Muhammad Farhan Mardadi
1506757743

TUTORIAL 3

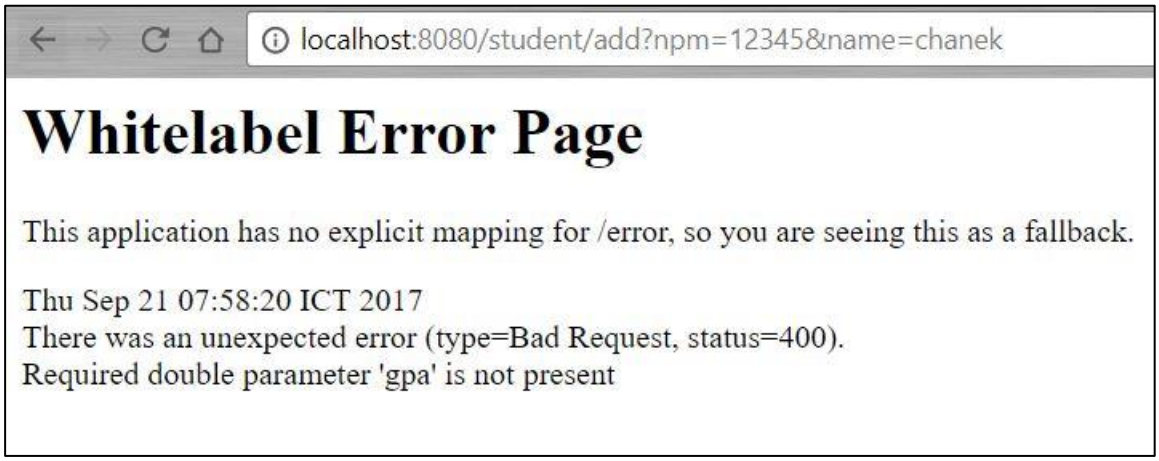
a. Ringkasan materi yang dipelajari

Yang saya pelajari dari latihan ini bagaimana memakai model dalam konsep MVC(Model-Controller-View). Jadi model merepresentasikan informasi dari sesuatu hal, pada contoh kali ini adalah mahasiswa, yang memiliki informasi nama, NPM dan IPK. Dan juga saya belajar mengenai service, service merupakan layer yang memediatori controller dan database, pada service layer disimpan business logic yang digunakan untuk mengolah data yang terdapat di dalam database.

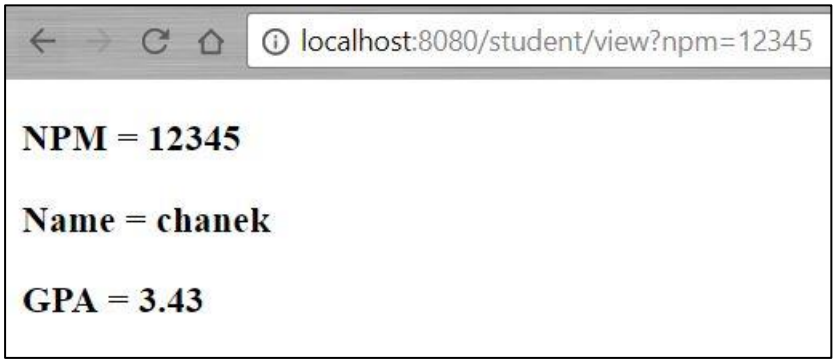
b. Jawaban tutorial

1. localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43
   Pertanyaan 1: apakah hasilnya? Jika error, tuliskan penjelasan Anda.
   Berhasil menambahkan data.

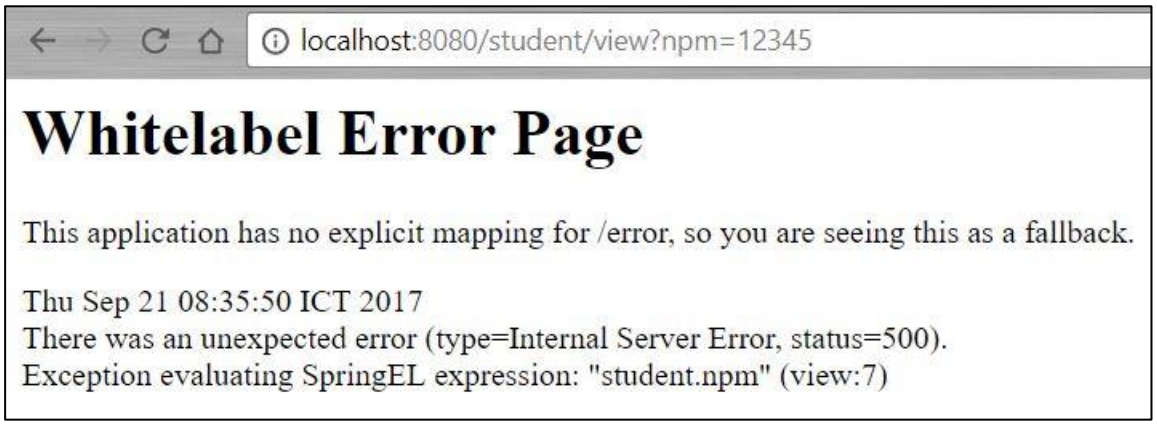
2. localhost:8080/student/add?npm=12345&name=chanek
   Pertanyaan 2: apakah hasilnya? Jika error, tuliskan penjelasan Anda.
   Error, karena tidak memberikan parameter gpa.



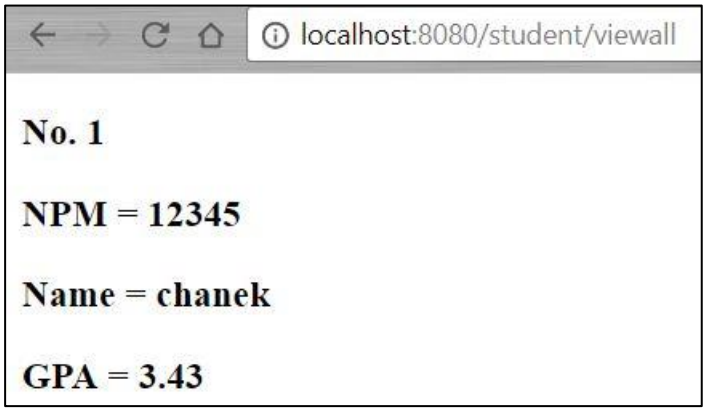
3. localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/view?npm=12345
   Pertanyaan 3: apakah data Student tersebut muncul? Jika tidak, mengapa?
   Iya, data student itu muncul.



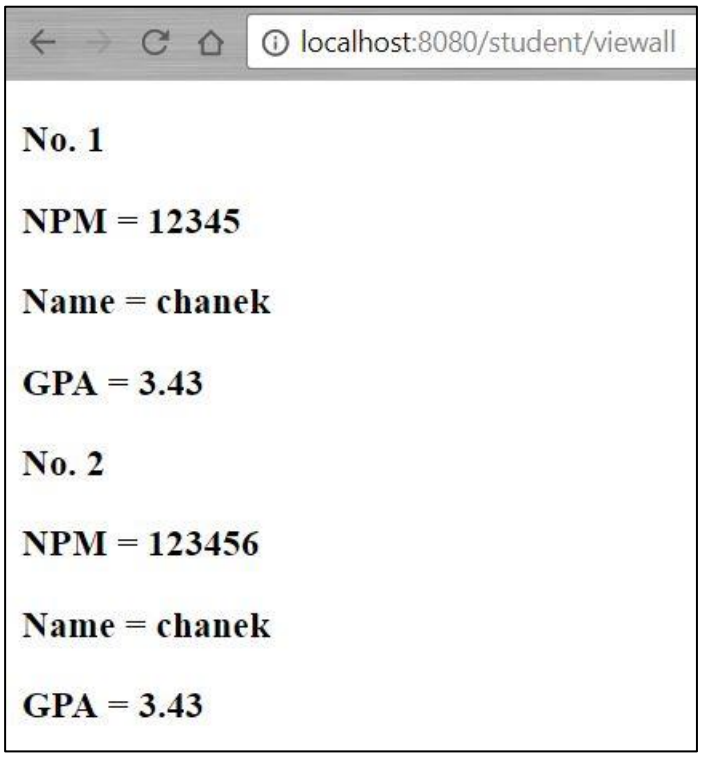
4. localhost:8080/student/view?npm=12345
   Pertanyaan 4: apakah data Student tersebut muncul? Jika tidak, mengapa?
   Tidak muncul, karena belum ada data yang ditambahkan sedangkan sudah ingin melihat data.



5. 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.



6. Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka localhost:8080/student/viewall,
   Pertanyaan 6: Apakah semua data Student muncul?
   Ya, semua data student muncul.



c. Method selectStudent yang diimplementasikan

```
@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;
}
```

Jadi untuk mengimplemetasikan method ini saya hanya melakukan loop sebanyak data student yang tersimpan di dalam arraylist, lalu apabila npm student sama dengan npm yang sedang kita cari makanya method akan mengembalikan object student tersebut, apabila tidak ditemukan maka akan mengembalikan nilai null.

d. Penjelasan fitur delete

Untuk membuat fitur delete ini saya menambahkan satu method pada interface StudentService agar setiap kelas yang mengimplements interface ini harus mengimplementasi method tersebut. Method ini mengembalikan boolean karena saya ingin mengetahui apakah penghapusan student berhasil atau tidak.

```
public interface StudentService {
    boolean deleteStudent(String npm);
```

Untuk mengimplementasiannya mirip dengan method selectStudent, yakni dengan melakukan looping sebanyak data mahasiswa yang ada, apabila maka data npm mahasiswa sama dengan npm yang dicari maka akan dilakukan penghapusan data mahasiswa tersebut, dan method akan menegembalikan nilai true. Apabila tidak terjadi penghapusan maka method akan mengembalikan nilai false.

```
@Override
public boolean deleteStudent(String npm) {
    boolean isDeleted = false;

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