A. Ringkasan dari materi yang Anda telah pelajari pada tutorial kali ini

Pada tutorial kali ini, saya belajar lebih lanjut tentang implementasi konsep model-view-controller (MVC) menggunakan Spring Boot. Di tutorial kali ini lebih fokus mendalami model dan service. Model adalah objek yang merepresentasikan dan menyimpasn informasi terhadap suatu hal. Model digunakan untuk merepresentasikan hal-hal tersebut dalam atribut yang dimiliki sebuah model. Sedangkan service adalah suatu layer yang menjadi mediator antara controller dan database pada suatu program.

B. Hasil jawaban dari setiap poin pada bagian tutorial

1. Membuat Class Model

```
package com.example.tutorial3.model;

public class StudentModel {
    private String name;
    private String npm;
    private double gpa;

    public StudentModel(String npm, String name, double gpa) {
        this.name = name;
        this.npm = npm;
        this.gpa = gpa;
    }

    public String getNpm() {
        return npm;
    }

    public void setNpm(String npm) {
        this.npm = npm;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getGpa() {
        return gpa;
    }
}
```

2. Membuat Service
   - Membuat Interface StudentService.java

```java
package com.example.tutorial3.service;

import java.util.List;

import com.example.tutorial3.model.StudentModel;

public interface StudentService {
    StudentModel selectStudent(String npm);
    List<StudentModel> selectAllStudents();
    void addStudent(StudentModel student);

}
```

- Membuat class InMemoryStudentService.java yang meng-implements StudentService.java

```java
package com.example.tutorial3.service;

import java.util.List;

public class InMemoryStudentService implements StudentService {
    private static List<StudentModel> studentList = new ArrayList<StudentModel>();


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

            return student;
            }
        }
        return null;
    }

    @Override
    public List<StudentModel> selectAllStudents() {
        return studentList;
    }

    @Override
    public void addStudent(StudentModel student) {
        studentList.add(student);
    }


}
```

3. Membuat Controller dan Fungsi Add
   - Membuat Class StudentController

```
package com.example.tutorial3.controller;

import java.util.List;

@Controller
public class StudentController {
    private final StudentService studentService;

    public StudentController() {
        studentService = new InMemoryStudentService();
    }

    @RequestMapping("/student/add")
    public String add(@RequestParam(value = "npm", required = true) String npm,
            @RequestParam(value = "name", required = true) String name,
            @RequestParam(value = "gpa", required = true) double gpa) {
        StudentModel student = new StudentModel(npm,name,gpa);
        studentService.addStudent(student);
        return "add";
    }
```

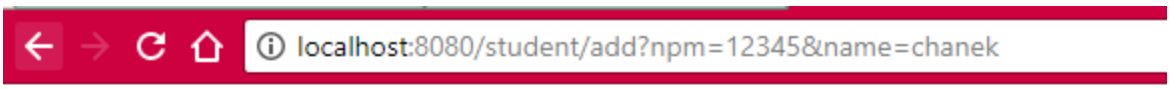
   - Membuat add.html pada directory resources/templates

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Add</title>
    </head>

    <body>
        <h2>Data berhasil ditambahkan</h2>
    </body>

</html>
```

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

Data berhasil ditambahkan

- Jalankan program dan buka localhost:8080/student/add?npm=12345&name=chanek. Apakah hasilnya?



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Sep 23 10:55:11 ICT 2017
There was an unexpected error (type=Bad Request, status=400).
Required double parameter 'gpa' is not present

Error, Karena pada url tidak ada gpa sedangkan gpa di define required

4. Method View by NPM
   - Pada class StudentController tambahkan method berikut

```
@RequestMapping("/student/view")
public String view(Model model, @RequestParam(value = "npm",
        required = true) String npm) {
    StudentModel student = studentService.selectStudent(npm);
    model.addAttribute("student", student);
    return "view";
}
```

   - Pada directory resources/templates tambahkan view.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Student by NPM</title>
    </head>

    <body>
        <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
        <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
        <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
    </body>
</html>
```

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

Data berhasil ditambahkan

NPM = 12345

Name = chanek

GPA = 3.43

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

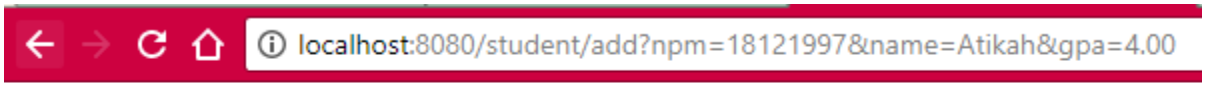
Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

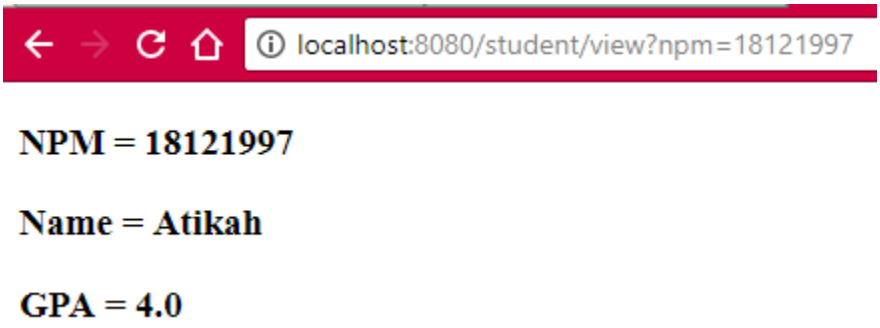
Sat Sep 23 11:08:08 ICT 2017
There was an unexpected error (type=Internal Server Error, status=500).
Exception evaluating SpringEL expression: "student.npm" (view:8)

Saat program dihentikan, semua data yang disimpan dalam arraylist akan terhapus. Apabila sebelumnya tidak ditambahkan terlebih dahulu, maka view tidak akan menampilkan apa-apa dan terjadi error.

- Coba tambahkan data Student lainnya dengan NPM yang berbeda



Data berhasil ditambahkan



5. Method View All
   - Pada class StudentController tambahkan method berikut

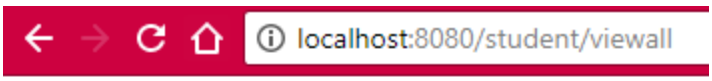
```
@RequestMapping("/student/viewall")
public String viewAll(Model model) {
    List<StudentModel> students = studentService.selectAllStudents();
    model.addAttribute("students", students);
    return "viewall";
}
```

   - Selanjutnya pada directory resources/templates tambahkan viewall.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View All Students</title>
    </head>

    <body>
    <div th:each="student,iterationStatus: ${students}">
        <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
        <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
        <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
        <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
        <hr/>
    </div>
    </body>
</html>
```

- Jalankan program dan buka localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/viewall, apakah data student tersebut muncul?

Data berhasil ditambahkan



No. 1

NPM = 12345

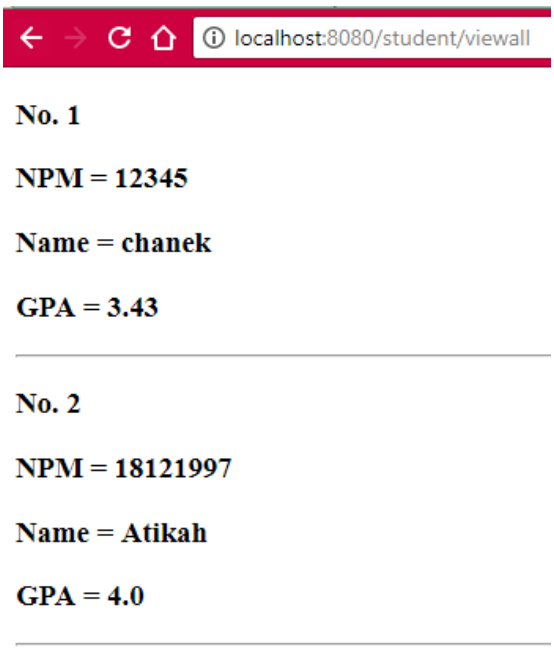
Name = chanek

GPA = 3.43

- Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka

Data berhasil ditambahkan

localhost:8080/student/viewall, Apakah semua data Student muncul?



C. Method selectStudent yang Anda implementasikan

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

        return student;
        }
    }
    return null;
}

@Override
public List<StudentModel> selectAllStudents() {
    return studentList;
}

@Override
public void addStudent(StudentModel student) {
    studentList.add(student);
}
```

D. Penjelasan fitur delete yang Anda buat pada bagian latihan.
Di method deleteStudent akan mengambil String npm yang dimasukkan pada URL memakai get.npm();. Method ini akan melakukan penghapusan jika npm yang dimasukkan (pada url yang ditulis) sama dengan npm yang berada di arraylist. Jika hasil npm sama maka akan menuju ke laman delete.html dan npm terhapus. Apabila npm tidak cocok dengan npm yang dimasukkan dengan arraylist maka di method selectStudent akan return null. Jika npm samadengan null maka akan menuju laman failedtodelete.html.