Nama : Dina Febrianti Krisna **Writeup Tutorial 3 APAP**
NPM : 1506757421

RINGKASAN MATERI

Pada tutorial kali ini, saya telah mempelajari implementasi konsep *Model* dan *Service* menggunakan Spring Boot. *Model* merupakan sebuah objek yang merepresentasikan dan menyimpan informasi terhadap suatu hal. Tutorial mencontohkan *model* yaitu *object student* yang memiliki nama, NPM dan GPA. *Model* digunakan untuk merepresentasikan hal-hal tersebut dalam atribut yang dimiliki sebuah *Model*.

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*. Pengolahan ini meliputi kalkulasi data yang diambil dari *database*, manipulasi *user* input kedalam *database*, dan sebagainya. Pada tutorial ini, *service* yang diimplementasikan adalah melakukan manipulasi data *student* seperti menambahkan data, melakukan pencarian data *student* serta mengambil seluruh data *student*.

Selain itu, pada tutorial ini dibahas pula komponen *view* dan *controller* secara lebih mendalam serta kaitannya dengan komponen *model* dan *service*.

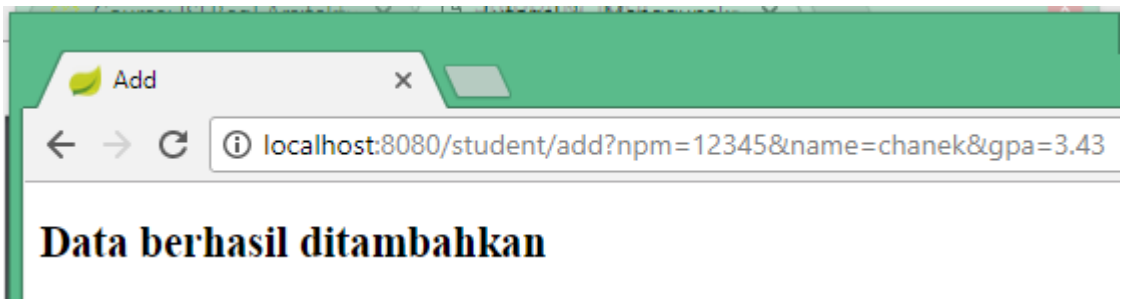
IMPLEMENTASI

Implementasi *Method selectStudent(String Model)* pada *Class InMemoryStudentService*

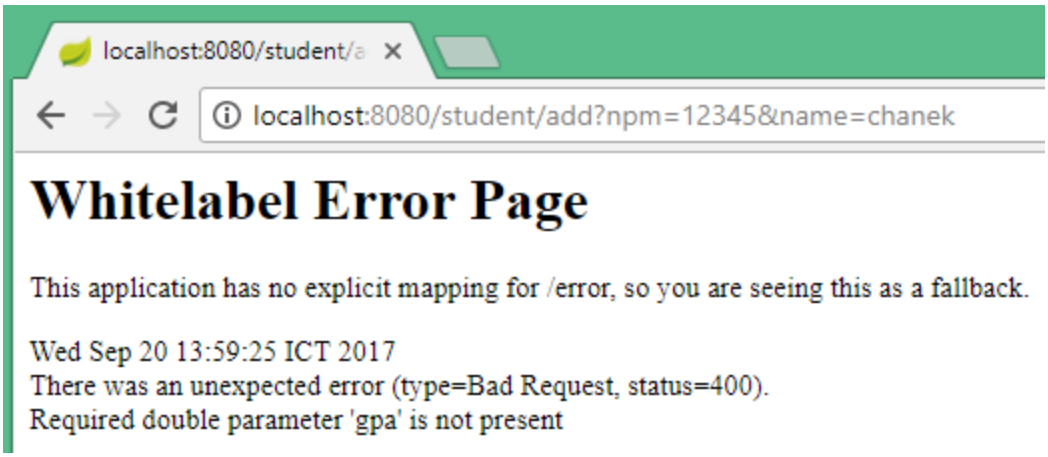
```
@Override
	public StudentModel selectStudent(String npm) {
		for(int i = 0; i < studentList.size(); i++) {
			if(studentList.get(i).getNpm().equals(npm)) {
				return studentList.get(i);
			}
		}
		return null;
	}
```

PERTANYAAN TUTORIAL

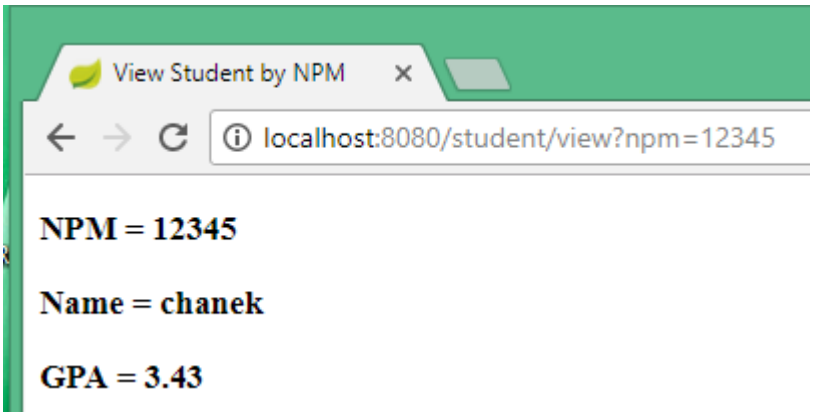
Jawaban 1 : Halaman localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 tidak *error*



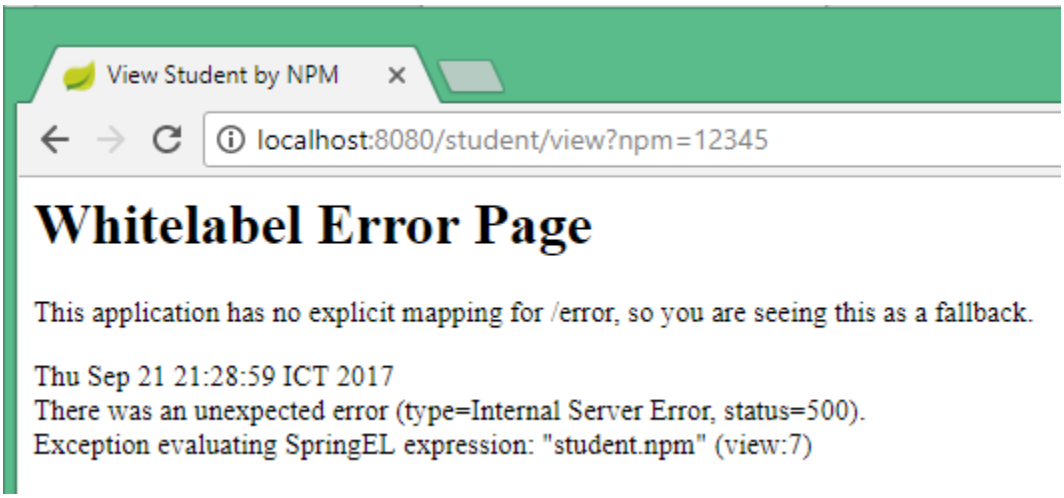
Jawaban 2 : Halaman localhost:8080/student/add?npm=12345&name=chanek *error* karena semua parameter pada halaman *add* (termasuk gpa) bertipe *required*.



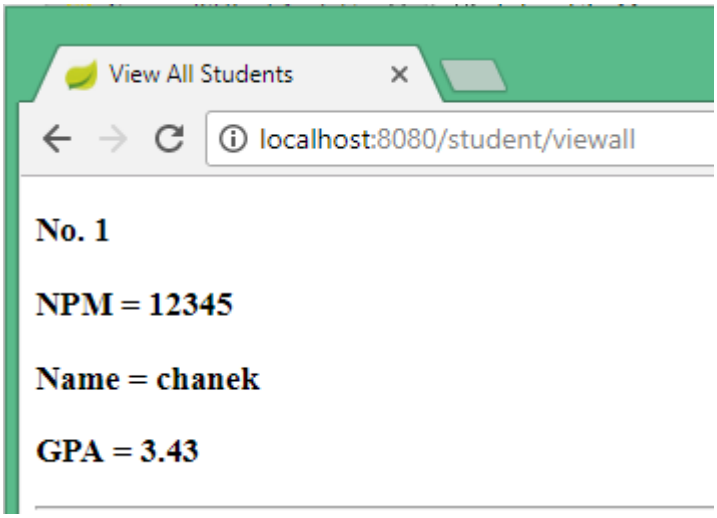
Jawaban 3 : Data *Student* muncul



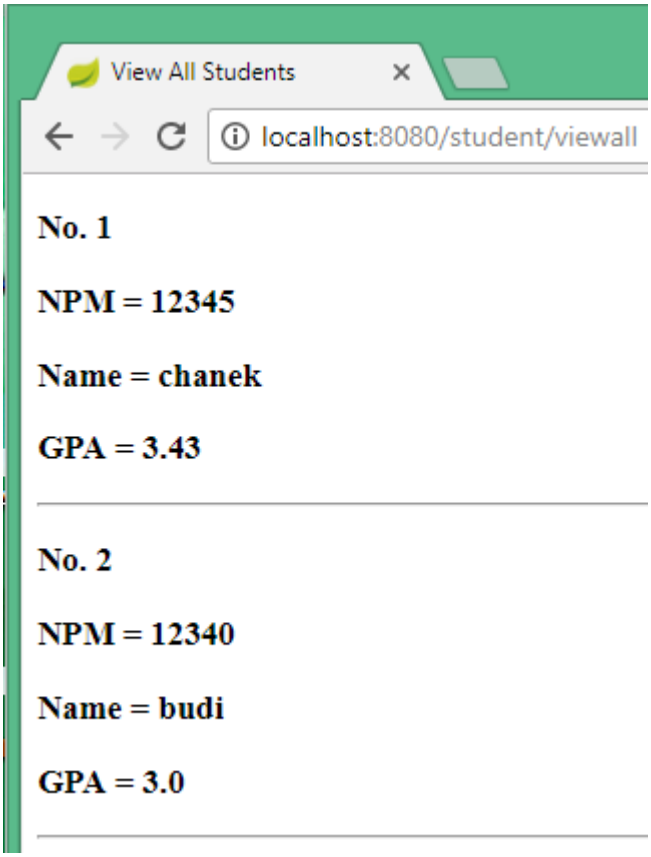
Jawaban 4: Data *student* tidak muncul dikarenakan *database* hanya menyimpan data ketika aplikasi sedang dijalankan. Ketika aplikasi di-*terminate* dan dijalankan kembali maka database baru akan terbentuk sehingga tidak dapat dilakukan pemanggilan data yang di-*input* pada penjalanan aplikasi sebelumnya



Pertanyaan 5 : Data *student* muncul



Pertanyaan 6 : Data *students* muncul



LATIHAN

1. ***Method View* dengan *Path Variable***
   a. Membuat *method viewPath* pada *class StudentController*, yang memiliki nilai *RequestMapping student/view* dan *student/view/*{npm}. Pada *method* ini akan dicek apakah

pengguna memasukkan *path* npm, apabila tidak maka pesan *error* dan *object studentModel* kosong akan di-*pass* ke halaman *view*. Setelah pengguna memasukkan npm, akan dicek apakah npm ada di dalam list *studentService*, apabila tidak maka pesan *error* dan *object studentModel* kosong akan di-*pass* ke halaman *view*. Apabila ada, *object student* akan di-*pass* ke halaman view.html

b. Menambahkan *tag* untuk pesan *error* pada *file* view.html

2. *Method Delete*

a. Menambahkan *method deleteStudent(String* npm*)* pada *interface StudentService* dan mengimplementasikannya pada *class InMemoryStudentService*

b. Mengimplementasikan *method deletePath(String* npm*)* pada *class StudentController* yang akan menerima *mapping* dari *path student/delete* dan *student/delete/*{npm}. Pada *method* ini akan dicek apakah pengguna memasukkan *path* npm, apabila tidak maka pesan *error* akan di-*pass* ke halaman *delete*. Setelah pengguna memasukkan npm, akan dicek apakah npm ada di dalam *list studentService*, apabila tidak maka pesan *error* akan di-*pass* ke halaman *view*. Apabila ada, *object student* akan dihapus menggunakan *method deleteStudent(String* npm*)* yang diimplementasikan pada bagian 2a dan pesan bahwa penghapusan telah berhasil akan di-*pass* ke halaman view.html

c. Membuat *file* delete.html dan *tag* yang akan digunakan untuk menerima *variable* yang di-*pass* oleh *method deletePath(String* npm*)* pada *class StudentController*