Tutorial 4 Writeup

A. Latihan Menambahkan Delete

- Method deleteStudent di kelas StudentMapper

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

- Method delete pada class StudentController

```
@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";
    }
}
```

1. View all setelah melakukan beberapa insert

All Students

No. 1

NPM = 11111

Name = a

GPA = 3.0

Delete Data

No. 2

NPM = 22222

Name = b

GPA = 2.66

Delete Data

No. 3

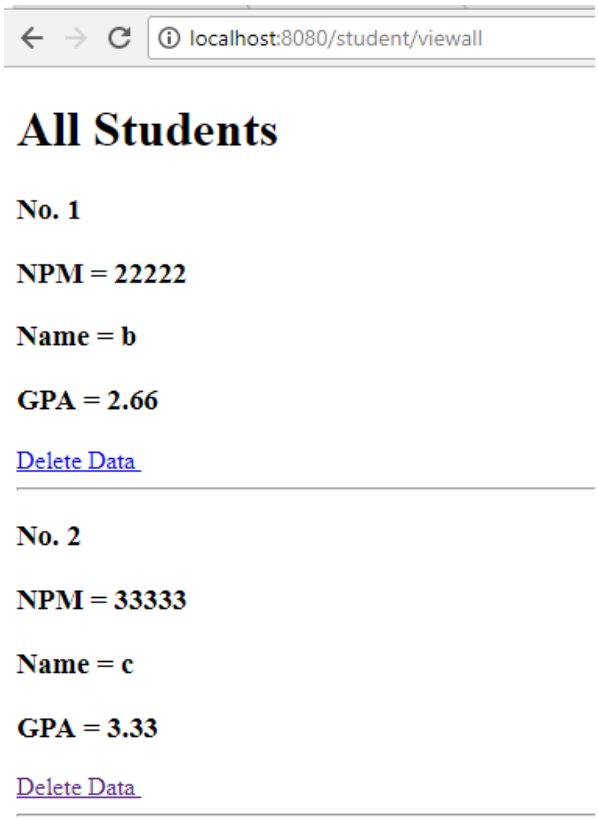
NPM = 33333

Name = c

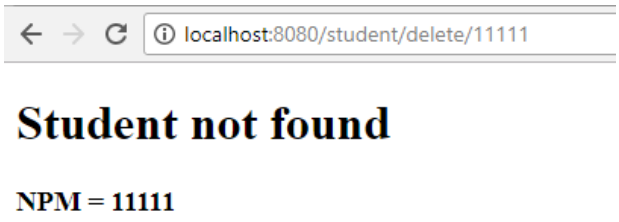
GPA = 3.33

2. Delete student dengan NPM 11111

3. View all student



4. Delete kembali student dengan NPM 11111



B. Latihan Menambahkan Update

- Method updateStudent di kelas studentMapper

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

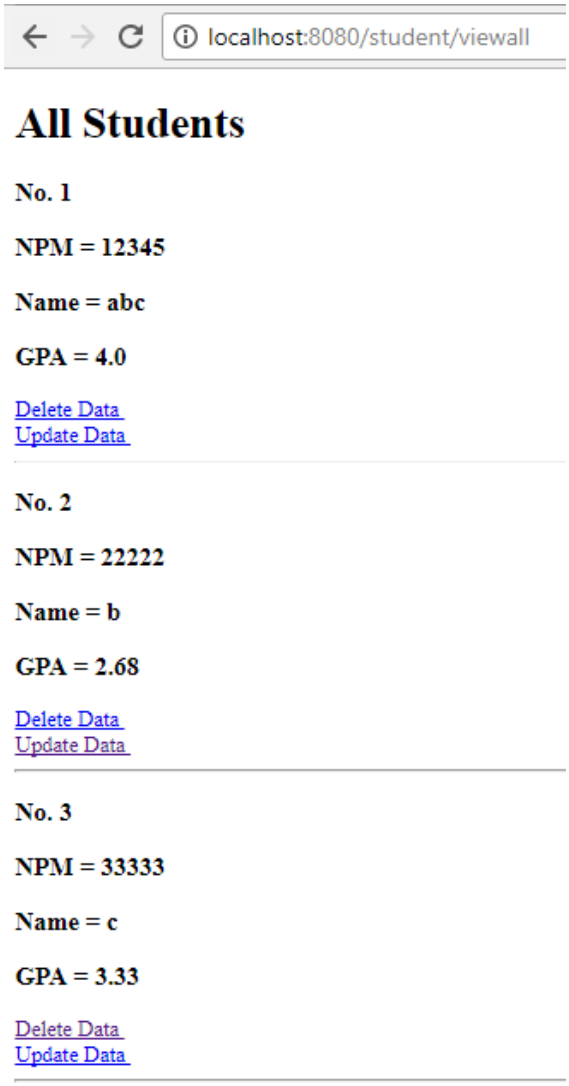
- Method update di kelas StudentController

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

        return "form-update";
    }
}
```

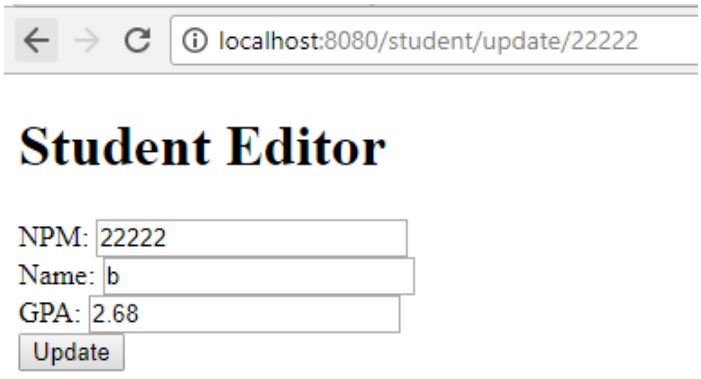
- Method updateSubmit di kelas StudentController

```
@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";
}
```

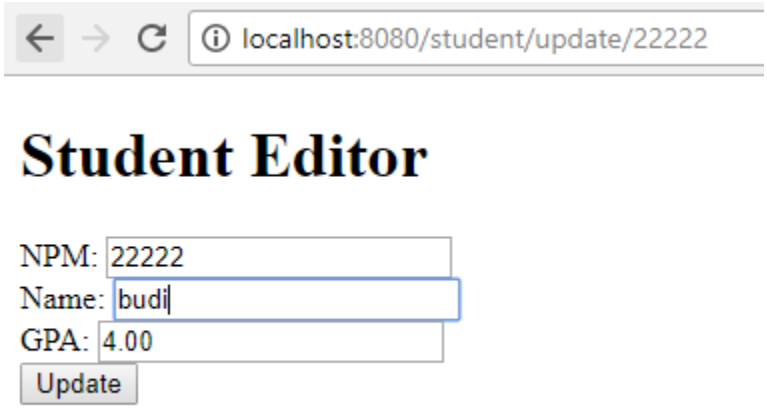
1. Menjalankan view all student



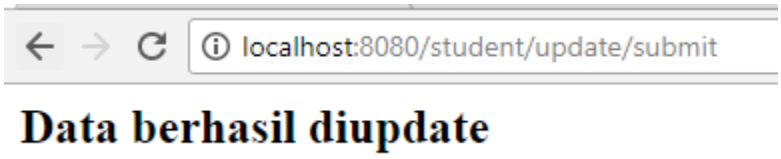
2. Memilih opsi update data student dengan NPM 22222



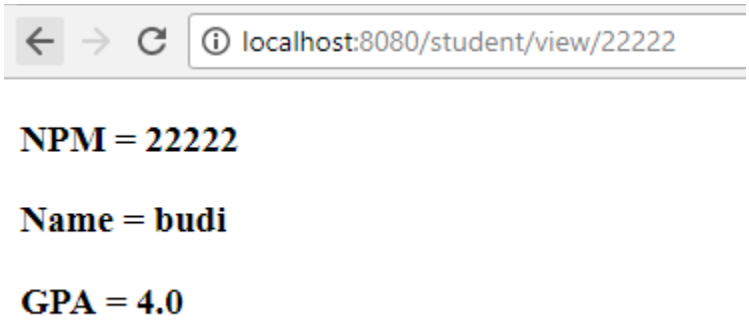
3. Melakukan pengubahan pada data student



4. Notifikasi berhasil mengubah data student



5. Mengecek perubahan yang dilakukan



C. Latihan Menggunakan Object Sebagai Parameter

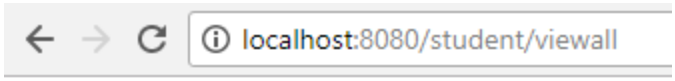
- Handling form menggunakan Thymeleaf dan SpringBoot

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

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

- Method updateSubmit di kelas StudentController dengan menggunakan object sebagai parameter

```
@RequestMapping(value = "/student/update/submit", method = RequestMethod.POST)
public String updateSubmit (@ModelAttribute("student") StudentModel student) {

    studentDAO.updateStudent(student);
    return "success-update";
}
```

1. View all student



All Students

No. 1

NPM = 12345

Name = abc

GPA = 4.0

Delete Data
Update Data

No. 2

NPM = 22222

Name = budi

GPA = 4.0

Delete Data
Update Data

No. 3

NPM = 33333

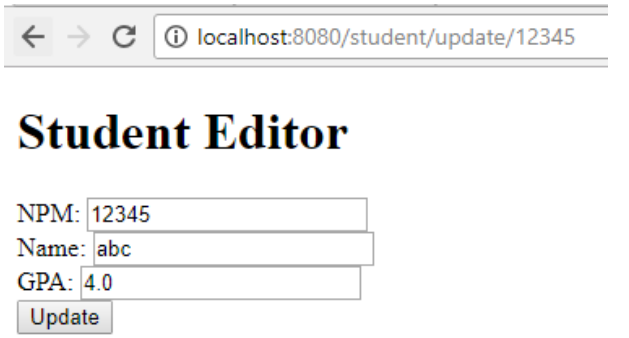
Name = c

GPA = 3.33

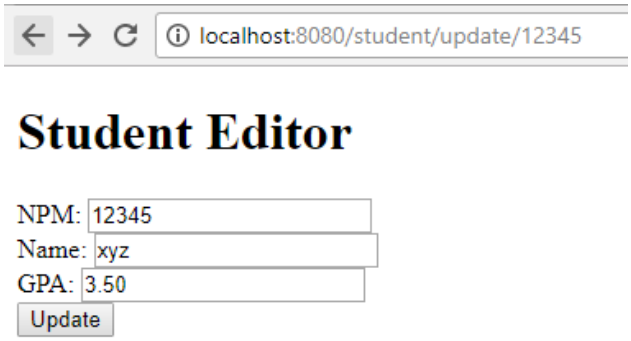
Delete Data
Update Data

---

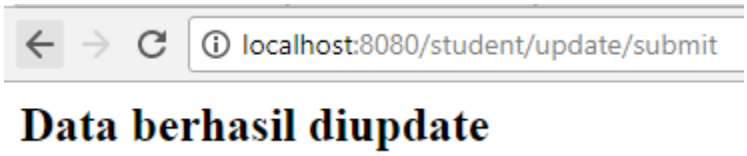
2. Pilih opsi update data mahasiswa dengan NPM 12345



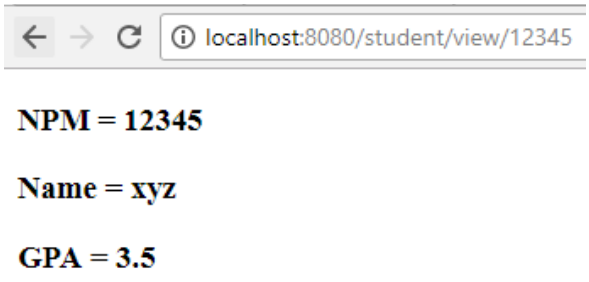
3. Mengubah data mahasiswa



4. Notifikasi telah berhasil mengubah data mahasiswa



5. Mengecek perubahan yang telah dilakukan



D. Pertanyaan

1. Jika menggunakan Object sebagai parameter pada form POST, validasi input required atau optional dapat dilakukan dengan anotasi @Valid dan memasukkan BindingRequest ke dalam parameter. Selain itu, perlu juga dilakukan validasi secara manual dengan menggunakan if-else condition yang sesuai dengan kebutuhan validasi form
2. Karena method POST tidak melakukan passing variable melalui URL, data pada parameter akan cenderung lebih aman dibandingkan method GET. Method POST memiliki action yang berbeda dengan method GET pada tag <form>

   Form dengan method POST:

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

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

   Form dengan method GET:

```
<form action="/student/add/submit" method="get">
    <div>
        <label for="npm">NPM</label> <input type="text" name="npm" />
    </div>
    <div>
        <label for="name">Name</label> <input type="text" name="name" />
    </div>
    <div>
        <label for="gpa">GPA</label> <input type="text" name="gpa" />
    </div>

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

3. Ya, terdapat kemungkinan satu method menerima method GET sekaligus method POST. Jika terdapat kondisi tersebut mungkin saja terdapat alur penanganan dengan menthod request yang berbeda pada kelas Controller program tersebut