

Menambahkan Delete

1. Menambahkan link pada viewall.html dimana mendirect ke

<http://localhost:8080/student/viewall>

```
<a th:href="'/student/delete/' + ${student.npm}"> Delete Data</a><br />
```

2. Menambahkan method delete di studentmapper dimana fungsi method ini adalah memanggil query untuk menghapus data berdasarkan parameter string npm.

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

3. Menambahkan method deleteStudent yang ada di class StudentServiceDatabase yang di dalamnya juga ada log.info yang berfungsi mempermudah mendibug fitur delete.

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

4. Menambahkan method delete pada class StudentController, method ini sebagai control dari fungsi delete query dan kemudian menampilkan hasilnya di view

```
@RequestMapping("/student/delete/{npm}")  
public String delete(Model model, @PathVariable(value = "npm") String npm) {  
    StudentModel student = studentDAO.selectStudent(npm);  
  
    if (student != null) {  
        studentDAO.deleteStudent(npm);  
        return "delete";  
    } else {  
        return "not-found";  
    }  
}
```

5. Contoh program:

Data-data:

← → ↻ localhost:8080/student/viewall

All Students

No. 1

NPM = 1234567890

Name = No Name

GPA = 1.0

[Delete Data](#)

[Update Data](#)

No. 2

NPM = 1506724096

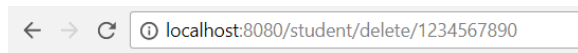
Name = Fajar Marseto

GPA = 4.0

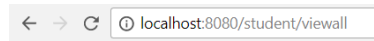
[Delete Data](#)

[Update Data](#)

Delete No Name:



Data berhasil dihapus



All Students

No. 1

NPM = 1506724096

Name = Fajar Marseto

GPA = 4.0

[Delete Data](#)
[Update Data](#)

Menghapus data untuk kedua kalinya npm 1234567890:



Student not found

NPM = 1234567890

Menambahkan Update

1. Menambahkan method updateStudent pada class StudentMapper, method ini menggunakan query @update untuk mengupdate data di database dengan paramaternya model StudentModel

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

2. Menambahkan method updateStudent pada interface StudentService yang berfungsi untuk memanggil method update student yang diimplement di studentcontroller

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

3. Menambahkan implementasi method updateStudent pada class StudentServiceDatabase yang di dalamnya juga ada log.info yang berfungsi mempermudah mendibug fitur update

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

4. Menambahkan link Update Data pada viewall.html dimana mendirect ke

[http://localhost:8080/student/form-update/\(npm\)](http://localhost:8080/student/form-update/(npm))

```
<a th:href="'/student/form-update/' + ${student.npm}"> Update Data</a><br />
```

5. Membuat view yang isinya form untuk mengupdate data di form-update.html.

```
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</h1>
13
14 <form action="/student/update/submit" method="post">
15 <div>
16 <label for="npm">NPM</label> <input type="text" name="npm"
17     readonly="true" 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>
32
33 </body>
34
35 </html>
```

6. Membuat view yang isinya jika kita berhasil mengupdate data maka akan menampilkan view di success-update.html.

```
1 <html>
2 <head>
3 <title>Update</title>
4 </head>
5 <body>
6 <h2>Data berhasil diupdate</h2>
7 </body>
8 </html>
```

7. Membuat method update pada class StudentController sebagai control yang melanjutkan perintah tombol update viewall kemudian akan menselect objek untuk diambil datanya, kemudian akan dimasukan ke dalam form view di form-update. Jika gagal akan dialihkan ke view not-found.

```
@RequestMapping("/student/form-update/{npm}")
public String update(Model model, @PathVariable(value = "npm") String npm) {
    StudentModel student = studentDAO.selectStudent(npm);



    if (student != null) {
        model.addAttribute("student", student);
        return "form-update";
    } else {
        return "not-found";
    }
}
```

8. Membuat method updateSubmit pada class StudentController sebagai control setelah form update di submit maka di diambil data-data kemudian dimasukan ke dalam objek student yang kemudian di alihkan ke query update

```
@RequestMapping(value = "/student/update/submit", method = RequestMethod.POST)
public String updateSubmit(@RequestParam(value = "npm", required = false) String npm,
    @RequestParam(value = "name", required = false) String name,
    @RequestParam(value = "gpa", required = false) double gpa) {
    StudentModel student = new StudentModel(npm, name, gpa);
    studentDAO.updateStudent(student);
    return "success-update";
}
```

9. Contoh program:

Data-data:

  localhost:8080/student/viewall	
<h2>All Students</h2>	
No. 1	
NPM = 1506724096	
Name = Fajar Marseto	
GPA = 4.0	
Delete Data	
Update Data	
<hr/>	
No. 2	
NPM = 88888	
Name = Angka Delapan	
GPA = 3.0	
Delete Data	
Update Data	
<hr/>	

Update data npm 88888 dengan name=Bukan Tujuh dan gpa=2.9:

localhost:8080/student/form-update/88888

Update

NPM
Name
GPA

localhost:8080/student/form-update/88888

Update

NPM
Name
GPA

localhost:8080/student/update/submit

Data berhasil diupdate

localhost:8080/student/viewall

All Students

No. 1

NPM = 1506724096

Name = Fajar Marseto

GPA = 4.0

[Delete Data](#)

[Update Data](#)

No. 2

NPM = 88888

Name = Bukan Tujuh

GPA = 2.9

[Delete Data](#)

[Update Data](#)

Menggunakan Object Sebagai Parameter

1. Mengubah tag form menjadi object dengan cara, menambahkan th:object="\${student}" dan mengganti inputan dengan th:field="*{npm}"

```
<form action="/student/update/submit" th:object="${student}" method="post">
  <div>
    <label for="npm">NPM</label> <input type="text" name="npm"
      readonly="true" th:field="*{npm}" />
    </div>
    <div>
    <label for="name">Name</label> <input type="text" name="name"
      th:field="*{name}" />
    </div>
    <div>
    <label for="gpa">GPA</label> <input type="text" name="gpa"
      th:field="*{gpa}" />
    </div>

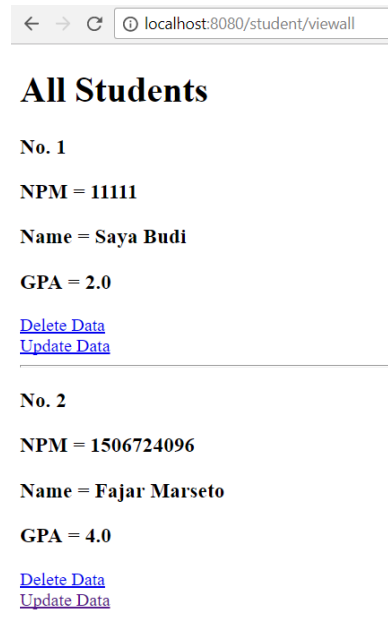
    <div>
    <button type="submit" name="action" value="save">Update</button>
    </div>
</form>
```

2. Mengubah method updateSubmit pada StudentController yang hanya menerima parameter berupa StudentModel dan tanpa memasukan data-data ke dalam object

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

3. Contoh program:

Data-data:



All Students

No. 1

NPM = 11111

Name = Saya Budi

GPA = 2.0

[Delete Data](#)
[Update Data](#)

No. 2

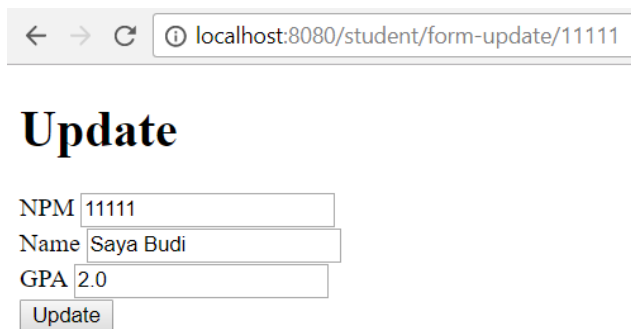
NPM = 1506724096

Name = Fajar Marseto

GPA = 4.0

[Delete Data](#)
[Update Data](#)

Mengubah data npm=11111 dengan name=Saya bukan Budi dan gpa=3.9

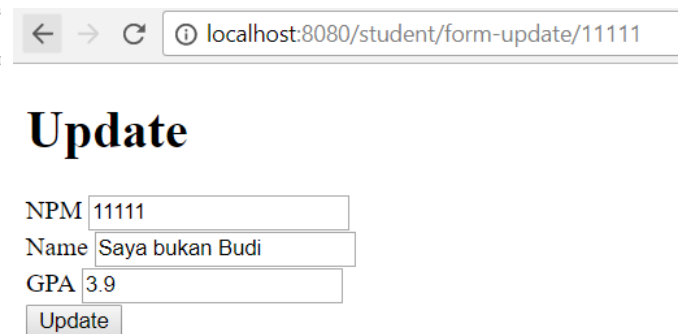


Update

NPM

Name

GPA

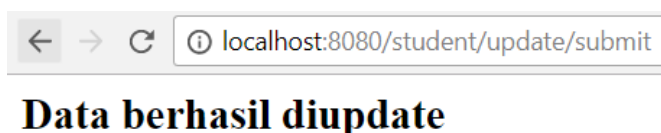


Update

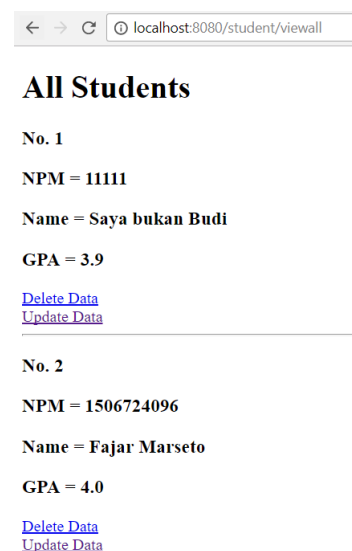
NPM

Name

GPA



Data berhasil diupdate



All Students

No. 1

NPM = 11111

Name = Saya bukan Budi

GPA = 3.9

[Delete Data](#)
[Update Data](#)

No. 2

NPM = 1506724096

Name = Fajar Marseto

GPA = 4.0

[Delete Data](#)
[Update Data](#)

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.

Validasi diperlukan karena dalam parameter tidak bisa menggunakan required seperti biasanya, hal yang perlu dilakukan adalah menambahkan @NotEmpty variable String npm di StudentModel dan @NotNull variable String nama dan Double gpa di StudentModel, fungsi dari @NotEmpty adalah valuenya harus diisi tidak boleh kosong dan valuenya tidak boleh null, fungsi dari @NotNull adalah valuenya tidak boleh null dan boleh kosong tetap disimpan.

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?

Karena method GET parameternya akan tersimpan di dalam browser sehingga bisa saja diakses oleh orang lain sedangkan method POST parameternya tidak akan disimpan di browser dan langsung terkirim melalui url

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

Bisa saja seperti “method = {RequestMethod.POST, RequestMethod.GET}”