TUGAS TUTORIAL 4

ARSITEKTUR

DAN PEMROGRAMAN APLIKASI PERUSAHAAN

Nama : Julio Muhammad Pranadano

NPM : 1706106785

FAKULTAS ILMU KOMPUTER

UNIVERSITAS INDONESIA

DEPOK

2018

Latihan Menambahkan Delete

1. Tambahkan kode berikut pada viewall.html

```
<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>
        <hr/>
    </div>
</body>
```

2. Tambahkan method deleteStudent yang ada di class StudentMapper

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

Method deleteStudent pada class StudentMapper akan mengeksekusi query sql dengan parameter yang diberikan yaitu npm.

3. Lengkapi method deleteStudent yang ada di class StudentServiceDatabase

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

Method delete student akan memanggil method deleteStudent pada StudentMapper untuk mengeksekusi delete tersebut.

4. Lengkapi method delete pada class StudentController dengan menambahkan fitur validasi

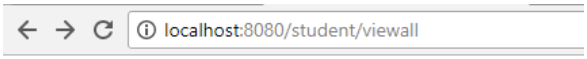
```
@RequestMapping("/student/delete/{npm}")
public String delete (Model model, @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent(npm);
    if(student == null)
    {
        model.addAttribute("npm",npm);
        return "not-found";
    }
    studentDAO.deleteStudent (npm);

    return "delete";
}
```

Method dalam StudentController akan memanggil method deleteStudent dengan mengirim parameter yang dimasukkan yaitu npm dari class

StudentService dimana class StudentService diimplementasikan oleh class StudentServiceDatabase dan didalam class tersebut membuat object StudentMapper dimana student mapper yang memiliki query langsung ke database.

5. Jalankan Springboot app dan lakukan beberapa insert



All Students

No. 1

NPM = 12345

Name = Danu

GPA = 4.5

Delete Data
Update Data

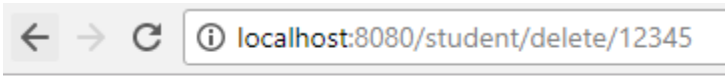
No. 2

NPM = 1706106785

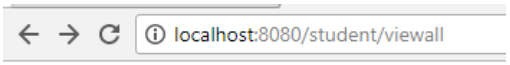
Name = Julio Muhammad Pranadano

GPA = 3.6

Delete Data
Update Data



Data berhasil dihapus



All Students

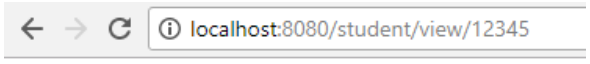
No. 1

NPM = 1706106785

Name = Julio Muhammad Pranadano

GPA = 3.6

Delete Data
Update Data



Student not found

NPM = 12345

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. Jangan lupa tambahkan log info pada fitur ini.

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

4. Tambahkan link Update Data pada viewall.html

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

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 Student</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="npm">NPM</label> <input type="text" name="gpa" th:value="${student.gpa}"/>
        </div>

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

</body>
```

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

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

7. Tambahkan method **update** pada class **StudentController** dan tambahkan fitur validasi untuk fitur ini.

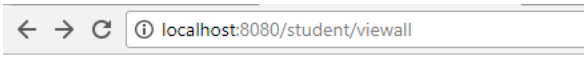
```
@RequestMapping("/student/update/{npm}")
public String update (Model model, @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent(npm);
    if(student == null)
    {
        model.addAttribute("npm",npm);
        return "not-found";
    }
    model.addAttribute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

8. Tambahkan method **updateSubmit** pada class **StudentController** dan menambahkan method post untuk request mappingnya dan requestParam berupa npm, name dan gpa sehingga method update dapat di proses dan data dapat dimasukkan ke dalam database.

```
@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";    }
}
```

9. Jalankan Spring Boot dan test program anda



All Students

No. 1

NPM = 123456789

Name = Test123

GPA = 4.5

Delete Data
Update Data

No. 2

NPM = 1706106785

Name = Julio Muhammad Pranadano

GPA = 3.6

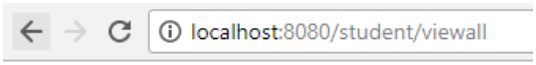
Delete Data
Update Data

Update Student

NPM 123456789
Name Test123
NPM 4.5
Save

Update Student

NPM 123456789
Name Test456
NPM 3.5
Save



All Students

No. 1

NPM = 123456789

Name = Test456

GPA = 3.5

Delete Data
Update Data

No. 2

NPM = 1706106785

Name = Julio Muhammad Pranadano

GPA = 3.6

Delete Data
Update Data

Latihan Menggunakan Object Sebagai Parameter

Berikut ini method submit akan menerima parameter berupa model StudentModel, berikut langkahnya :

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

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

2. Menambahkan th:field=”*{nama_field}” pada setiap input

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

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

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

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

    return "success-update";   }
}
```

Dengan mengirim parameter berupa object StudentModel, pada view akan ditangkap sebagai model dengan th:object dan masing-masing fieldnya di keluarkan dengan th:field

Pertanyaan

Beberapa pertanyaan yang perlu Anda jawab:

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.

   Jawab : Jika mengirimkan parameter berupa object pada form POST maka view thymeleaf yang akan melakukan validasi dan yang akan mengirimkan pesan error atau tidaknya.

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?

   Jawab : Karena, isi dari parameter yang dimasukkan tidak akan ditampilkan pada URL sehingga akan lebih aman dalam mengirim parameter.

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

   Jawab : Satu method tidak dapat menerima lebih dari satu jenis request method.