

## WRITE-UP

### Pertanyaan

1. Kalau harus validasi backend sepertinya lebih susah dibanding jika menggunakan @RequestParam, karena kalau @RequestParam bisa divalidasi satu-satu di controller sementara kalau pakai @ModelAttribute karena sudah terbungkus jadi satu object kalau submit data yg ga valid keburu kena whitelabel error gitu (?). Bagusnya pakai validasi di frontend aja.
2. Method GET sepertinya lebih cocok untuk proses seperti menambah object baru karena dia nge-get parameternya gitukan dari form. Untuk penanganan kok kayanya ga beda tapi ya GET ama POST? Yg ngefek beda kayanya parameternya pake @RequestParam atau @ModelAttribute
3. Tidak, karena penggunaan method harus dispecify dari awal yang mana yg digunakan dan hanya bisa salah satu aja.

### Method Delete

Di mapper taruh query yg bakal diexecute terus bikin methodnya yg bakal make parameter npm

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

Trus di service database panggil method di mapper + add log buat debug

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

Di controller kasih validasi dulu student yg mau didelete ada ga, kalo ada delete trus direct ke halaman delete, kalo ga ada direct ke halaman not found.

```
@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";  
    }  
}
```

Halaman view all



## All Students

**No. 1**

**NPM = 123**

**Name = Tinky Winky**

**GPA = 3.4**

[Delete Data](#)

---

**No. 2**

**NPM = 456**

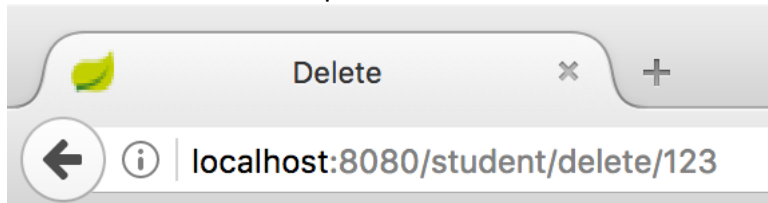
**Name = Chanek Jr.**

**GPA = 3.4**

[Delete Data](#)

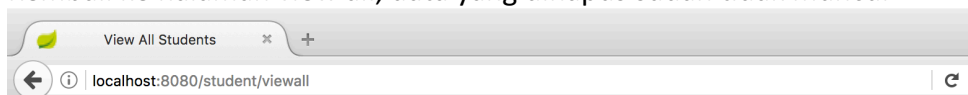
---

Setelah klik Delete Data pada NPM 123



## Data berhasil dihapus

Kembali ke halaman view all, data yang dihapus sudah tidak muncul



## All Students

**No. 1**

**NPM = 456**

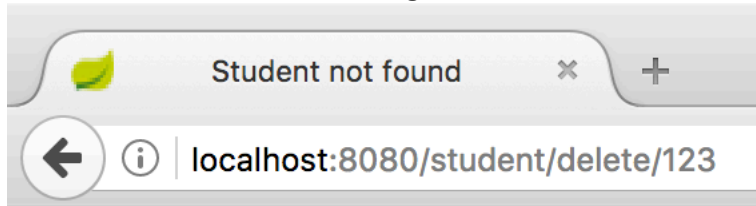
**Name = Chanek Jr.**

**GPA = 3.4**

[Delete Data](#)

---

Kalo dicoba delete NPM 123 lagi, bakal didirect ke halaman not found



# Student not found

**NPM = 123**

## Method Update

Query update di mapper, paramnya ga pake StudentModel karena bentuknya form kaya proses addStudent, jadi yg diperlukan langsung atribut-atributnya

```
@Update("update student set name = #{name}, gpa = ${gpa} where npm = ${npm}")
void updateStudent (@Param("npm") String npm, @Param("name") String name, @Param("gpa") double gpa);
```

Service database

```
@Override
public void updateStudent (String npm, String name, double gpa)
{
    log.info ("update student npm {} ", npm);
    studentMapper.updateStudent(npm, name, gpa);
}
```

Di controller, ada dua mapping yang diperlukan. Pertama, mapping untuk direct ke halaman update form. Pada mapping ini ada validasi juga apakah npm dari student yg mau diupdate itu ada atau tidak.

```
@RequestMapping("/student/update/{npm}")
public String updatePath (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";
    }
}
```

Mapping kedua untuk submit. Requestparam nya sama seperti proses add student.

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

Halaman form-update yang diambil dari proses add dengan sedikit perubahan

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>

<title>Add student</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>

    <h1 class="page-header">Problem Editor</h1>

    <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>

</body>
```

Halaman success-update

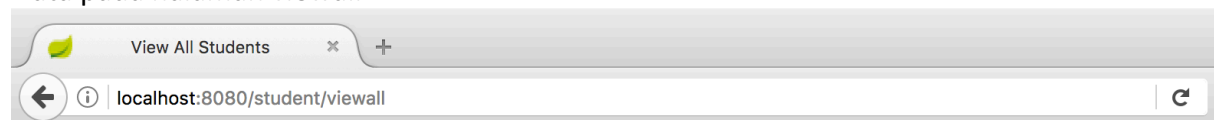
```
<html>
  <head>
    <title>Add</title>
  </head>
  <body>
    <h2>Data berhasil diubah</h2>
  </body>
</html>
```

Pada halaman viewall saya tambahkan link untuk update data

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>View All Students</title>
  </head>
  <body>
    <h1>All Students</h1>

    <div th:each="student, iterationStatus: ${students}">
      <h3 th:text="No. ' + ${iterationStatus.count}">No. 1</h3>
      <h3 th:text="NPM = ' + ${student.npm}">Student NPM</h3>
      <h3 th:text="Name = ' + ${student.name}">Student Name</h3>
      <h3 th:text="GPA = ' + ${student.gpa}">Student GPA</h3>
      <a th:href="/student/delete/' + ${student.npm}">Delete Data</a>
      <a th:href="/student/update/' + ${student.npm}">Update Data</a>
      <hr/>
    </div>
  </body>
</html>
```

Data pada halaman viewall



## All Students

**No. 1**

**NPM = 456**

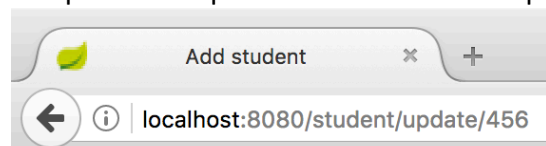
**Name = Chanek Jr.**

**GPA = 3.4**

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

---

Klik pada link Update Data dan lakukan perubahan pada data (gpa)



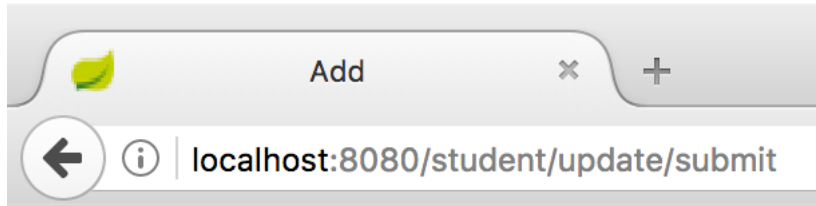
## Problem Editor

NPM

Name

GPA

Klik submit



## Data berhasil diubah

Kembali ke halaman viewall, perubahan data telah tersimpan



## All Students

**No. 1**

**NPM = 456**

**Name = Chanek Jr.**

**GPA = 3.55**

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

---

### Method Update dengan Parameter Object

Parameter method di mapper diganti jadi StudentModel

```
@Update("update student set name = #{name}, gpa = ${gpa} where npm = ${npm}")
void updateStudent (StudentModel student);
```

Di ServiceDatabase juga diganti parameter methodnya

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

Controller yang diganti hanya mapping submitnya saja, parameternya diganti jadi StudentModel

```
@RequestMapping("/student/update/submit")
public String updateSubmit (
    @ModelAttribute StudentModel student)
{
    studentDAO.updateStudent (student);
    return "success-update";
}
```

perubahan di form-update.html diantaranya nambahin th:object di <form>, nambah th:field di setiap kolom input

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>

<title>Add student</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>

    <h1 class="page-header">Problem Editor</h1>

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

</body>
```

Halaman view all data



## All Students

No. 1

NPM = 456

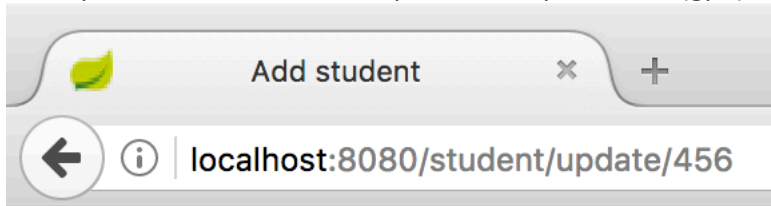
Name = Chanek Jr.

GPA = 3.55

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

---

Klik Update Data lalu lakukan perubahan pada data (gpa)



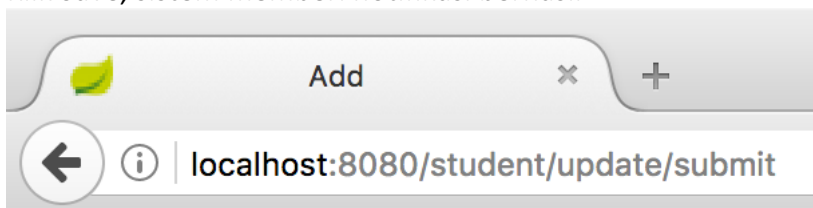
## Problem Editor

NPM

Name

GPA

Klik Save, sistem memberi notifikasi berhasil



## Data berhasil diubah

Kembali ke halaman view all, perubahan data berhasil tersimpan



## All Students

No. 1

NPM = 456

Name = Chanek Jr.

GPA = 3.75

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

---