
Tutorial 4

Nama : Sofyan Hadi

NPM : 1706107011

-
1. Jelaskan apa saja hal yang Anda pelajari dari tutorial ini.
Menggunakan database sebagai data yang digunakan pada Spring Boot. Melakukan Input, delete, dan update data dari database.

2. Jawab pertanyaan pada bagian Pertanyaan dan penjelasannya.

Pertanyaan

- a. 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.
Ya diperlukan karena untuk mengecek nilai dari model sesuai atau tidak.

- b. Menurut Anda, mengapa form submit biasanya menggunakan POST method dibanding GET method? Penggunaan method POST lebih aman karena data tidak akan terlihat pada URL yang akan diakses.

Apakah perlu penanganan berbeda di header atau body method di controller jika form di post dikirim menggunakan method berbeda? Ya, perlu

- c. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST? Tidak mungkin. Hanya bisa salah satu.

3. Method yang dibuat pada Latihan

- Menambahkan Delete, jelaskan
 - a. Menambakan baris pada viewall.html

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

- b. Menambahkan method delete pada interface StudentMapper

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

- c. Menambahkan method delete pada StudentServiceDatabase

```
@Override  
public void deleteStudent (String npm)  
{  
    studentMapper.deleteStudent(npm);  
}
```

- d. Pada class StudentContoller ditambahkan method delete seperti dibawah ini

```
@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";  
    }  
}
```

-
- e. Mencoba melakukan delete pada data nomor 2

No. 2

NPM = 11212121

Name = Clay1222

GPA = 3.12

[Delete Data](#)

No. 3

NPM = 124

Name = Chanek

GPA = 3.5

[Delete Data](#)

Saat di hapus akan muncul halaman ini.



Data berhasil dihapus

- f. saat melihat semua data nomor 2 akan terhapus

No. 2

NPM = 124

Name = Chanek

GPA = 3.5

[Delete Data](#)

- Menambahkan Update, jelaskan
 - a. Menambahkan method update pada interface StudentMapper

```
@Update("UPDATE student SET name = #{name}, gpa = #{gpa} WHERE npm = #{npm}")
void updateStudent(StudentModel student);
```
 - b. Menambahkan method update atau mengimplementasikan pada StudentService

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

- c. Menambahkan method update pada class StudentServiceDatabase

```
@Override
public void updateStudent(StudentModel student) {
    // TODO Auto-generated method stub
    studentMapper.updateStudent(student);
}
```

- d. Menambahkan baris di viewall.html

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

- e. Membuat form-update.html dan ubah data data sebagai berikut.

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

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

    <h1 class="page-header">Update Student</h1>

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

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

- f. Menambahkan method pada controller

```
@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.setNpm(npm);
    student.setName(name);
    student.setGpa(gpa);

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

- g. Mencoba melakukan update data

No. 2

NPM = 121212

Name = Sia

GPA = 3.33

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

- h. Memasukan data baru

← → ↻ ⓘ localhost:8080/student/update/121212

Update Student

NPM

Name

GPA

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

Data berhasil diUpdate

- i. Hasil dari data yang di update

No. 2

NPM = 121212

Name = Sisi

GPA = 3.4

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

- Menggunakan Object Sebagai Parameter, jelaskan

- a. Melakukan edit pada controller update

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

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

- b. Mencoba melakukan update data

No. 2

NPM = 121212

Name = Sia

GPA = 3.33

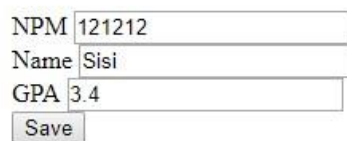
[Delete Data](#) [Update Data](#)

- c. Mengedit data



← → ↻ ⓘ localhost:8080/student/update/121212

Update Student



NPM

Name

GPA



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

Data berhasil diUpdate

- d. Hasil dari data yang diupdate

No. 2

NPM = 121212

Name = Sisi

GPA = 3.4

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