

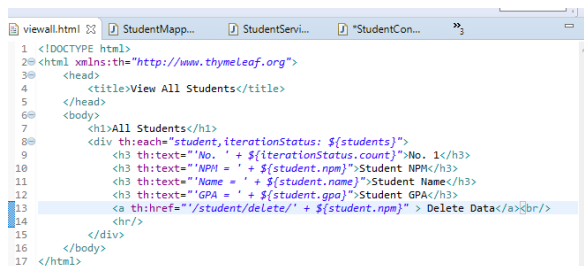
Tutorial 4

Lesson Learned

Dalam membuat aplikasi enterprise dibutuhkan data yang persistence, oleh karena itu data disimpan pada database, untuk melakukan operasi pada data yang ada di database tersebut kita menggunakan mapper. SpringBoot dan Thymeleaf memungkinkan agar method pada controller menerima parameter berupa objek dari suatu model.

Latihan Menambahkan Delete

1. Pada viewall.html tambahkan Delete Data Di dalam div dengan th:each



```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>View All Students</title>
5 </head>
6 <body>
7 <div>
8 <div th:each="student, iterationStatus: ${students}">
9 <div th:text="No. ' + ${iterationStatus.count}>No. 1</div>
10 <div th:text="NPM = ' + ${student.npm}>Student NPM</div>
11 <div th:text="Name = ' + ${student.name}>Student Name</div>
12 <div th:text="GPA = ' + ${student.gpa}>Student GPA</div>
13 <a th:href="/student/delete/' + ${student.npm}" > Delete Data</a><br/>
14 </div>
15 </div>
16 </body>
17 </html>
```

2. Tambahkan method deleteStudent yang ada di class StudentMapper

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

3. Lengkapi method deleteStudent yang ada di class StudentServiceDatabase

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

4. Lengkapi 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) {
        studentDAO.deleteStudent (npm);
        return "delete";
    } else {
        model.addAttribute ("npm", npm);
        return "not-found";
    }
}
```

Fadzil Rizki
1506723225

5. Jalankan Spring Boot dan coba test program Anda.

← → ↻ localhost:8080/student/viewall

All Students

No. 1

NPM = 123

Name = Chanek

GPA = 3,6

[Delete Data](#)

No. 2

NPM = 124

Name = Chanek Jr.

GPA = 3,4

[Delete Data](#)

← → ↻ localhost:8080/student/delete/123

Student not found

NPM = 123

← → ↻ localhost:8080/student/viewall

All Students

No. 1

NPM = 124

Name = Chanek Jr.

GPA = 3,4

[Delete Data](#)

← → ↻ localhost:8080/student/delete/123

Data berhasil dihapus

Latihan Menambahkan Update

1. Tambahkan method updateStudent pada class StudentMapper

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

2. Tambahkan method updateStudent pada interface StudentService

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

3. Tambahkan implementasi method updateStudent pada class StudentServiceDatabase.

```
@Override  
public void updateStudent (StudentModel student)  
{  
    studentMapper.updateStudent(student);  
    log.info("student " + student.getNpm() + " updated" );  
}
```

4. Tambahkan link Update Data pada viewall.html

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

5. Copy view form-add.html menjadi form-update.html.

```
<!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 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>
```

6. Copy view success-add.html menjadi success-update.html.

```
1 <html>  
2   <head>  
3     <title>Update</title>  
4   </head>  
5   <body>  
6     <h2>Data berhasil diupdate</h2>  
7   </body>  
8 </html>
```

7. Tambahkan method update pada class StudentController

```
@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 {
        return "not-found";
    }
}
```

8. Tambahkan method updateSubmit pada class 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) {

    studentDAO.updateStudent(new StudentModel(npm, name, gpa));
    return "success-update";
}
```

9. Jalankan Spring Boot dan coba test program Anda.

← → ↻ localhost:8080/student/viewall

All Students

No. 1

NPM = 124

Name = Chanek Jr.

GPA = 3.7

[Delete Data](#)

[Update Data](#)

← → ↻ localhost:8080/student/viewall

← → ↻ localhost:8080/student/update/124

Update Editor

NPM 124
Name Chanek Jr.
GPA 3.99

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

Data berhasil diupdate

All Students

No. 1

NPM = 124

Name = Chanek Jr.

GPA = 3.99

[Delete Data](#)

[Update Data](#)

Latihan Menggunakan Object Sebagai Parameter

1. Menambahkan th:object="\${student}" pada tag <form>di view

```
<form action="/student/update/submit" th:object="${student}" method="POST">
    <div>
```

2. Menambahkan th:field="*{[nama_field]}" pada setiap input

```
    <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>
```

3. Ubah method updateSubmit pada StudentController yang hanya menerima parameter berupa StudentModel

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

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

4. Tes lagi aplikasi Anda.

← → ↻ localhost:8080/student/viewall

All Students

No. 1

NPM = 124

Name = Chanek Jr.

GPA = 3.9

[Delete Data](#)

[Update Data](#)

← → ↻ localhost:8080/student/update/124

Update Editor

NPM 124

Name Chanek Jr.

GPA 3.7

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

Data berhasil diupdate

← → ↻ localhost:8080/student/viewall

All Students

No. 1

NPM = 124

Name = Chanek Jr.

GPA = 3.7

[Delete Data](#)

[Update 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? Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.
Dengan membuat validasi @Valid pada parameter method.
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 method POST sudah lebih baik dari segi keamanan dibanding method GET dan method POST bisa menampung data lebih banyak dibanding method GET .
3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
Mungkin bisa, tetapi lebih baik jika hanya menggunakan satu jenis request untuk satu method.