Latihan

A. Delete

Pada viewall.html > di bawah th : each tambahkan

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

Tag diatas berfungsi untuk memanggil url student/delete/[student_npm]

studentMapper berikut berfungsi melakukan delete dengan menambah query:

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

Pada method **deleteStudent** di class StudentController tambahkan method di bawah ini :

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

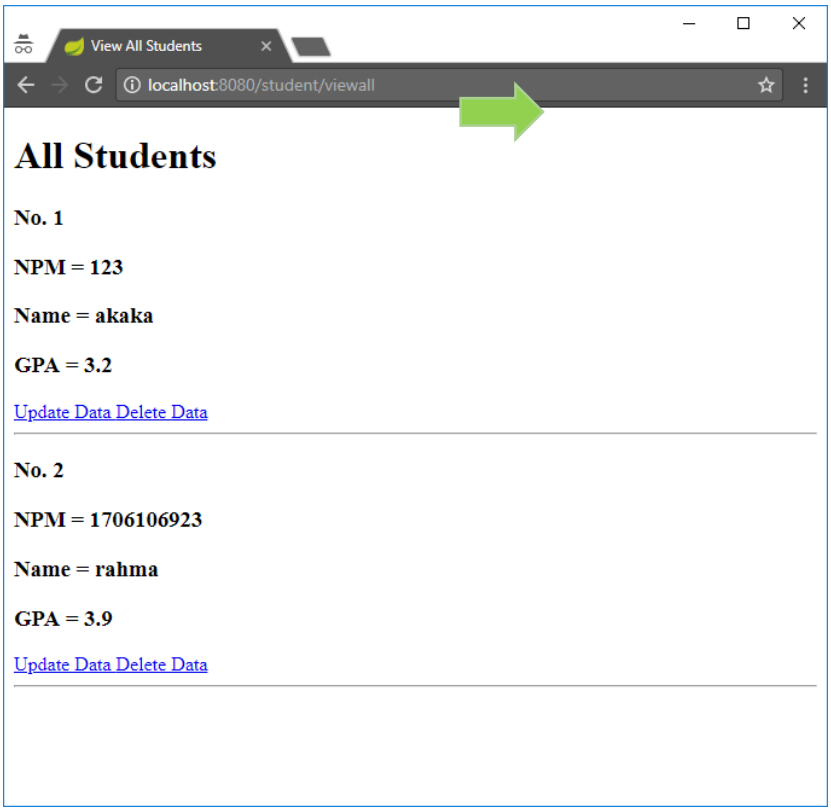
Code diatas adalah code yang berfungsi untuk melakukan debug yang mencetak student [nomor_NPM] jika proses penghapusan berhasil dilakukan.

Sementara untuk class **StudentController** tambahkan method **delete** seperti berikut :

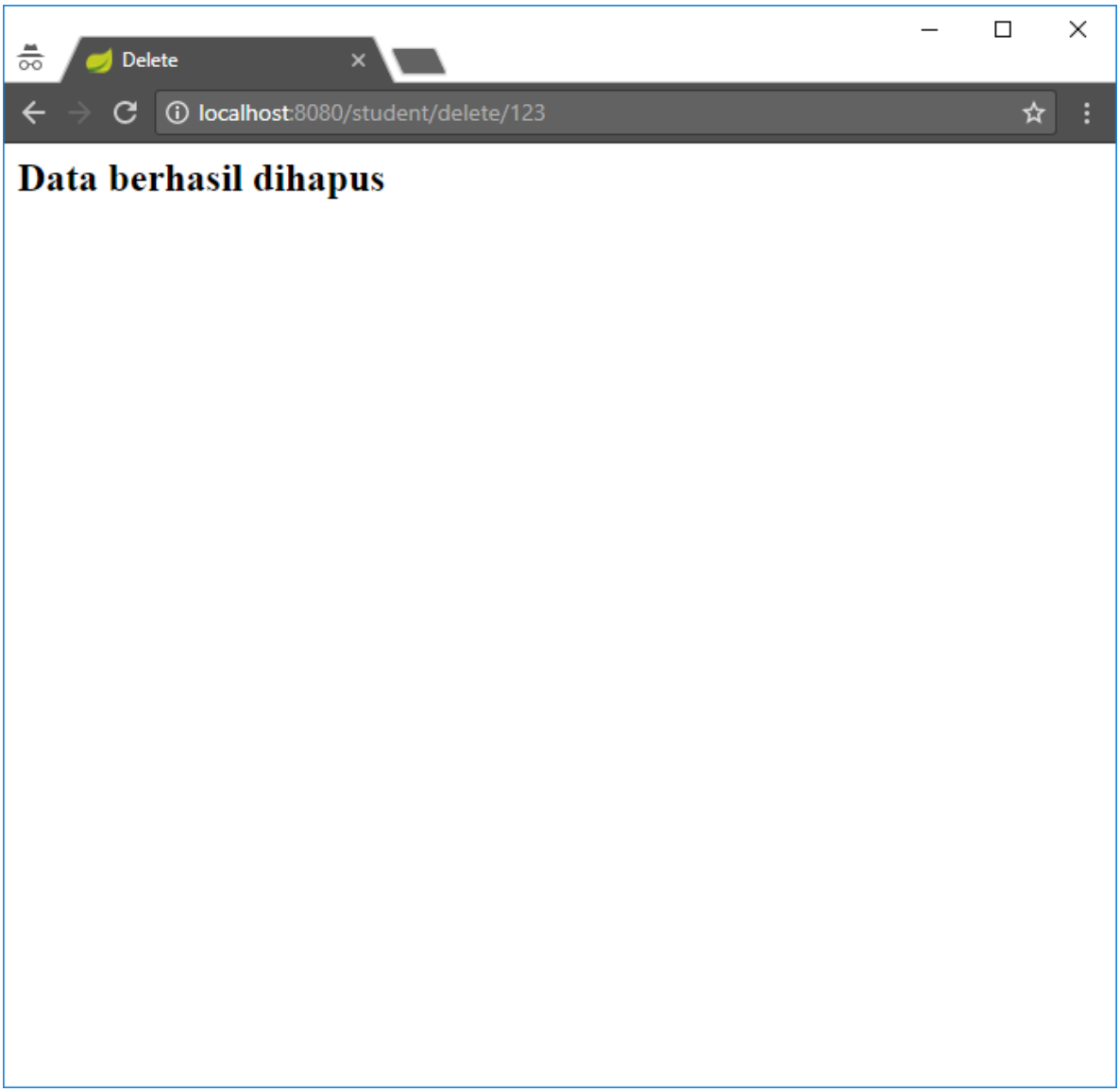
```
@RequestMapping("/student/delete/{npm}")
public String delete (Model model,
          @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent (npm);

    if (student != null){
         studentDAO.deleteStudent(npm);
         return "delete";
    }else {
         model.addAttribute ("npm", npm);
         return "not-found";
    }
}
```

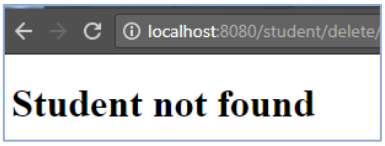
Dengan menambahkan method ini, apabila data berhasil ditemukan, method deleteStudent akan menampilkan view delete. Tapi jika gagal, akan muncul page “not-found. Berikut ini adalah page proses delete :



Data npm 123 dihapus


Delete
localhost:8080/student/delete/123
Data berhasil dihapus

Apabila tidak ada data:



B. Update

Pada form viewall.html tambahkan tag berikut :

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

Tag ini berguna untuk memanggil url update sesuai dengan npmStudent

class **StudentMapper** > tambahakan tag berikut yang berfungsi untuk query update dan memanggil fungsi update

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

class StudentService interface tambahkan berikut:

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

Pada **StudentServiceDatabase** terdapat method **updateStudent**

```
@Override
public void updateStudent (StudentModel student)
{
   studentMapper.updateStudent(student);
}
```

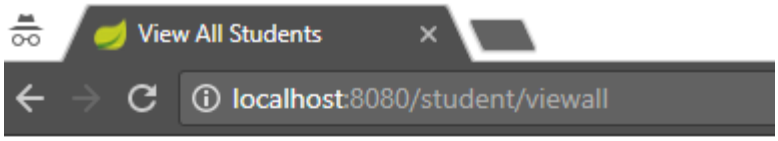
class **StudentController** ditambahkan method update apabila berhasul maka akan mampilkan form-update apabila tidak, maka akan menampilkan view not found

```
@RequestMapping ("/student/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 {
          model.addAttribute ("npm", npm);
          return "not-found";
    }
}

    @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";
    }
```



All Students

No. 1

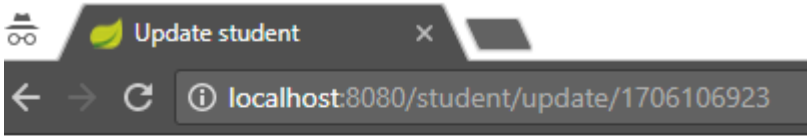
NPM = 1706106923

Name = rahma

GPA = 3.9

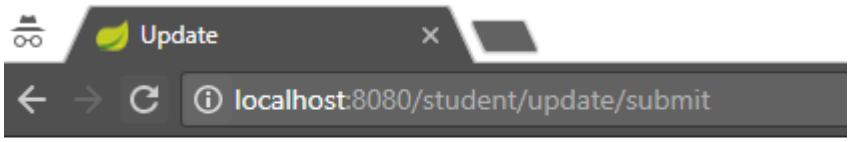
Update Data Delete Data

Apabila berhasil:



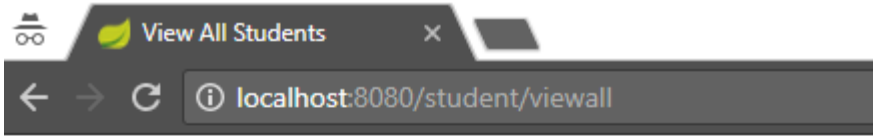
Problem Editor

NPM 1706106923
Name rahmaaaaaaa
GPA 3.9
Save



Data berhasil diperbarui

Hasil dari update:



All Students

No. 1

NPM = 1706106923

Name = rahmaaaaa

GPA = 3.9

Update Data Delete Data

Hasil tidak adata yang bisa di update

Student not found

C. Object Sebagai Parameter

form-update.html diubah:

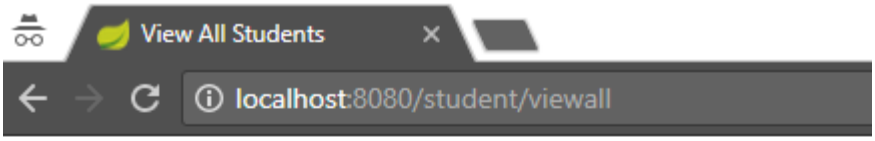
```
	<form action="/student/update/submit" th:object=${student}
method="post">
		<div>
			<label for="npm">NPM</label> <input type="text"
name="npm" readonly="true" th:value="${student.npm}" th:field="*{npm}"/>
		</div>
		<div>
			<label for="name">Name</label> <input type="text"
name="name" th:value="${student.name}"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

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

class StudentController :

```
    @RequestMapping (value = "/student/update/submit", method =
RequestMethod.POST)
    public String updateSubmit (@ModelAttribute ("student") StudentModel
student, BindingResult br)
    {
       if (br.hasErrors()) {
                studentDAO.updateStudent (student);
                return "success-update";
       }else {
              return "not-found";
       }
    }
```

Tampilan awal :



All Students

No. 1

NPM = 1706106923

Name = rahmaaaaa

GPA = 3.9

Update Data Delete Data

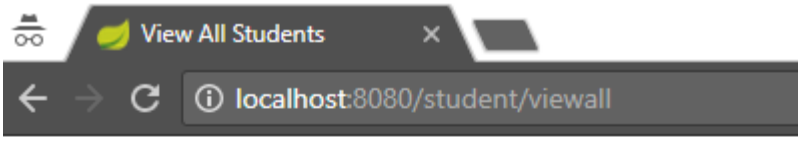
Jika berhasil

Data berhasil diperbarui

Jika gagal npm salah

Student not found

Hasil data setelah berhasil



All Students

No. 1

NPM = 1706106923

Name = rahm

GPA = 3.9

Update Data Delete Data

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?
   Validasi tidak diperlukan karena required bersifat optional.

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 POST dapat tidak memunculkan informasi yaitu value dan parameter yang terhubung dengan query di url dan GET sebaliknya.

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   Tidak bisa karena harus memilih salah satu dari request akan melewati request yang mana.

```
@RequestMapping (value = "/student/update/submit", method =
RequestMethod.POST)

@RequestMapping (value = "/student/update/submit", method =
RequestMethod.GET)
```

Rahmawati Putrianasari | 1706106923

RANGKUMAN:

Kita belajar menggunakan update dan delete menggunakan salah satu method get dan post.