

1. Hal yang dipelajari

Pada tutorial kali ini, saya mempelajari cara penggunaan database yaitu dengan menggunakan anotasi dan juga bagaimana melakukan debugging pada Project Spring Boot menggunakan log. Selain itu saya juga mempelajari perbedaan antara Request Method POST dan GET pada form html serta bagaimana penggunaan thymeleaf form bekerja dalam suatu file html.

2. Latihan Menambahkan Delete

a. Method deleteStudent pada class StudentMapper

```
public interface StudentMapper
{
    @Select("select npm, name, gpa from student where npm = #{npm}")
    StudentModel selectStudent (@Param("npm") String npm);

    @Select("select npm, name, gpa from student")
    List<StudentModel> selectAllStudents ();

    @Insert("INSERT INTO student (npm, name, gpa) VALUES (#{npm}, #{name}, #{gpa})")
    void addStudent (StudentModel student);

    @Delete("DELETE FROM student WHERE npm = #{npm}")
    void deleteStudent(String npm);
}
```

Method ini memiliki anotasi delete yang nantinya akan berhubungan dengan penggunaan operasi “DELETE” pada *database*. *Query* yang ada memiliki fungsi menghapus data dari student dengan npm tertentu. Jika dilakukan pemanggilan method deleteStudent dengan parameter String npm, *query* yang ada pada anotasi akan bekerja dan melakukan operasi “DELETE” pada *database* sesuai dengan parameter “npm” yang diberikan

b. Method deleteStudent pada class StudentServiceDatabase

```
StudentCont... StudentModel... StudentServi... view
31 {
32     Log.info ("select all students");
33     return studentMapper.selectAllStudents ();
34 }
35
36
37 @Override
38 public void addStudent (StudentModel student)
39 {
40     studentMapper.addStudent (student);
41 }
42
43
44 @Override
45 public void deleteStudent (String npm)
46 {
47     Log.info("student" + npm + "deleted");
48     studentMapper.deleteStudent(npm);
49 }
50
```

Method ini menerima parameter String npm yang datanya akan digunakan untuk memanggil method deleteStudent yang ada pada class studentMapper. Selain itu terdapat log yang berfungsi melihat apakah parameter yang dimasukkan benar atau tidak, dimana hal ini akan berguna pada proses *debugging*. Log menampilkan npm peserta yang didapat dari parameter method.

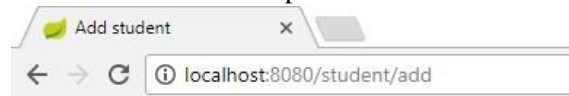
c. Method delete pada class StudentController

```
StudentCont... StudentModel... StudentServi... viewall.html StudentMapp...
81
82
83 @RequestMapping("/student/viewall")
84 public String view (Model model)
85 {
86     List<StudentModel> students = studentDAO.selectAllStudents ();
87     model.addAttribute ("students", students);
88
89     return "viewall";
90 }
91
92
93 @RequestMapping("/student/delete/{npm}")
94 public String delete (Model model, @PathVariable(value = "npm") String npm)
95 {
96     StudentModel student = studentDAO.selectStudent (npm);
97
98     if (student != null) {
99         studentDAO.deleteStudent (npm);
100         return "delete";
101     } else {
102         return "not-found";
103     }
104 }
```

Method ini berfungsi untuk melihat apakah student dengan npm yang akan dihapus ada di dalam database atau tidak. Npm yang akan dicari diambil dari url dengan menggunakan RequestMapping. Jika student ditemukan maka method deleteStudent pada class StudentService dipanggil dengan parameter npm tersebut dan menampilkan delete.html. Jika npm student tidak ditemukan pada *database*, maka akan ditampilkan not-found.html

d. Screenshoot test program

i. Menambahkan student pertama



## Problem Editor

NPM	<input type="text" value="1234567890"/>
Name	<input type="text" value="fulan"/>
GPA	<input type="text" value="3.90"/>
<input type="button" value="Save"/>	



**Data berhasil ditambahkan**

ii. Menambahkan student kedua



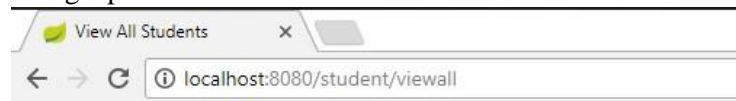
## Problem Editor

NPM   
Name   
GPA



**Data berhasil ditambahkan**

iii. Menghapus student



## All Students

**No. 1**

**NPM = 1234567890**

**Name = fulan**

**GPA = 3.9**

[Delete Data](#)

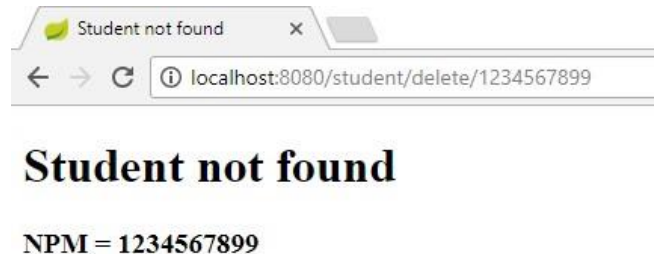
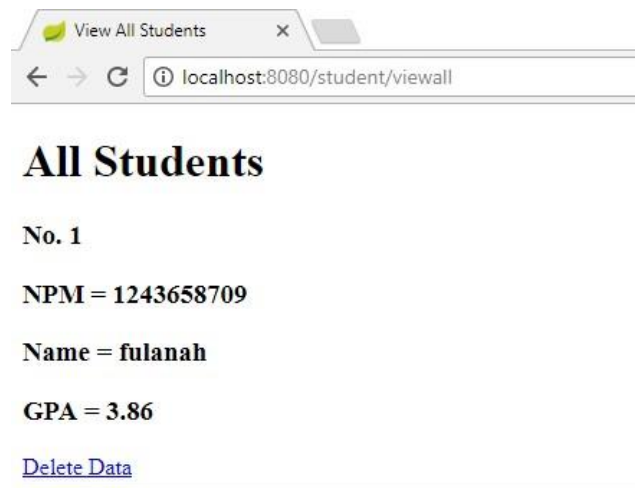
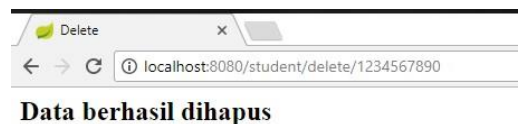
**No. 2**

**NPM = 1243658709**

**Name = fulanah**

**GPA = 3.86**

[Delete Data](#)



3. Latihan Menambahkan Update
- a. Method updateStudent pada class StudentMapper

```
28  
29 @Update("UPDATE student SET name = #{name}, gpa = #{gpa} WHERE npm = #{npm}")  
30 void updateStudent(StudentModel student);  
31 }  
32
```

Method ini memiliki anotasi update yang nantinya akan berhubungan dengan penggunaan operasi "UPDATE" pada *database*. *Query* yang ada memiliki fungsi mengubah data dari student dengan npm tertentu. Jika dilakukan pemanggilan method updateStudent dengan parameter StudentModel student, *query* yang ada pada anotasi akan bekerja dan melakukan operasi "UPDATE" pada *database* sesuai dengan parameter student yang diberikan.

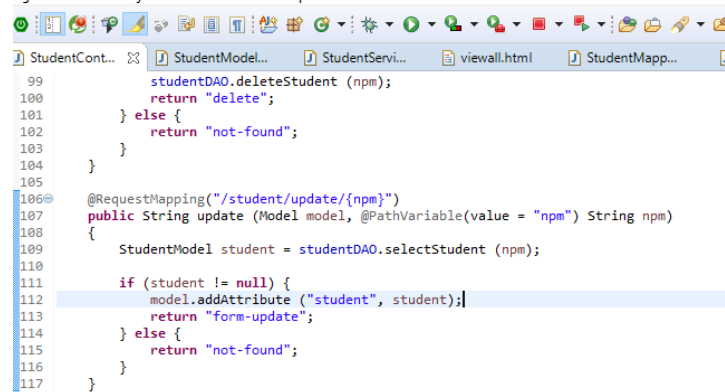
- b. Method updateStudent pada class StudentServiceDatabase

```
,  
  
@Override  
public void updateStudent(StudentModel student)  
{  
    log.info("student" + student.getNpm() + "updated");  
    studentMapper.updateStudent(student);  
}
```

Method ini menerima parameter StudentModel student yang datanya akan digunakan untuk memanggil method updateStudent yang ada pada class studentMapper. Selain itu terdapat log yang berfungsi melihat apakah parameter yang dimasukkan benar atau tidak, dimana hal ini akan berguna pada proses *debugging*. Log menampilkan npm dari student dengan menggunakan method getNpm() yang dimiliki oleh objek StudentModel

- c. Method update pada class StudentController

I04/src/main/java/com/example/controller/StudentController.java - Eclipse  
vigate Search Project Run Window Help



```
99        studentDAO.deleteStudent (npm);  
100        return "delete";  
101    } else {  
102        return "not-found";  
103    }  
104 }  
105  
106 @RequestMapping("/student/update/{npm}")  
107 public String update (Model model, @PathVariable(value = "npm") String npm)  
108 {  
109     StudentModel student = studentDAO.selectStudent (npm);  
110  
111     if (student != null) {  
112         model.addAttribute ("student", student);  
113         return "form-update";  
114     } else {  
115         return "not-found";  
116     }  
117 }  
...
```

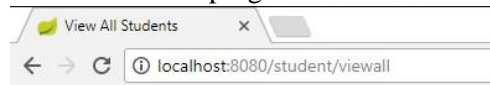
Method ini akan berfungsi ketika url yang ditampilkan adalah “/student/update/{npm}”. Lalu method melihat apakah student dengan npm yang akan diubah ada di dalam database atau tidak. Npm yang akan dicari diambil dari url dengan menggunakan RequestMapping. Jika student ditemukan, maka akan dikirimkan model yang memiliki atribut dengan key “student” dan value student yang datanya ingin diubah, lalu dikirimkan ke form-update.html. Jika student tidak ditemukan, maka akan ditampilkan not-found.html

- d. Method updateSubmit pada class StudentController

```
18  
19 @RequestMapping(value = "/student/update/submit", method = RequestMethod.POST)  
20 public String updateSubmit (  
21     @RequestParam(value = "npm", required = false) String npm,  
22     @RequestParam(value = "name", required = false) String name,  
23     @RequestParam(value = "gpa", required = false) double gpa)  
24 {  
25     StudentModel student = studentDAO.selectStudent (npm);  
26     student.setName(name);  
27     student.setGpa(gpa);  
28     studentDAO.updateStudent(student);  
29     return "success-update";  
30 }  
31 }
```

Method ini akan berfungsi ketika url yang ditampilkan adalah “/student/update/submit” dan method POST. Method akan menerima parameter String npm, String name, dan double gpa yang didapatkan dari method yang ada pada url. Dimana data ini berasal dari form-update.html yang dipanggil oleh method updateStudent pada class StudentControoler. Selanjutnya akan dilakukan pencarian terhadap npm tersebut di database, lalu dilakukan perubahan terhadap atribut name dan gpa student menggunakan method set yang dimiliki oleh objek student. Method ini lalu memanggil method updateStudent pada class StudentService dengan parameter student yang telah diubah, lalu menampilkan success-update.html

e. Screenshoot test program



## All Students

No. 1

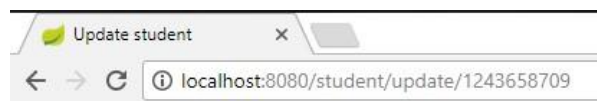
NPM = 1243658709

Name = fulanah

GPA = 3.86

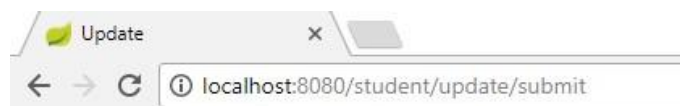
[Delete Data](#)

[Update Data](#)

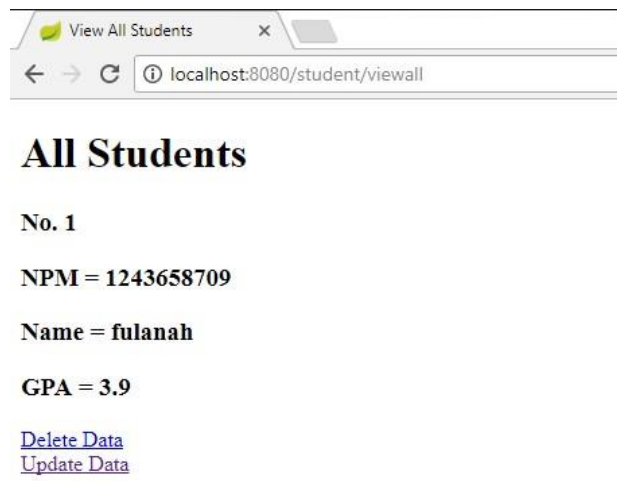


## Edit Student

NPM	<input type="text" value="1243658709"/>
Name	<input type="text" value="fulanah"/>
GPA	<input type="text" value="3.86"/>
<input type="button" value="Update"/>	



**Data berhasil diubah**

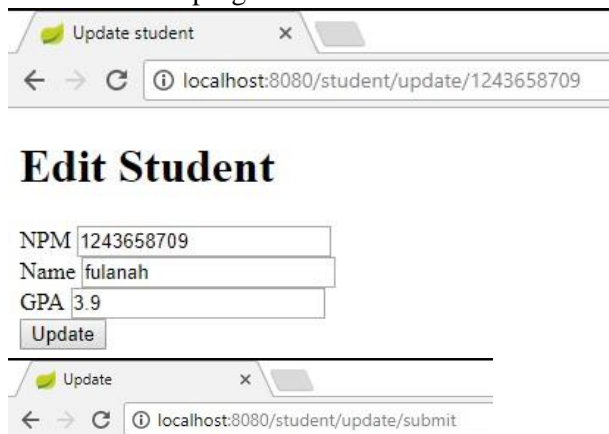


4. Latihan Menggunakan Object Sebagai Parameter  
Mengubah method updateSubmit

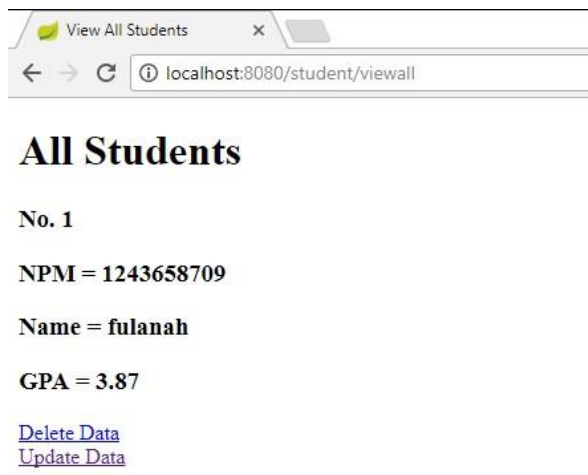
```
@RequestMapping(value = "/student/update/submit", method = RequestMethod.POST)
public String updateSubmit (@ModelAttribute StudentModel student)
{
    studentDAO.updateStudent(student);
    return "success-update";
}
```

Method ini akan dipanggil ketika url yang dibuka adalah “/student/update/submit”, dan mengambil data yang diberikan oleh method = POST. Pada alur program, form-update.html lah yang akan beralih ke url “/student/update/submit” dengan mengirimkan objek dengan tipe data StudentModel. Lalu method ini memanggil method updateStudent pada class StudentService dengan parameter tipe data StudentModel, lalu menampilkan success-update.html .

- a. Screenshot test program



**Data berhasil diubah**



5. Pertanyaan

- a. Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required? Apakah validasi diperlukan?

Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.

Tetap dibutuhkan validasi input dikarenakan objek tetap akan dibuat oleh program walaupun dengan input null. Namun untuk validasi pada tipe data sendiri telah ditangani langsung oleh thymeleaf. Validasi dapat dilakukan pada Controller dengan mengambil atribut dari model objek

- b. Menurut Anda, mengapa form submit biasanya menggunakan POST method dibanding GET method? Apakah perlu penanganan berbeda di header atau body method di controller jika form di post dikirim menggunakan method berbeda?

Dikarenakan GET method meminta representasi dari *resource* yang spesifik. Hal ini mengakibatkan GET tidak tepat digunakan untuk operasi yang menyebabkan efek samping seperti *actions* pada *web applications*. Hal ini dapat digunakan secara sewenang-wenang oleh *robots* atau *crawlers*. (Sumber: <https://stackoverflow.com/questions/3477333/what-is-the-difference-between-post-and-get> )

Dibutuhkan penanganan berbeda karena method memiliki cara-cara yang berbeda dalam mengirimkan data

- c. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?

Tidak, karena satu method hanya bisa menerima satu request method