

# Write-Up Tutorial 4 APAP

Nama : Lintang Matahari Hasani

NPM : 1506689231

---

## Latihan Menambahkan Delete

### Method delete pada Class StudentController

```
93 @RequestMapping("/student/delete/{npm}")
94 public String delete (Model model, @PathVariable(value = "npm") String npm)
95 {
96     StudentModel student = studentDAO.selectStudent(npm);
97
98     if(student == null) {
99         return "not-found";
100     }
101
102     studentDAO.deleteStudent (npm);
103     return "delete";
104 }
```

Method ini menerima parameter String NPM. Kemudian, method akan melakukan validasi dengan memeriksa apakah terdapat Student dengan NPM tersebut pada database. Jika tidak ada, method mengembalikan view “not-found”. Jika ada, method menghapus data Student tersebut dan mengembalikan view “delete”

### Method deleteStudent pada Interface StudentMapper

```
26 @Delete("DELETE from student WHERE npm = #{npm}")
27 void deleteStudent(String npm);
28
```

Method ini menggunakan SQL query DELETE from student WHERE npm = #{npm}. Method menerima parameter String NPM untuk kemudian dimasukkan ke dalam query dan melakukan penghapusan data Student pada database

### Method deleteStudent pada Class StudentServiceDatabase

```
44 @Override
45 public void deleteStudent (String npm)
46 {
47     studentMapper.deleteStudent(npm);
48     Log.info("student " + npm + " deleted");
49 }
50
```

Method ini memanggil method deleteStudent pada Interface StudentMapper. Terdapat log.info pada method ini untuk mempermudah melakukan debugging pada fitur delete

## Menjalankan fitur delete

- Menjalankan viewall untuk menampilkan data student yang ada di database

← → ↻ localhost:8080/student/viewall

### All Students

No. 1

NPM = 12345

Name = Joni

GPA = 3.5

[Update Data](#)

[Delete Data](#)

---

No. 2

NPM = 12346

Name = George

GPA = 3.2

[Update Data](#)

[Delete Data](#)

---

No. 3

NPM = 12347

Name = Smith

GPA = 2.5

- Menjalankan fitur delete pada student dengan NPM 12346

← → ↻ localhost:8080/student/delete/12346

**Data berhasil dihapus**

No. 2

NPM = 12346

Name = George

GPA = 3.2

[Update Data](#)

[Delete Data](#)

---

- Menjalankan fitur delete pada student tersebut untuk kedua kalinya

← → ↻ localhost:8080/student/delete/1246

### Student not found

NPM = 1246

- Menjalankan fitur delete untuk NPM yang tidak ada pada database

← → ↻ localhost:8080/student/delete/1000000

### Student not found

NPM = 1000000

# Latihan Menambahkan Update

## Method update pada Class StudentController

```
107@ @RequestMapping("/student/update/{npm}")
108 public String update (Model model, @PathVariable(value = "npm") String npm)
109 {
110     StudentModel student = studentDAO.selectStudent(npm);
111     if(student == null) {
112         return "not-found";
113     }
114     model.addAttribute("student", student);
115     return "form-update";
116 }
117 --
```

Method ini menerima parameter model dan String NPM. Sebagaimana method delete pada Latihan sebelumnya, pada method ini melakukan pemeriksaan apakah NPM yang dimasukkan terdapat pada database. Jika NPM tidak ditemukan pada database, method mengembalikan view "not-found". Jika NPM ditemukan, method akan memasukkan object Student yang ada ke model dan mengembalikan view "form-update"

## Method updateSubmit pada Class StudentController

```
119@ @RequestMapping(value = "/student/update/submit", method = RequestMethod.POST)
120 public String updateSubmit (
121     @RequestParam ( value = "npm", required = false ) String npm ,
122     @RequestParam ( value = "name", required = false ) String name,
123     @RequestParam ( value = "gpa", required = false ) double gpa
124 )
125 {
126     StudentModel updatedStudent = new StudentModel(npm, name, gpa);
127     studentDAO.updateStudent(updatedStudent);
128     return "success-update";
129 }
130 --
```

Method ini menerima parameter String NPM, String Name, dan Double GPA yang diambil dari pengisian form pada view "form-update". Kemudian method akan membuat object baru updatedStudent yang merepresentasikan Student dengan data yang telah diubah dan memanggil method updateStudent

## Method updateStudent pada Interface StudentMapper

```
29@ @Update("UPDATE student SET name = #{name}, gpa = #{gpa} WHERE npm = #{npm}")
30 void updateStudent(StudentModel student);
31 }
32 --
```

Method ini menerima parameter StudentModel dan menjalankan query UPDATE student SET name = #{name}, gpa = #{gpa} WHERE npm = #{npm}

## Method updateStudent pada Class StudentServiceDatabase

```
51@ @Override
52 public void updateStudent(StudentModel student) {
53     studentMapper.updateStudent(student);
54     log.info("student " + student.getNpm() + " updated");
55 }
56 --
```

Method ini menerima StudentModel dan memanggil method updateStudent dari studentMapper. Terdapat log.info untuk mempermudah debugging fitur update

## Menjalankan fitur update

- Menjalankan viewall untuk menampilkan data student yang ada di database

← → ↻ localhost:8080/student/viewall

### All Students

No. 1

NPM = 12345

Name = Joni

GPA = 3.5

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

---

No. 2

NPM = 12347

Name = Smith

GPA = 2.5

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

---

- Menjalankan update untuk Student dengan NPM 12347

← → ↻ localhost:8080/student/update/12347

← → ↻ localhost:8080/student/update/submit

**Data berhasil diubah**

### Update Student

NPM 12347

Name Jedediah Smith

GPA 2.5

- Menjalankan view dengan NPM Student tersebut. Dapat dilihat bahwa update berhasil

← → ↻ localhost:8080/student/view/12347

NPM = 12347

Name = Jedediah Smith

GPA = 2.5

- Menjalankan update untuk Student dengan NPM yang tidak valid/tidak terdapat di database

← → ↻ localhost:8080/student/update/10000

### Student not found

NPM = 10000

# Latihan Menggunakan Objek Sebagai Parameter

## Method update pada Class StudentController

```
132 @RequestMapping("/student/update/{npm}")
133 public String update (Model model, @PathVariable(value = "npm") String npm)
134 {
135     StudentModel student = studentDAO.selectStudent(npm);
136     if(student == null) {
137         return "not-found";
138     }
139     model.addAttribute("student", student);
140     return "form-update-object-parameter";
141 }
142
```

Method ini serupa dengan method update pada Latihan sebelumnya (Latihan Membuat Update), akan tetapi jika NPM ditemukan method ini mengembalikan view “form-update-object-parameter”

## Method updateSubmit pada Class StudentController

```
144 @RequestMapping(value = "/student/update/submit", method = RequestMethod.POST)
145 public String updateSubmit (StudentModel student)
146 {
147     StudentModel updatedStudent = student;
148     studentDAO.updateStudent(updatedStudent);
149     return "success-update";
150 }
151
152
```

Method ini menerima parameter StudentModel yang ingin diubah. Method kemudian akan memanggil method updateStudent dari Class StudentServiceDatabase

*Keterangan* : method-method update lainnya pada Class StudentServiceDatabase dan Interface StudentMapper sama seperti yang diimplementasikan sebelumnya (Latihan Menambahkan Update)

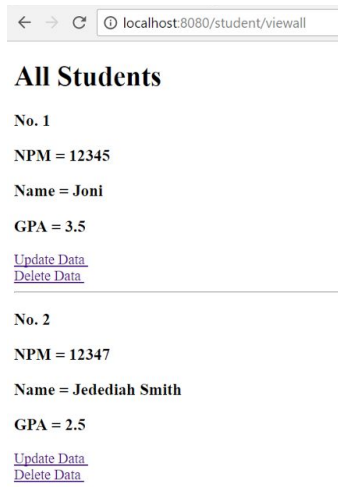
## Tampilan view form update object parameter

```
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 Student</h1>
13 <form action="/student/update/submit" method="post" th:object="${student}">
14 <div>
15 <label for="npm">NPM</label> <input type="text" name="npm" readonly="true" th:field="${student.npm}" />
16 </div>
17 <div>
18 <label for="name">Name</label> <input type="text" name="name" th:field="${student.name}" />
19 </div>
20 <div>
21 <label for="gpa">GPA</label> <input type="text" name="gpa" th:field="${student.gpa}" />
22 </div>
23
24 <div>
25 <button type="submit" name="action" value="update">Update</button>
26 </div>
27 </form>
28
29 </body>
30
31 </html>
32
```

*Keterangan* : Untuk menjalankan fitur update dengan object parameter, method update dan updateSubmit yang terkait latihan sebelumnya (Latihan Membuat Update) perlu di-“comment” terlebih dahulu dikarenakan method tersebut menggunakan path yang sama. Untuk menjalankan fitur update pada latihan sebelumnya, harap “comment” method terkait latihan update dengan object parameter.

## Menjalankan fitur update

- Menjalankan viewall untuk menampilkan data student yang ada di database



← → ↻ localhost:8080/student/viewall

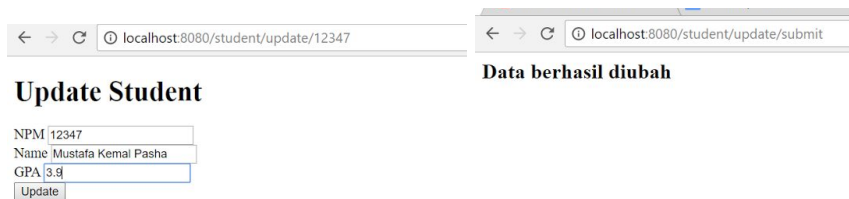
### All Students

No. 1  
NPM = 12345  
Name = Joni  
GPA = 3.5  
[Update Data](#)  
[Delete Data](#)

---

No. 2  
NPM = 12347  
Name = Jedediah Smith  
GPA = 2.5  
[Update Data](#)  
[Delete Data](#)

- Menjalankan update untuk Student dengan NPM 12347



← → ↻ localhost:8080/student/update/12347

### Update Student

NPM 12347  
Name Mustafa Kemal Pasha  
GPA 3.9

← → ↻ localhost:8080/student/update/submit

**Data berhasil diubah**

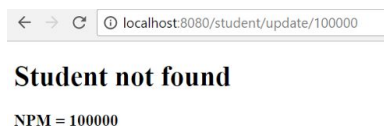
- Menjalankan view dengan NPM Student tersebut. Dapat dilihat bahwa update berhasil



← → ↻ localhost:8080/student/view/12347

NPM = 12347  
Name = Mustafa Kemal Pasha  
GPA = 3.9

- Menjalankan update untuk Student dengan NPM yang tidak valid/tidak terdapat di database



← → ↻ localhost:8080/student/update/100000

### Student not found

NPM = 100000

## Pertanyaan

1. *Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required? Apakah validasi Diperlukan?*

Validasi diperlukan untuk menangani isian yang kosong pada form update di view “form-update-object-parameter” karena jika salah satu/sebagian isian form update kosong, aplikasi akan mengembalikan laman error. Validasi terhadap input yang wajib diisi dilakukan dengan menambahkan atribut `required="true"` pada setiap tag `<input>`

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?*

Umumnya form submit menggunakan POST method karena pertimbangan keamanan data. Saat mengirim form submit menggunakan GET method, isian pada form dapat dilihat pada path sehingga beresiko terjadi pencurian data. Dengan menggunakan POST method, isian pada form tidak dapat dilihat pada path.

Diperlukan penanganan berbeda pada body method di controller ketika form dikirim dengan menggunakan method yang berbeda.

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

Ya, satu method dapat menerima lebih dari satu jenis request method. Pada Anotasi `@RequestMapping` dapat ditambahkan `method = {RequestMethod.GET, RequestMethod.POST}`

## Lesson Learned

Pada tutorial 4 ini, saya belajar banyak mengenai menggunakan Database SQL pada Project Spring Boot serta cara melakukan Debugging. Pada tutorial ini, SQL Query dibuat di Interface StudentMapper sesuai dengan parameter yang diterima method-method (misal: npm). Hal ini ditunjukkan dengan adanya anotasi `@Select`, `@Update` dan `@Delete` di bagian atas method-method pada interface tersebut. Kemudian Query yang dibuat oleh Interface StudentMapper akan dijalankan di database dan mengembalikan data sesuai query tersebut.

Pada tutorial ini, saya juga belajar mengenai cara membuat method yang menerima parameter Object untuk menangani pengisian form pada view. Terdapat penanganan khusus untuk

method tersebut, antara lain: menambahkan anotasi `th:object="${nama_object}"` pada tag `<form>` di view form, menambahkan anotasi `th:field="${nama_object.nama_field}"` pada setiap tag `<input>` di view form, dan mengubah method pada controller menjadi hanya menerima object sebagai parameter.

Terkait penggunaan method POST/GET, khususnya untuk method yang menerima parameter object, perlu dipertimbangkan melakukan validasi pada form di view. Salah satu langkah untuk melakukan validasi input adalah dengan menambahkan atribut `required="true"` pada tag `<input>`

Selain mempelajari penggunaan database dan penanganan method yang menerima parameter object pada project Spring Boot, saya juga belajar menggunakan `log.info` untuk melakukan debugging pada method-method yang digunakan.