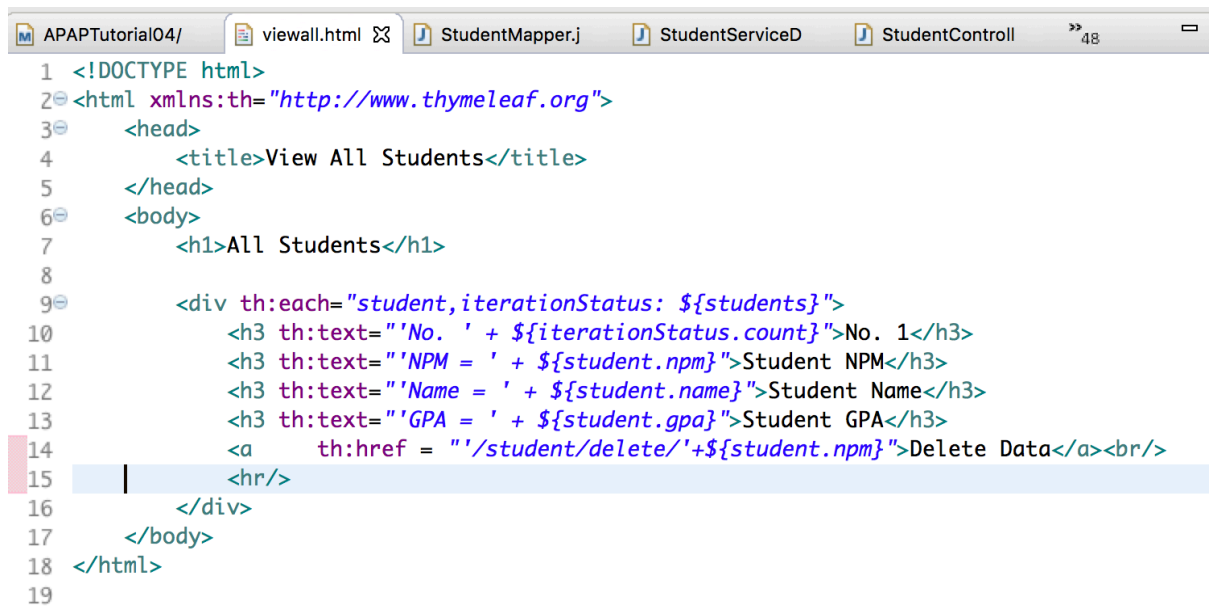


=====  
Emma Sharahwati  
1406557466  
APAP-B  
=====

## Latihan MENAMBAHKAN DELETE

1. Pada tutorial sebelumnya Anda sudah menambahkan method delete. Sekarang implementasikan method tersebut menggunakan database.
2. Pada viewall.html



```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3   <head>
4     <title>View All Students</title>
5   </head>
6   <body>
7     <h1>All Students</h1>
8
9     <div th:each="student, iterationStatus: ${students}">
10      <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
11      <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
12      <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
13      <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
14      <a th:href = "/student/delete/' + ${student.npm}">Delete Data</a><br/>
15      <hr/>
16    </div>
17  </body>
18 </html>
19
```

3. Tambahkan method **deleteStudent** yang ada di class **StudentMapper**
  - a. Tambahkan method delete student yang menerima parameter NPM.
  - b. Tambahkan annotation delete di atas dan SQL untuk menghapus

```
APAPTutorial04/ viewall.html StudentMapper.j StudentServiceD StudentControll
3 import java.util.List;
4
5 import org.apache.ibatis.annotations.Delete;
6 import org.apache.ibatis.annotations.Insert;
7 import org.apache.ibatis.annotations.Mapper;
8 import org.apache.ibatis.annotations.Param;
9 import org.apache.ibatis.annotations.Select;
10
11 import com.example.model.StudentModel;
12
13 @Mapper
14 public interface StudentMapper
15 {
16     @Select("select npm, name, gpa from student where npm = #{npm}")
17     StudentModel selectStudent (@Param("npm") String npm);
18
19     @Select("select npm, name, gpa from student")
20     List<StudentModel> selectAllStudents ();
21
22     @Insert("INSERT INTO student (npm, name, gpa) VALUES (#{npm}, #{name}, #{gpa})")
23     void addStudent (StudentModel student);
24
25     @Delete("DELETE FROM student where npm = #{npm}")
26     void deleteStudent(StudentModel student);
27 }
--
```

4. Lengkapi method **deleteStudent** yang ada di class **StudentServiceDatabase**
  1. Tambahkan log untuk method tersebut dengan cara menambahkan
  2. Panggil method delete student yang ada di Student Mapper

```
@Override
public void deleteStudent (String npm)
{
    log.info("student "+npm+" deleted");
    StudentModel student = studentMapper.selectStudent(npm);
    studentMapper.deleteStudent(student);
}
}
```

5. Lengkapi method **delete** pada class **StudentController**
  1. Tambahkan validasi agar jika mahasiswa tidak ditemukan tampilkan view not-found
  2. Jika berhasil delete student dan tampilkan view delete

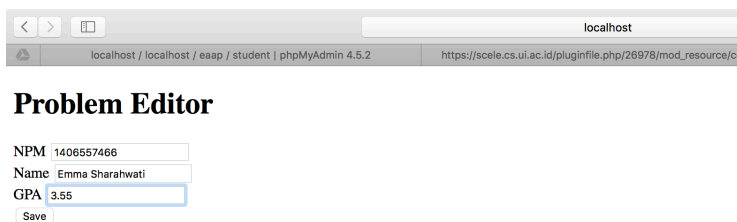
```

74 @RequestMapping("/student/delete/{npm}")
75 public String delete(Model model, @PathVariable(value = "npm") String npm) {
76     StudentModel student = studentDAO.selectStudent(npm);
77     if (student != null) {
78         studentDAO.deleteStudent(npm);
79         return "delete";
80     } else {
81         return "not-found";
82     }
83 }
84

```

6. Jalankan Spring Boot app dan lakukan beberapa insert

Menambahkan data student



localhost

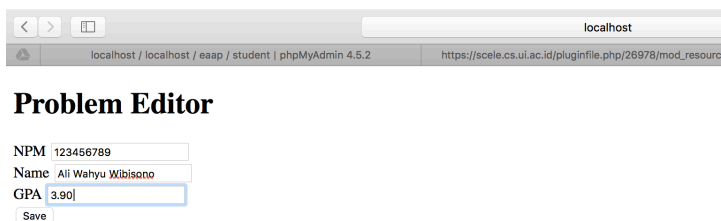
localhost / localhost / eaap / student | phpMyAdmin 4.5.2

### Problem Editor

NPM

Name

GPA



localhost

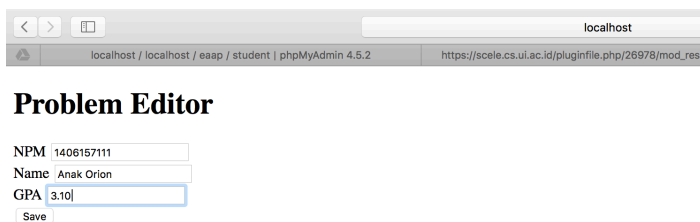
localhost / localhost / eaap / student | phpMyAdmin 4.5.2

### Problem Editor

NPM

Name

GPA



localhost

localhost / localhost / eaap / student | phpMyAdmin 4.5.2

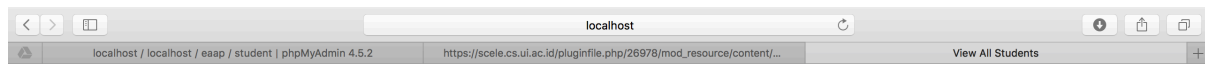
### Problem Editor

NPM

Name

GPA

## Menampilkan semua data student



### All Students

No. 1

NPM = 123456789

Name = Ali Wahyu Wibisono

GPA = 3.9

[Delete Data](#)

No. 2

NPM = 1406157111

Name = Anak Orion

GPA = 3.1

[Delete Data](#)

No. 3

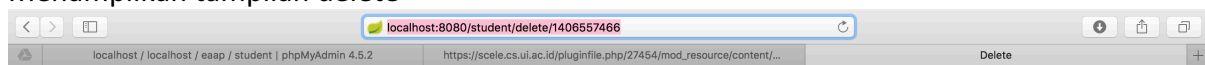
NPM = 1406557466

Name = Emma Sharahwati

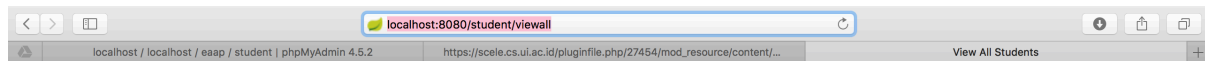
GPA = 3.55

[Delete Data](#)

## Menampilkan tampilan delete



Data berhasil dihapus



### All Students

No. 1

NPM = 123456789

Name = Ali Wahyu Wibisono

GPA = 3.9

[Delete Data](#)

No. 2

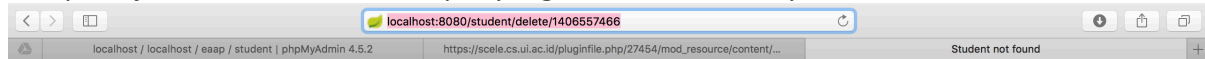
NPM = 1406157111

Name = Anak Orion

GPA = 3.1

[Delete Data](#)

Tampilan jika melakukan delete npm yang sama kedua kalinya



**Student not found**

NPM = 1406557466

## Latihan Menambahkan Update

1. Tambahkan method **updateStudent** pada class **StudentMapper**
  1. Parameternya adalah StudentModel student
  2. Annotationnya adalah @Update

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

2. Tambahkan method **updateStudent** pada interface **StudentService**

```
void updateStudent(StudentModel student);
```

3. Tambahkan implementasi method **updateStudent** pada class **StudentServiceDatabase**.

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

4. Tambahkan link Update Data pada **viewall.html**

```
viewall.html StudentMapper.j StudentServiceD StudentControll StudentService. 53
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3   <head>
4     <title>View All Students</title>
5   </head>
6   <body>
7     <h1>All Students</h1>
8
9     <div th:each="student, iterationStatus: ${students}">
10       <h3 th:text="No. ' + ${iterationStatus.count}">No. 1</h3>
11       <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
12       <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
13       <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
14       <a th:href = "/student/delete/' + ${student.npm}">Delete Data</a><br/>
15       <a th:href = "/student/update/' + ${student.npm}">Update Data</a><br/>
16     </div>
17   </body>
18 </html>
19
20
```

5. Copy view form-add.html menjadi **form-update.html** .

```
viewall.html StudentMapper.j StudentServiceD StudentService. form-update.htm 53
1 <!DOCTYPE HTML>
2 <html xmlns="http://www.w3.org/1999/xhtml"
3   xmlns:th="http://www.thymeleaf.org">
4   <head>
5
6     <title>Update student</title>
7     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
8   </head>
9
10  <body>
11
12    <h1 class="page-header">Update Student</h1>
13
14    <form action="/student/update/submit" method="post">
15      <div>
16        <label for="npm">NPM</label> <input type="text" name="npm" readonly="true"
17          th:value="${student.npm}" />
18      </div>
19      <div>
20        <label for="name">Name</label> <input type="text" name="name"
21          th:value="${student.name}" />
22      </div>
23      <div>
24        <label for="gpa">GPA</label> <input type="text" name="gpa"
25          th:value="${student.gpa}" />
26      </div>
27
28      <div>
29        <button type="submit" name="action" value="save">Update</button>
30      </div>
31    </form>

```

6. Copy view success-add.html menjadi **success-update.html**

```
viewall.html StudentServiceD StudentService. form-update.htm success-update. » 53
1 <html>
2   <head>
3     <title>Update</title>
4   </head>
5   <body>
6     <h2>Data berhasil diubah</h2>
7   </body>
8 </html>
```

7. Tambahkan method **update** pada class **StudentController**
8. Tambahkan method **updateSubmit** pada class **StudentController**

```
viewall.html StudentServiceD StudentController » form-update.htm success-update. » 53
84   }
85
86   @RequestMapping("/student/update/{npm}")
87   public String update(Model model, @PathVariable(value = "npm") String npm) {
88       StudentModel student = studentDAO.selectStudent(npm);
89       if (student != null) {
90           model.addAttribute("student", student);
91           return "form-update";
92       } else {
93           return "not-found";
94       }
95   }
96
97   @RequestMapping(value = "/student/update/submit", method = RequestMethod.POST)
98   public String updateSubmit(
99       @RequestParam(value = "npm", required = false) String npm,
100      @RequestParam(value = "name", required = false) String name,
101      @RequestParam(value = "gpa", required = false) double gpa)
102   {
103       StudentModel student = studentDAO.selectStudent(npm);
104       student.setName(name);
105       student.setGpa(gpa);
106       studentDAO.updateStudent(student);
107       return "success-update";
108   }
109 }
```

9. Jalankan Spring Boot dan coba test program Anda.

## Latihan MENGGUNAKAN OBJECT SEBAGAI PARAMETER

1. Pada tutorial di atas Anda masih menggunakan RequestParam untuk menghandle form submit. Sehingga ada banyak parameter pada method Anda. Bayangkan jika Anda memiliki form yang memiliki banyak field, maka parameternya akan sangat banyak dan tidak rapih.
2. SpringBoot dan Thymeleaf memungkinkan agar method **updateSubmit** menerima parameter berupa model **StudentModel** . Metode ini lebih disarankan dibandingkan menggunakan **RequestParam**.
3. Cara lengkapnya silakan ikuti pada link Handling Form berikut: (<https://spring.io/guides/gs/handling-form-submission/> )

4. Tahapannya kurang lebih sebagai berikut:
  1. Menambahkan `th:object='${student}'` pada tag **<form>** di view
  2. Menambahkan `th:field='*{[nama_field]}'` pada setiap input
  3. Ubah method `updateSubmit` pada `StudentController` yang hanya menerima parameter berupa `StudentModel`
4. Tes lagi aplikasi Anda.

## Latihan

1. Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required seperti jika menggunakan `RequestParam`? Apakah validasi diperlukan?

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

Jawab : cara melakukan validasi yaitu dengan mengecek pada objek pada parameternya. Validasi nya yaitu dengan menambahkan anotasi. Validasi diperlukan jika sebuah data harus selalu ada.

2. 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?

Jawab : Dalam melakukan pengiriman data, method POST lebih baik dibandingkan dengan method GET khususnya yang rahasia dan menggunakan password. Sedangkan method GET dapat digunakan pada pengiriman data yang tidak rahasia. Ya, butuh penanganan berbeda.

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

Jawab : Ya, mungkin. Satu method dapat menerima lebih dari satu jenis request method.