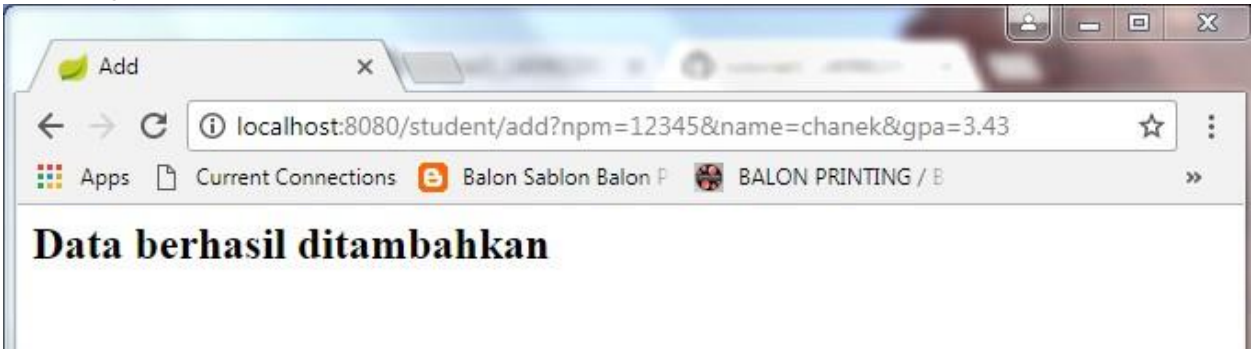
Ringkasan Materi :

Dengan mengerjakan tutorial 3 ini, saya memperoleh pengetahuan tentang MVC yaitu model view dan controller. Pada tutorial ini kita diharuskan menggunakan 3 buah package berbeda yaitu Model View dan Controller yang isinya masing-masing punya perannya. Selain itu saya mendapatkan pengetahuan mengenai penggunaan Interface dalam konsep MVC ini. Selain itu juga penggunaan ArrayList dan List dalam penyimpanan database sementara dalam aplikasi.



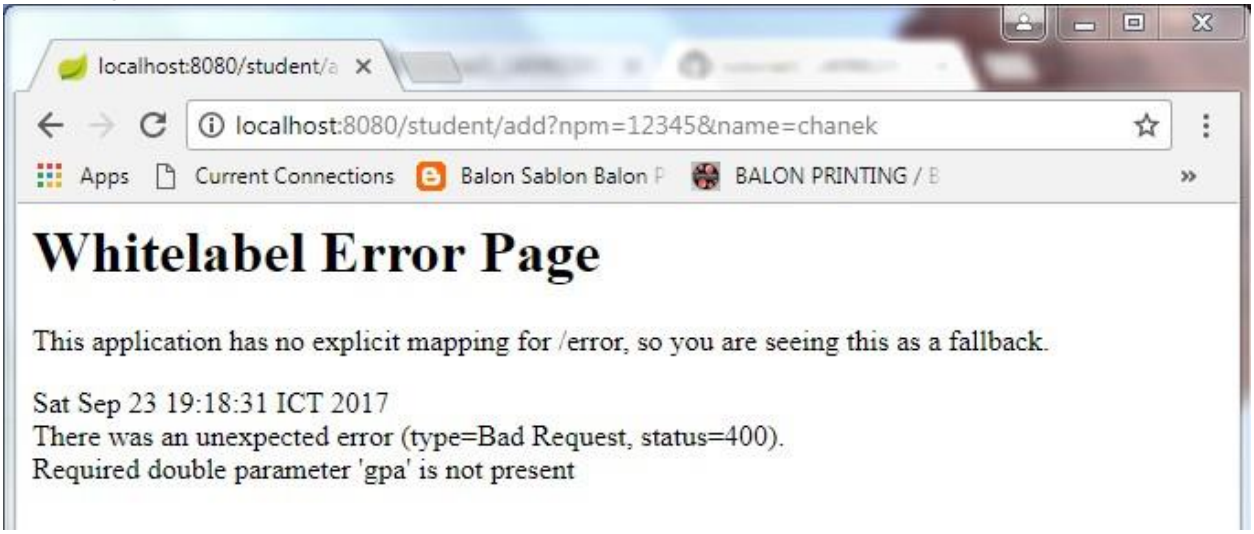
* tanda panah merepresentasikan alur interaksi dan data antar komponen

Pertanyaan 1:



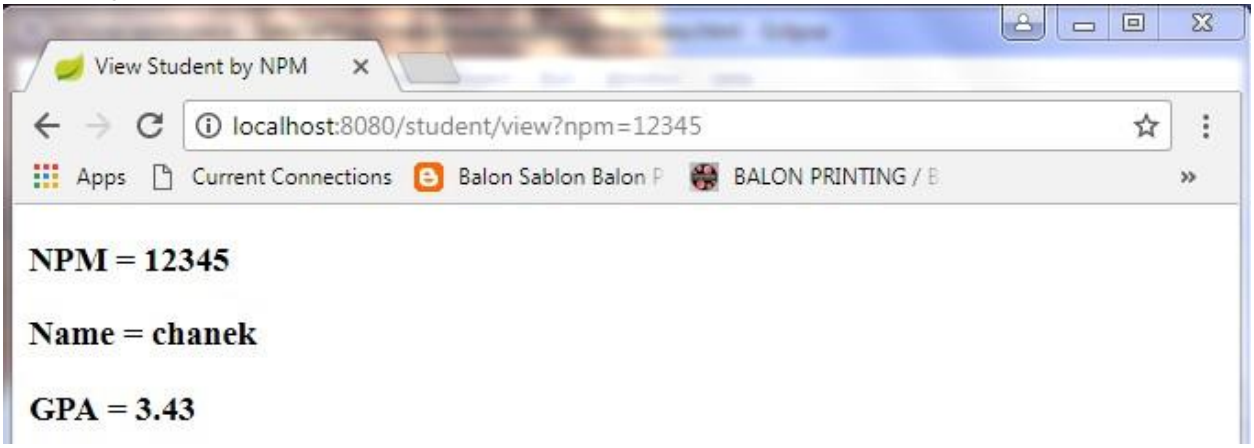
Hasilnya sebuah student yang bernama Chanek dengan NPM 12345 dan GPA 3,43 dimasukkan ke database berbentuk StudentList.

Pertanyaan 2:



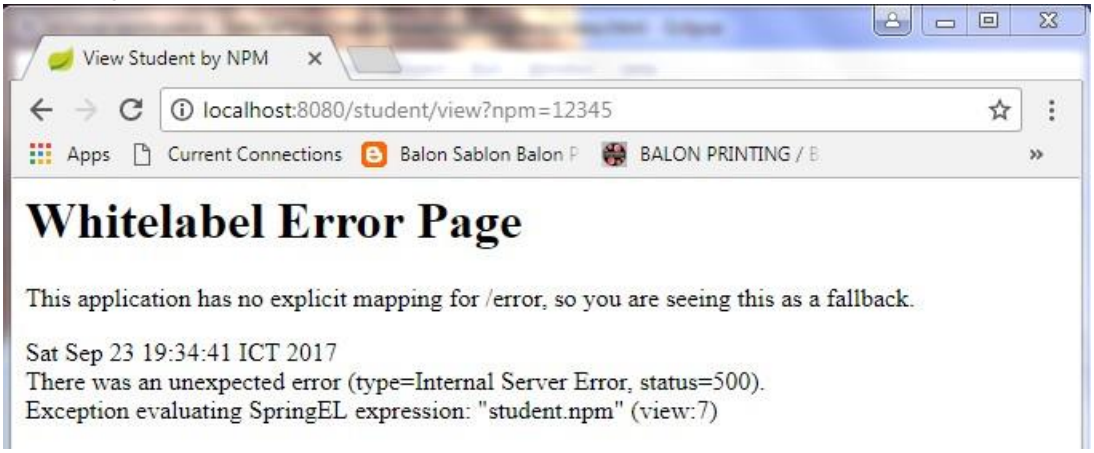
Hasilnya adalah error yang terjadi karena value gpa tidak diikutsertakan dalam URL Address, sehingga parameter gpa yang di request dan required secara true tidak terpenuhi.

Pertanyaan 3:



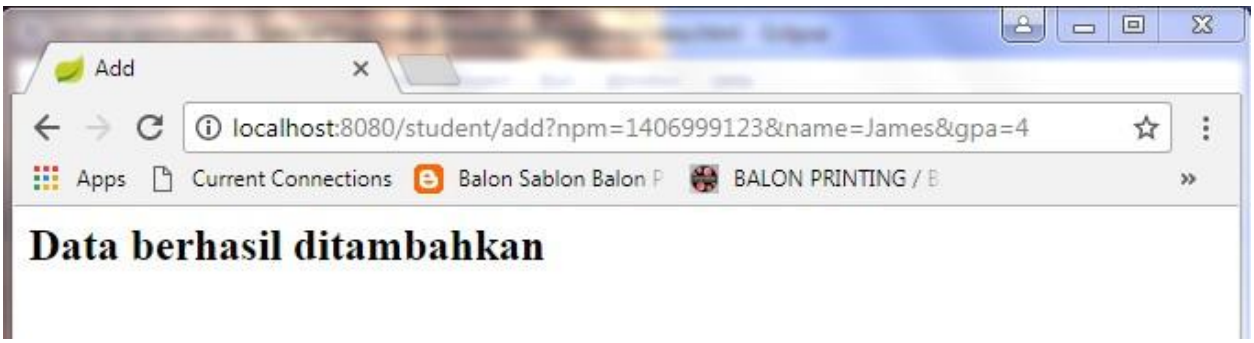
Muncul, karena sebelumnya telah dilakukan add studeng dengan npm 12345

Pertanyaan 4:

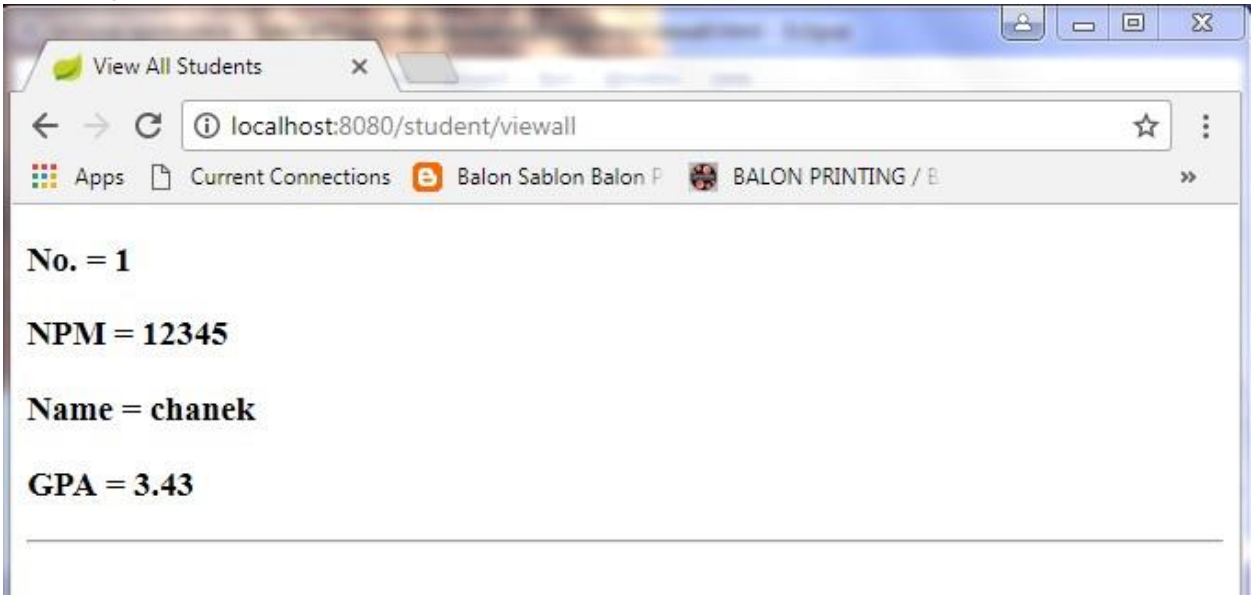


Tidak muncul. Karena setelah program dimatikan dan dijalankan kembali database StudentList juga telah di reset sehingga tidak ada student yang disimpan.

Coba tambahkan data student lainnya dengan NPM yang berbeda

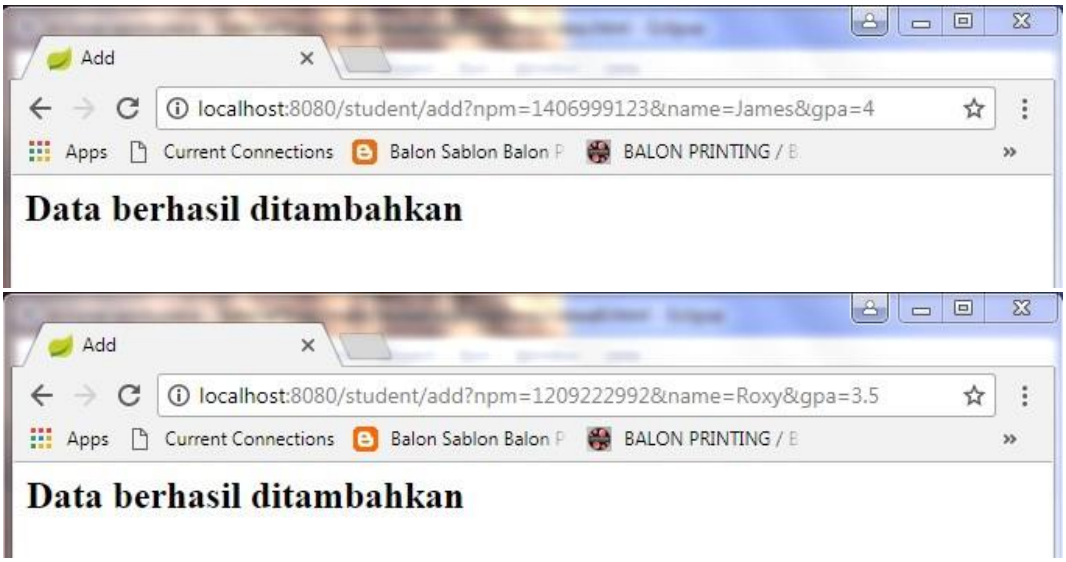


Pertanyaan 5:

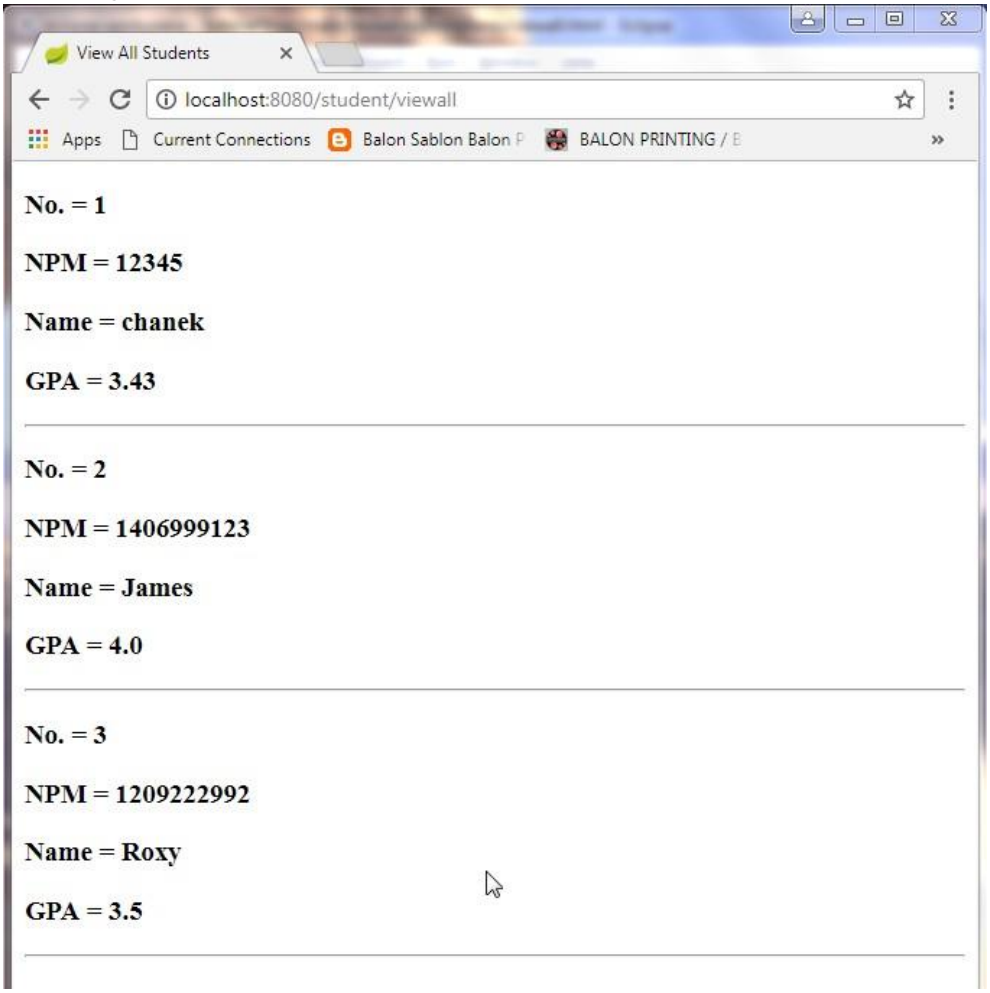


Muncul, karena sebelumnya telah di add student dengan NPM 12345 bernama Chanek dengan GPA 3,43. Ia adalah student No. 1 yang telah di add ke dalam database StudentList

Saya mencoba menambahkan 2 data dengan NPM yang berbeda lagi



Pertanyaan 6:



Muncul semua sesuai 2 data tambahan yang telah saya add.

Method Select Student

```
@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 method tersebut membutuhkan parameter npm. Di dalamnya terdapat for loop yang berfungsi untuk mencari student dengan npm yang diperoleh dari parameter. Ketika menemukannya maka akan direturn.

Latihan:

1. View Student via Path Variable by NPM

StudentController.java

```
@RequestMapping("/student/view/{npm}")
public String viewNpm(Model model, @PathVariable String npm) {

	StudentModel student = studentService.selectStudent(npm);
	if(student != null) {
		model.addAttribute("student", student);
		return "view";
	}
	else {
		model.addAttribute("npm", npm);
		return "viewerror";
	}
}
```

Viewerror.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Error</title>
</head>
<body>
    <h2 th:text="'NPM ' + ${npm} + ' tidak ditemukan'">NPM error</h2>
</body>
</html>
```

Terdapat 2 buah tambahan kode untuk merealisasikan fitur view student by NPM. Pertama yaitu method **viewNPM** yang direquest mapping ke **/student/view/{npm}** dimana npm tersebut dijadikan **@Path Variable** ke **String npm**. Ketika menjalankan method **selectStudent(npm)** maka dicari student dengan npm yang dicari, jika ada maka controller akan mereturn **view.html** dengan sebuah model student, jika tidak ada student dengan npm tersebut maka controller akan mereturn **viewerror.html** bersama model yang mengembalikan npm. Pada **viewerror.html** memanggil atribut npm yang telah dikirim melalui model dan **viewerror.html** akan mencetak teks berupa **"NPM <npm> tidak ditemukan"**

2. Delete Student by NPM

StudentController.java

```
@RequestMapping("/student/delete/")
public String deleteTanpaNpm() {
	return "tanpanpm";
}

@RequestMapping("/student/delete/{npm}")
public String deleteNpm(Model model, @PathVariable String npm) {

	StudentModel student = studentService.selectStudent(npm);
	if(student != null) {
		studentService.deleteStudent(npm);
		model.addAttribute("npm", npm);
		return "delete";
	}
	else {
		model.addAttribute("npm", npm);
		return "deleteerror";
	}
}
```

StudentService.java

```
void deleteStudent(String npm);
```

InMemoryStudentService.java

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

}
```

Tanpanpm.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Delete error</title>
</head>
<body>
    <h2 th:text="'Tolong isi NPM di URL Address'">Isi Npm</h2>
</body>
</html>
```

Delete.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Deleted</title>
</head>
<body>
    <h2 th:text="'Student dengan NPM ' + ${npm} + ' telah dihapus'">NPM deleted</h2>
</body>
</html>
```

Deleteerror.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Delete error</title>
</head>
<body>
    <h2 th:text="'Student dengan NPM ' + ${npm} + ' tidak ditemukan'">NPM error</h2>
</body>
</html>
```

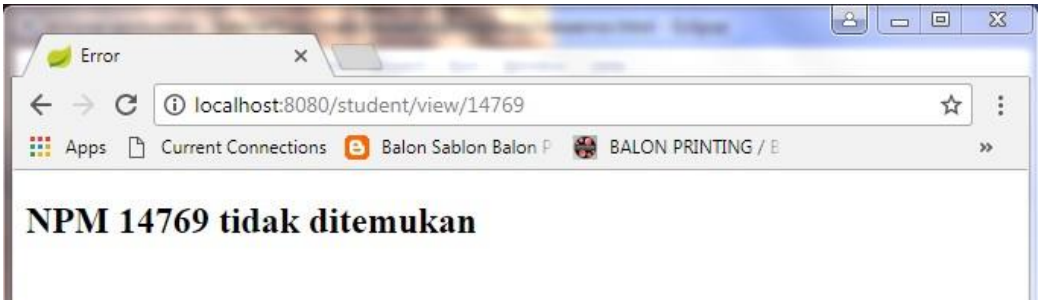
Terdapat tambahan kode pada **StudentController.java**, **StudentService.java**, dan **InMemoryStudentService.java**. Lalu juga terdapat 3 buah view (html) baru yaitu **tanpanpm.html**, **delete.html**, dan **deleteerror.html**.

Pada **InMemoryStudentService.java** terdapat **method deleteStudent** yang dioverride dari **StudentService interface**. Method tersebut menerima **parameter String npm**. Di dalamnya berisi for loop yang melakukan pencarian student dengan npm yang diambil dari parameter. Jika terdapat student dengan npm tersebut maka student tersebut dihapus dari database **StudentList**. Pada **StudentService.java** terdapat **void deleteStudent(String npm)** yang berguna

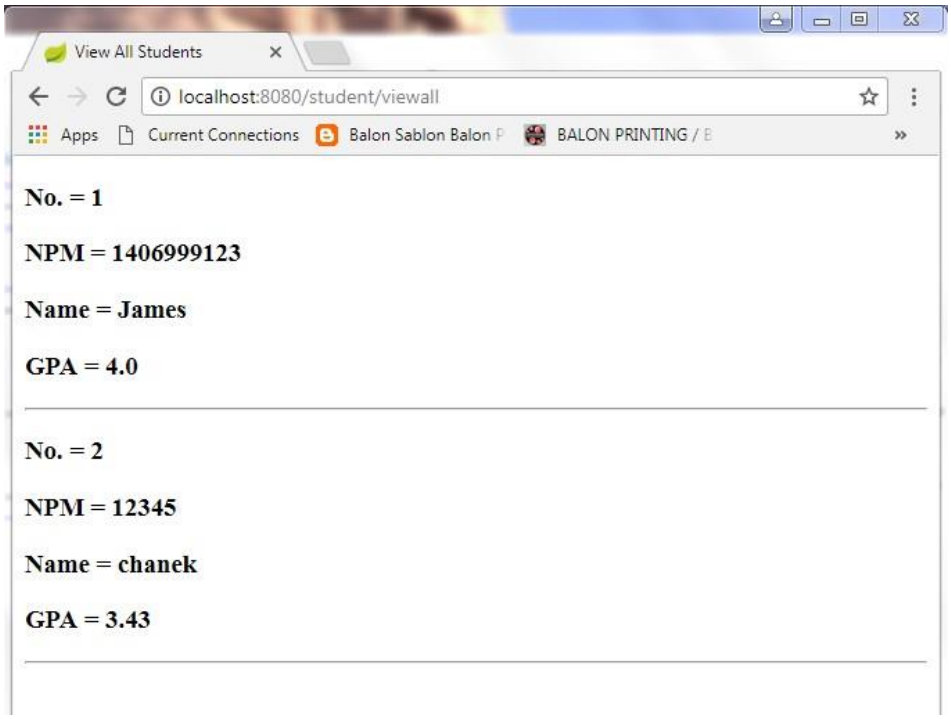
sebagai method yang dapat dipanggil oleh controller jika ingin memanggil lewat interface.

Pada **StudentController.java** terdapat 2 method. Method pertama yaitu **deleteTanpaNpm** yang direquest mapping ke **/student/delete/** yaitu method yang menghandle jika terjadi error ketika npm tidak diinput dalam URL address, sehingga controller mengembalikan sebuah view **tanpanpm.html** yang memberikan informasi bahwa untuk delete harus menaruh npm. Method kedua yaitu **deleteNPM(String npm)** yang di request mapping ke **/student/delete/{npm}** dimana npm dijadikan **@Path Variable** ke **String npm**. Ketika method tersebut dijalankan maka dicari student dengan npm tesebut, jika ada student dengan npm tersebut maka akan menjalankan method **deleteStudent(npm)** dan mengembalikan view berupa **delete.html** dengan mengirim model berupa npm. **Delete.html** yaitu view yang memberikan informasi bahwa **"Student dengan NPM <NPM> telah dihapus".** Jika tidak terdapat student dengan npm yang diinginkan maka akan controller akan mengembalikan view **deleteerror.html** dan model berupa npm. View tersebut akan memberikan informasi bahwa **"Student dengan NPM <NPM> tidak ditemukan"**

Percobaan 2 Fitur Tambahan



Ketika mencari student dengan NPM 14769 (Tidak ada dalam database StudentList)

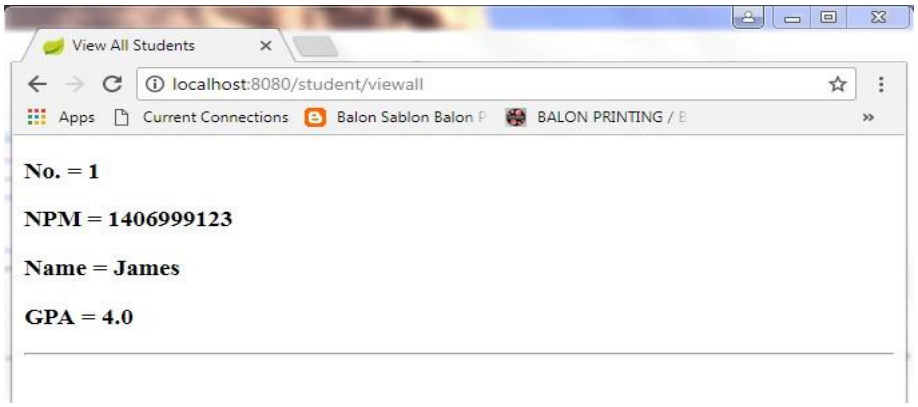


Isi sementara dari database StudentList

Ketika melakukan delete tapi tanpa npm (localhost:8080/student/delete)

Ketika melakukan delete student yang npmnya tidak ada di database StudentList

Ketika telah menghapus student dengan NPM 12345



Isi database setelah 1 student dihapus