

Muhammad Umar Farisi (1506725552)

Tutorial 3

PSP – C

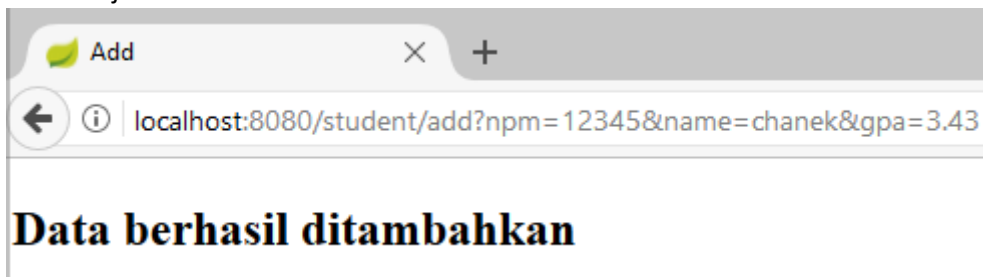
Ringkasan Materi

Pada tutorial ini mempelajari tentang MVC yang ada di Spring Boot. Pada MVC tersebut, terdapat kelas Service yang berfungsi sebagai jembatan antara *data access object* dengan kelas Controller. Selain itu, terdapat juga kelas Model yang berfungsi untuk merepresentasikan suatu objek yang akan disimpan di database. Tutorial kali ini juga mencakup bahan – bahan tutorial sebelumnya seperti PathVariable dan Controller.

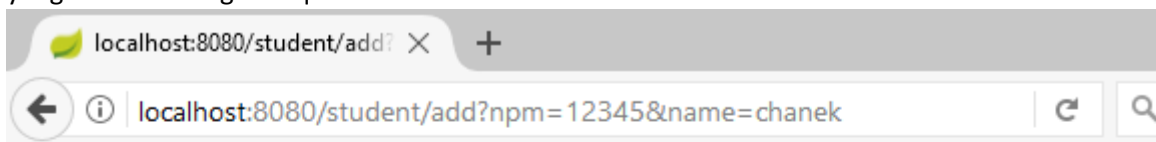
Tutorial

Jawaban Pertanyaan

- 1) Tidak terjadi error



- 2) Error karena tidak ada parameter gpa pada url tersebut, padahal parameter gpa bersifat wajib yang ditandai dengan required = true



Whitelabel Error Page

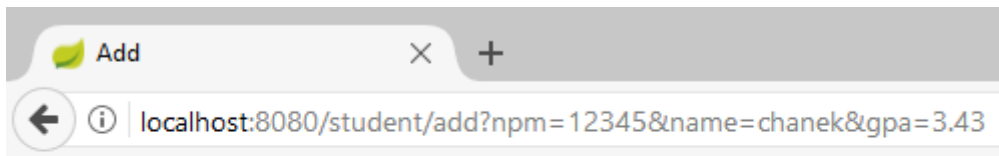
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Sep 20 16:36:54 ICT 2017

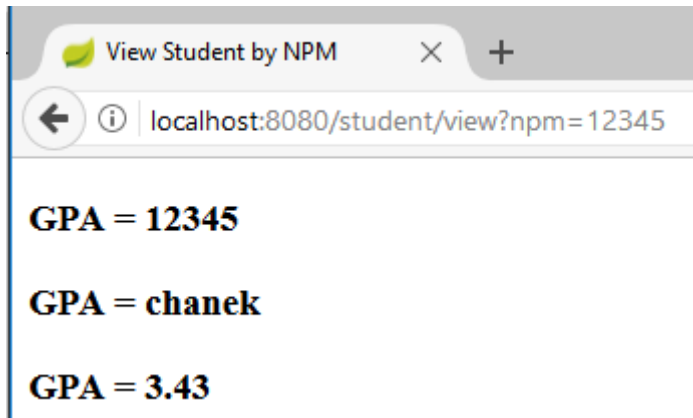
There was an unexpected error (type=Bad Request, status=400).

Required double parameter 'gpa' is not present

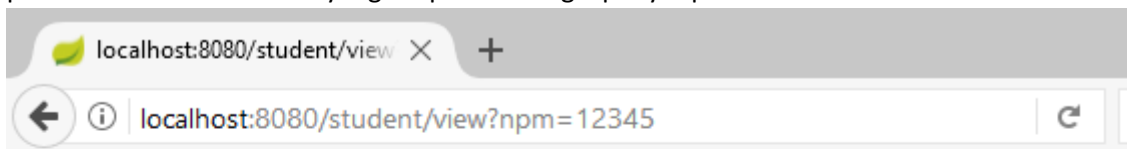
- 3) Iya Student tersebut muncul



Data berhasil ditambahkan



- 4) Student tersebut (npm 12345) tidak muncul karena dia belum ditambahkan ke objek studentList pada kelas StudentService yang berperan sebagai penyimpanan data Student tersebut.



Whitelabel Error Page

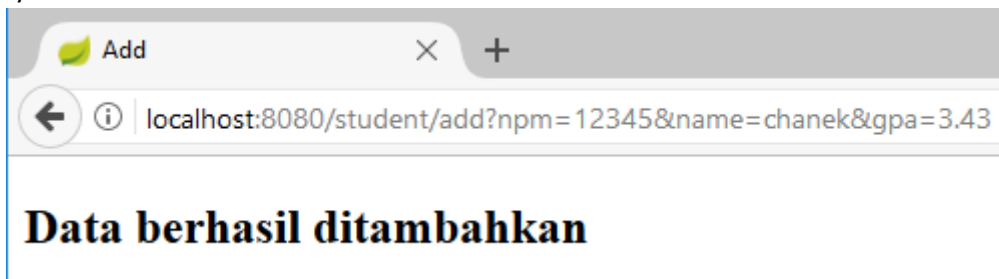
This application has no explicit mapping for /error, so you are seeing this as a fallback.

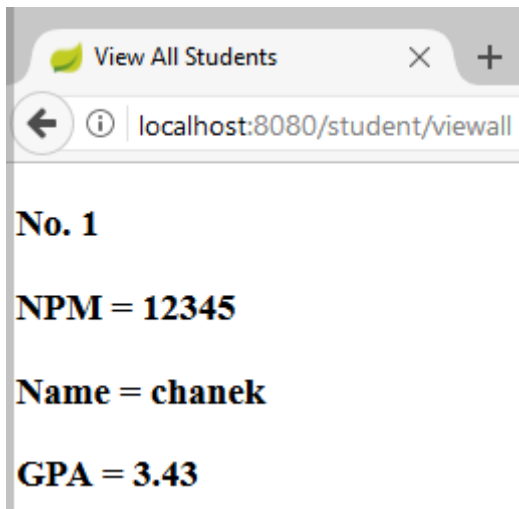
Wed Sep 20 16:43:12 ICT 2017

There was an unexpected error (type=Internal Server Error, status=500).

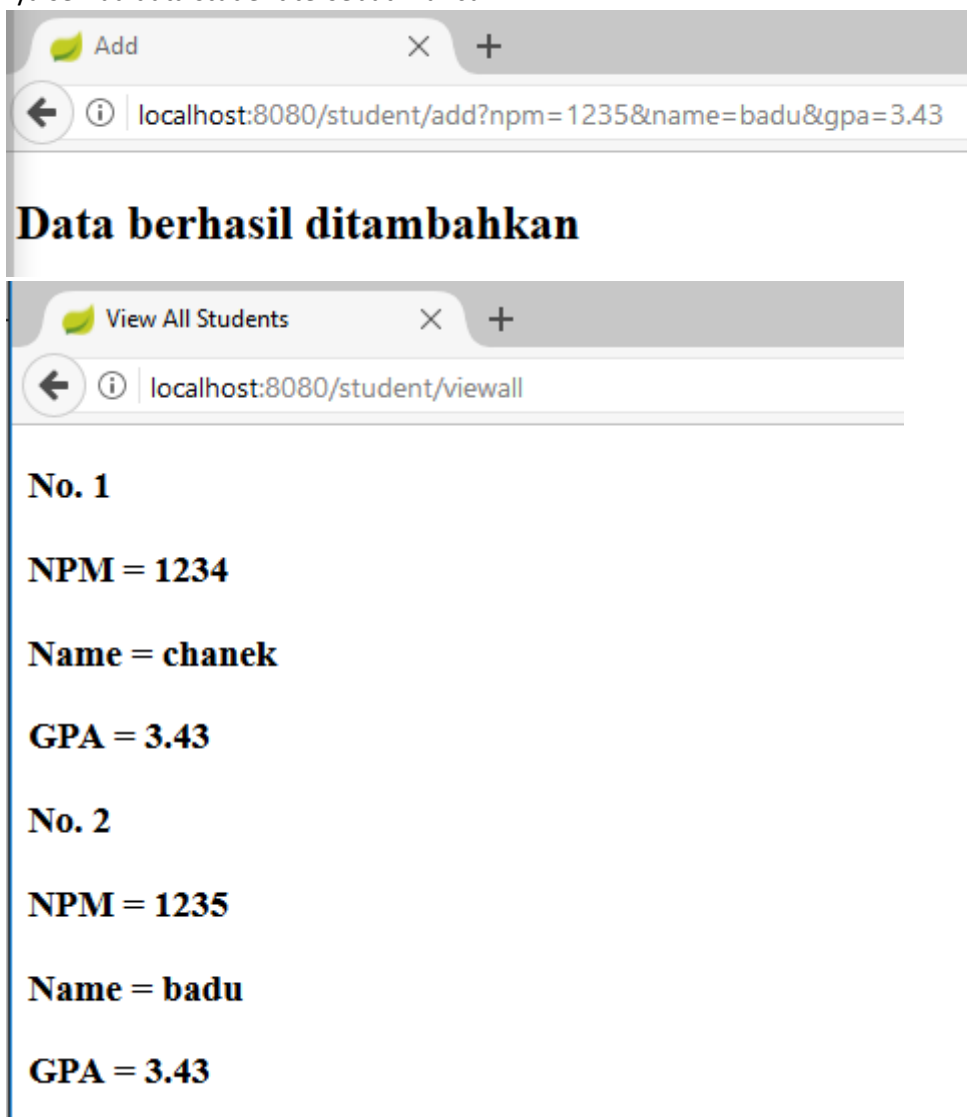
Exception evaluating SpringEL expression: "student.npm" (view:8)

- 5) Iya data student tersebut muncul





- 6) Iya semua data student tersebut muncul



Implementasi Method selectStudent

Implementasi method selectStudent:

```
public StudentModel selectStudent(String npm) {  
    StudentModel student = new StudentModel();  
    student.setNpm(npm);  
    int indexOfStudent = studentList.indexOf(student);  
    if(indexOfStudent != -1) student = studentList.get(indexOfStudent);  
    else student = null;  
    return student;  
}
```

note: saya implement method hashCode() dan method equals() pada class StudentModel yaitu sebagai berikut:

@Override

```
public int hashCode() {  
    final int prime = 31;  
    int result = 1;  
    result = prime * result + ((npm == null) ? 0 : npm.hashCode());  
    return result;  
}
```

@Override

```
public boolean equals(Object obj) {  
    if (this == obj)  
        return true;  
    if (obj == null)  
        return false;  
    if (getClass() != obj.getClass())  
        return false;  
    StudentModel other = (StudentModel) obj;  
    if (npm == null) {  
        if (other.npm != null)
```

```

        return false;
    } else if (!npm.equals(other.npm))
        return false;
    return true;
}

```

Tahap – Tahap pada Latihan

1. Tahap – tahap pengerjaan latihan nomor 1

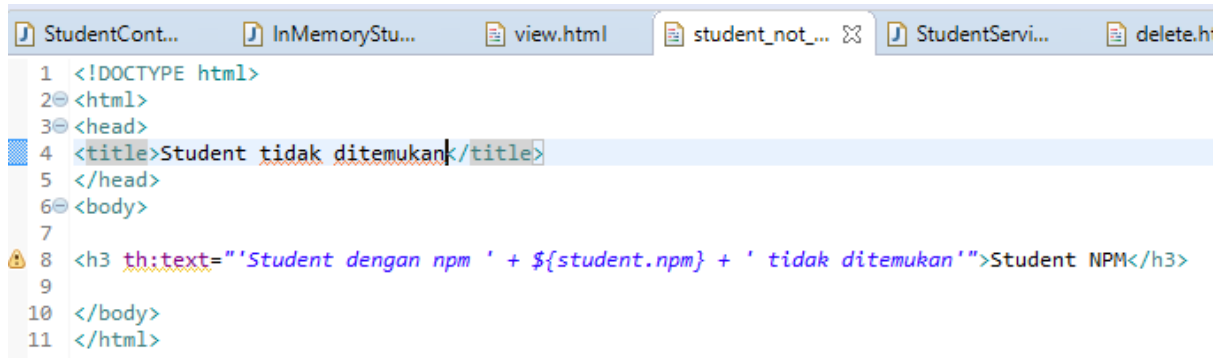
- 1) Buat method **viewStudent** pada kelas **StudentController** seperti *screenshot* berikut:

```

@RequestMapping("/student/view/{npm}")
public String viewStudent(Model model, @PathVariable(value = "npm", required = true) String npm) {
    StudentModel student = studentService.selectStudent(npm);
    if(student != null) {
        model.addAttribute("student", student);
        return "view";
    } else {
        student = new StudentModel();
        student.setNpm(npm);
        model.addAttribute("student", student);
        return "student_not_found";
    }
}

```

- 2) Buat file **student_not_found.html** pada **resources/templates** yang berfungsi untuk menampilkan pemberitahuan bahwa Student yang ingin dicari tidak ada dalam penyimpanan data. Berikut ini isi file **student_not_found.html** :



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Student tidak ditemukan</title>
5 </head>
6 <body>
7
8 <h3 th:text="'Student dengan npm ' + ${student.npm} + ' tidak ditemukan'">Student NPM</h3>
9
10 </body>
11 </html>

```

2. Tahap – tahap pengerjaan latihan nomor 2

- 1) Buat suatu method **deleteStudent** yang menerima String parameter dengan nama npm pada kelas **StudentService** seperti *screenshot* berikut:

```

StudentModel deleteStudent(String npm);

```

- 2) Implementasi method **deleteStudent** , yang berada di kelas **StudentService**, di kelas **InMemoryStudentService** seperti *screenshot* berikut:

```

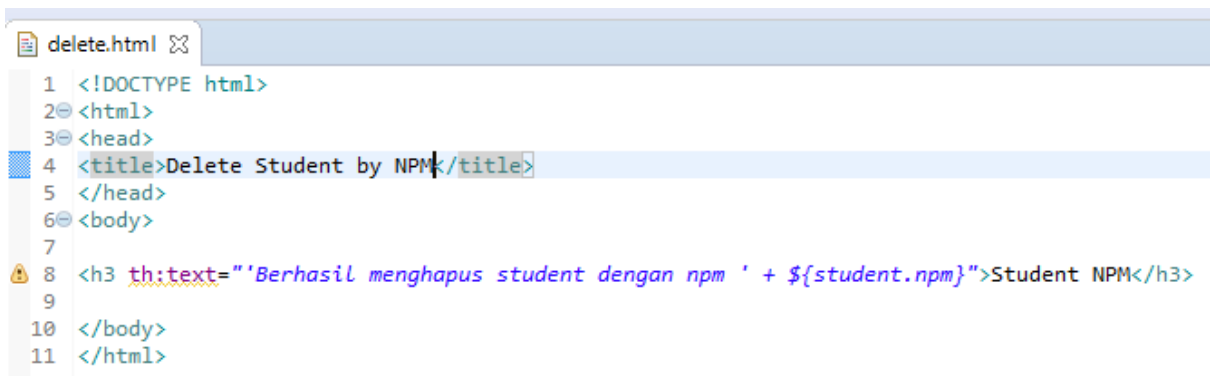
@Override
public StudentModel deleteStudent(String npm) {
    StudentModel student = selectStudent(npm);
    if(student != null && studentList.remove(student)) {
        return student;
    }
    return null;
}

```

- 3) Buat method **deleteStudent** pada kelas **StudentController** seperti *screenshot* berikut:

```
@RequestMapping("/student/delete/{npm}")
public String deleteStudent(Model model, @PathVariable(value = "npm", required = true) String npm) {
    StudentModel student = studentService.deleteStudent(npm);
    if(student != null) {
        model.addAttribute("student", student);
        return "delete";
    } else {
        student = new StudentModel();
        student.setNpm(npm);
        model.addAttribute("student", student);
        return "student_not_found";
    }
}
```

- 4) Buat file **student_not_found.html** seperti di nomor 1 langkah ke-2.
- 5) Buat file **delete.html** pada **resources/templates** yang berfungsi untuk menampilkan pemberitahuan bahwa Student yang ingin dihapus telah berhasil dihapus. Berikut ini isi dari file delete.html:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Delete Student by NPM</title>
5 </head>
6 <body>
7
8 <h3 th:text="'Berhasil menghapus student dengan npm ' + ${student.npm} + 'Student NPM'>
9
10 </body>
11 </html>
```

Penjelasan Method delete pada Latihan

Method delete yang saya implementasikan pada latihan ini (method deleteStudent pada kelas StudentController) memiliki beberapa alur. Pertama, user akan membuka url **http://localhost:8080/student/delete/1235** (1235 merupakan npm dari student yang ingin dihapus) yang akan menyebabkan method delete tersebut akan dieksekusi. Kedua, pada method delete tersebut akan memanggil method deleteStudent, dengan parameter npm student, yang berada di kelas StudentService. Method deleteStudent tersebut akan menghapus data student dari data penyimpanan. Jika method deleteStudent tersebut berhasil menghapus student, method tersebut akan mengembalikan objek student yang dihapus, dan mengembalikan null jika tidak berhasil menghapusnya (dapat disebabkan karena student tersebut tidak ada di data penyimpanan). Ketika objek student yang dikembalikan tidak null, method deleteStudent pada kelas StudentController akan menampilkan delete.html yang berisi pemberitahuan bahwa student yang ingin dihapus telah berhasil dihapus. Sebaliknya, ketika objek student yang dikembalikan null, method deleteStudent pada kelas StudentController akan menampilkan student_not_found.html yang berisi pemberitahuan bahwa student yang ingin dihapus tidak ada dalam data penyimpanan.