Tengku Chavia Zagita
1506757642
Tutorial 4
APAP – A

# Menggunakan Database dan Melakukan Debugging dalam Project Spring Boot

## Latihan Menambahkan Delete

1. Pada viewall.html tambahkan

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

2. 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

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

3. Lengkapi method deleteStudent yang ada di class StudentServiceDatabase
   a. Tambahkan log untuk method tersebut dengan cara menambahkan
   b. Panggil method delete student yang ada di Student Mapper

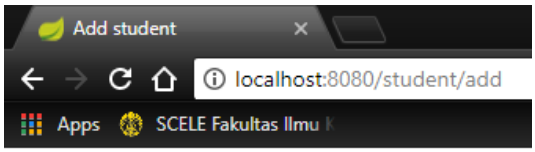
```
@Override
 public void deleteStudent (String npm)
 {
    log.info("student " + npm + " deleted");
    studentMapper.deleteStudent(npm);
 }
```

4. Lengkapi method delete pada class StudentController
   a. Tambahkan validasi agar jika mahasiswa tidak ditemukan tampilkan view not-found
   b. Jika berhasil delete student dan tampilkan view delete

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

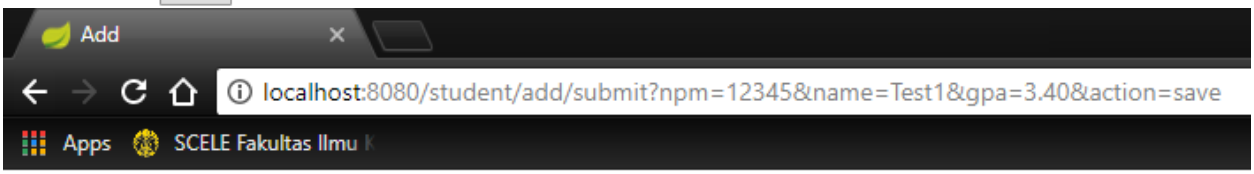
Tengku Chavia Zagita
1506757642
Tutorial 4
APAP – A

5. Jalankan Spring Boot app dan lakukan beberapa insert
    a. Insert 1



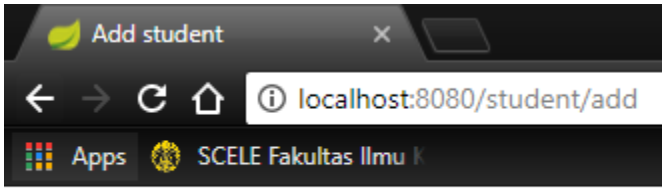

# Problem Editor

NPM 12345
Name Test1
GPA 3.40
Save



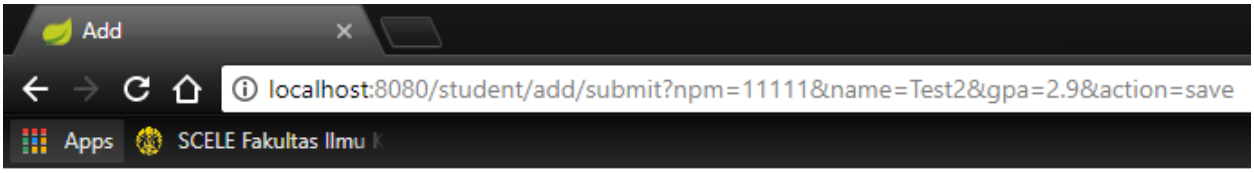

**Data berhasil ditambahkan**

    b. Insert 2




# Problem Editor

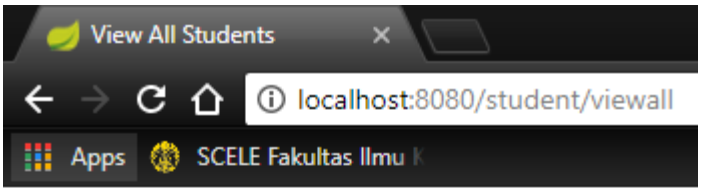
NPM 11111
Name Test2
GPA 2.9
Save




**Data berhasil ditambahkan**

Tengku Chavia Zagita
1506757642
Tutorial 4
APAP – A

6. View All




# All Students

**No. 1**

**NPM = 11111**

**Name = Test2**

**GPA = 2.9**

Delete data

---

**No. 2**

**NPM = 12345**

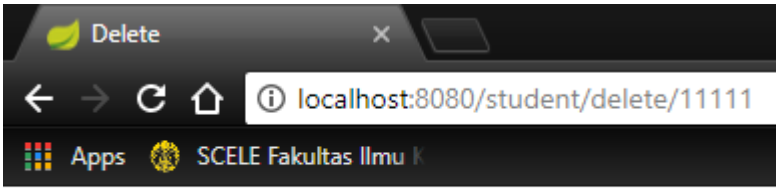
**Name = Test1**

**GPA = 3.4**

Delete data

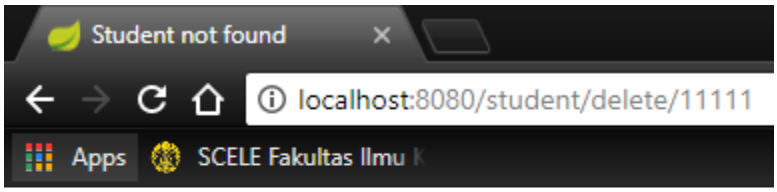
---

7. Delete student with npm = 11111




**Data berhasil dihapus**

8. Delete student with npm = 11111 again




# Student not found

**NPM = 11111**

## Latihan Menambahkan Update

1. Tambahkan method updateStudent pada class StudentMapper
    a. Parameternya adalah StudentModel student
    b. Annotationnya adalah @Update
    c. Lengkapi SQL update –nya

```
@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. Jangan lupa tambahkan logging pada method ini.

```
    @Override
    public void updateStudent (StudentModel student)
    {
        log.info("student " + student.getNpm() + "'s information has been updated");
        studentMapper.updateStudent(student);
    }
```

4. Tambahkan link Update Data pada viewall.html

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

5. Copy view form-add.html menjadi form-update.html.
    a. Ubah info-info yang diperlukan seperti title, page-header, tombol menjadi update dll.
    b. Ubah action form menjadi /student/update/submit
    c. Ubah method menjadi post

d. Untuk input npm dan gpa

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

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

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

```
<html>
	<head>
		<title>Update</title>
	</head>
	<body>
		<h2>Data berhasil diubah</h2>
	</body>
</html>
```

7. Tambahkan method update pada class StudentController
   a. Request mapping ke /student/update/{npm}
   b. Sama halnya seperti delete, lakukan validasi.
   c. Jika student dengan npm tidak ada tampilkan view not-found, jika ada tampilkan view form-update

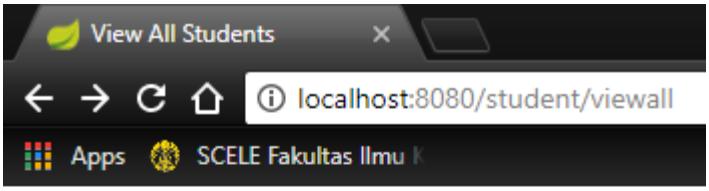
```
@RequestMapping("/student/update/{npm}")
 public String update(Model model, @PathVariable(value = "npm") String npm)
 {
	StudentModel student = studentDAO.selectStudent(npm);
	if (student == null) {
		return "not-found";
	} else {
		model.addAttribute("student", student);
		return "form-update";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

8. Tambahkan method updateSubmit pada class StudentController
   a. Karena menggunakan post method maka request mappingnya adalah sebagai berikut:
   b. Ganti header methodnya
   c. Lengkapi method dengan memanggil method update Student dan kembalikan view success-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 = studentDAO.selectStudent(npm);
student.setGpa(gpa);
student.setName(name);
studentDAO.updateStudent(student);

return "success-update";
}
```

9. Jalankan Spring Boot dan coba test program Anda.
    a. Pilih Update Data




# All Students

**No. 1**

**NPM = 12345**

**Name = Test1**

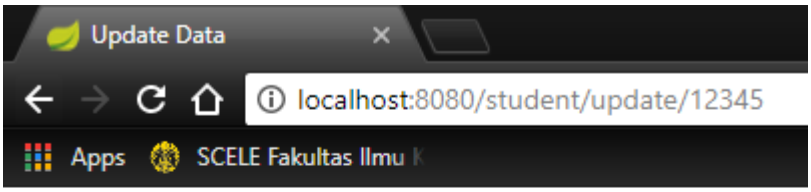
**GPA = 3.4**

Delete data Update data

---

Tengku Chavia Zagita
1506757642
Tutorial 4
APAP – A

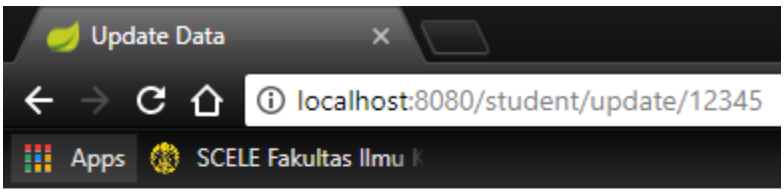
b. Sebelum perubahan




# Update Data

NPM 12345
Name Test1
GPA 3.4
Update

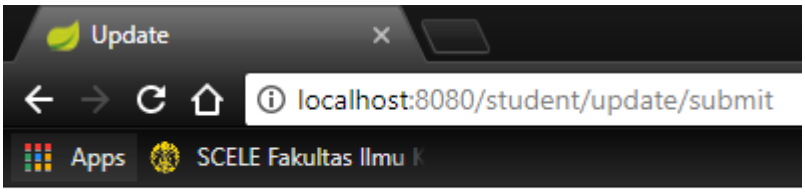
c. Sesudah perubahan pada nama




# Update Data

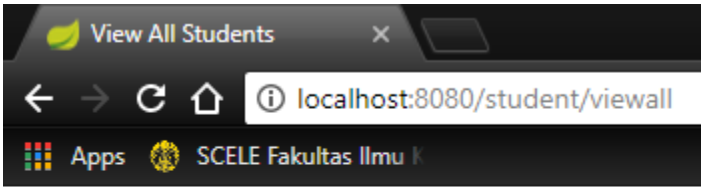
NPM 12345
Name Test update
GPA 3.4
Update

d. Data berahasil diubah




# Data berhasil diubah

Tengku Chavia Zagita
1506757642
Tutorial 4
APAP – A




# All Students

**No. 1**

**NPM = 12345**

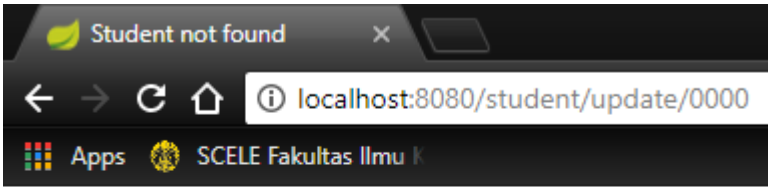
**Name = Test update**

**GPA = 3.4**

Delete data Update data

---

e. Coba update dengan npm yang tidak ada




# Student not found

**NPM = 0000**

## Latihan Menggunakan Object Sebagai Parameter

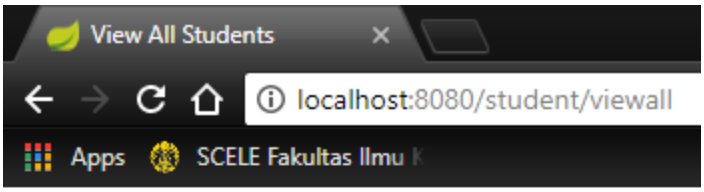
1. Menambahkan th:object=”${student}” pada tag <form> di view
2. Menambahkan th:field=”*{[nama_field]}” pada setiap input
3. Ubah method updateSubmit pada StudentController yang hanya menerima parameter berupa StudentModel

Tengku Chavia Zagita
1506757642
Tutorial 4
APAP – A

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

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

4. Tes lagi aplikasi Anda.
    a. Data awal




# All Students

**No. 1**

**NPM = 12345**

**Name = Test update**

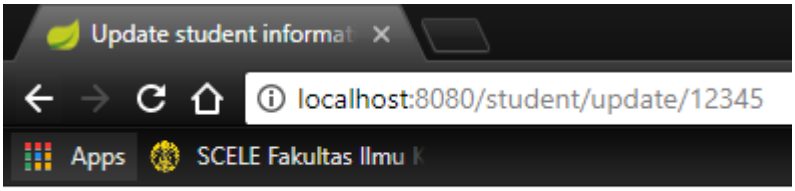
**GPA = 3.4**

Delete data Update data

---

    b. Update

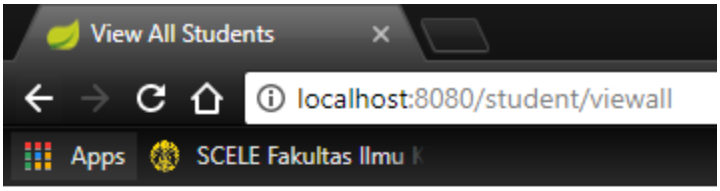
Tengku Chavia Zagita
1506757642
Tutorial 4
APAP – A




# Problem Editor

NPM 12345
Name Test update
GPA 3.99
Save

c. Data akhir




# All Students

**No. 1**

**NPM = 12345**

**Name = Test update**

**GPA = 3.99**

Delete data Update data

---

Tengku Chavia Zagita
1506757642
Tutorial 4
APAP – A

## Pertanyaan

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 input dapat ditambahkan anotasi @Not Null untuk *handle* input yang kosong

2. Menurut Anda, mengapa form submit biasanya menggunakan POST method disbanding GET method? Apakah perlu penanganan berbeda di header atau body method di controller jika form di post dikirim menggunakan method berbeda?
   →Karena method POST dapat memudahkan jumlah input yang dibutuhkan sangat banyak, serta keamanan informasi lebih terjamin karena data dimasukkan pada *field* tidak pada *address bar*
   →Ya, jika menggunakan method POST tambahkan request method POST pada header controller

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   → Ya, bisa. Controller bisa membedakan input sesuai jenis methodnya.