Adzkia Husnul Abidat - 1506757680

Write Up tutorial 4

SUMMARY

Pada tutorial minggu ini saya mempelajari cara menggunakan databse dan melakukan debugging dalam project Spring Boot dalam hal ini menglimplementasikan method Delete dan Update. Saya juga mempelajari perbedaan method POST dan GET serta mempelajari Objek sebagai parameter

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.

Penggunaan Object sebagai parameter memang lebih aman, namun validasi tetap diperlukan untuk mendeteksi null. Validasi optional atau required dapat dilakukan di kelas controller pada objek yang dijadikan parameter. Dibuat kondisi apabila form tidak diisi maka mengembalikan pesan error, misalnya.

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?

Form submit umumnya menggunakan method POST untuk menyimpan data yang bersifat privasi atau tidak ingin terlihat dan akan di simpan pada data store, dan dikirimkan melalui JSON, bukan melalui URL seperti pada method GET sehingga lebih aman.

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

Tidak bisa, formulir html hanya bisa mengeksekusi 1 method saja yaitu GET atau POST.

LATIHAN

1. Method Delete

- Menambahkan link pada viewall.html untuk melakukan delete

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

- Menambahkan method **deleteStudent** di class **StudentMapper**

```
<h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
<a th:href="'/student/delete/' + ${student.npm}">Delete Data</a><br/>
```

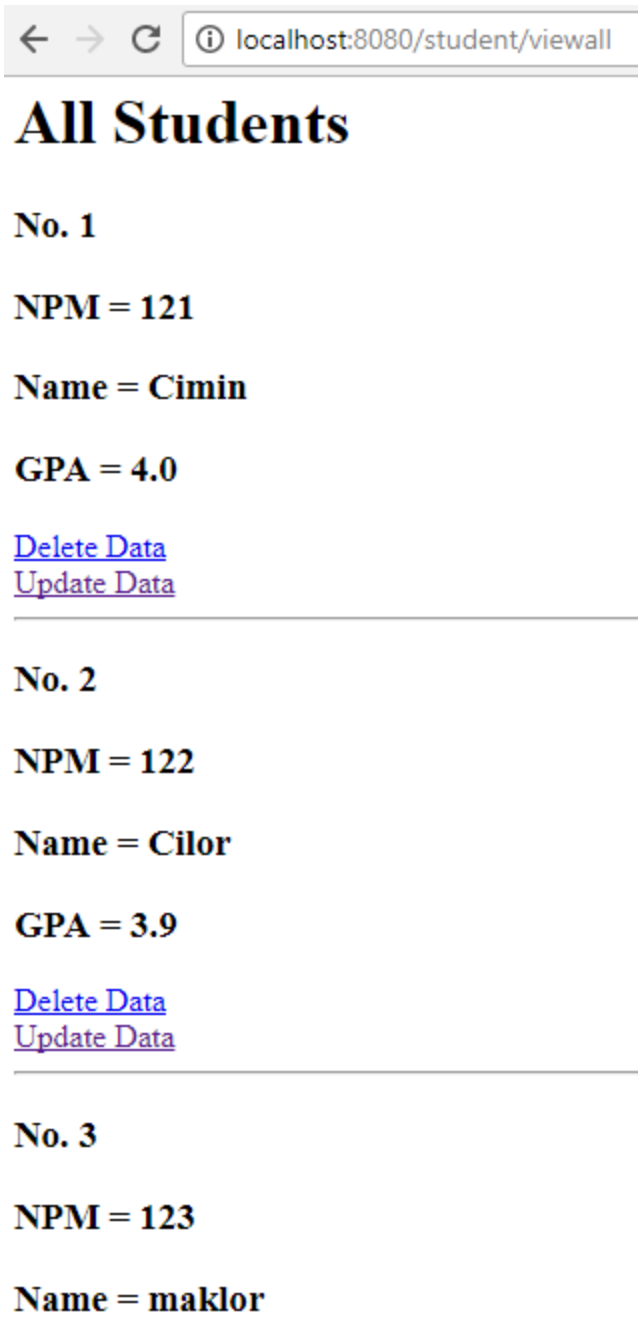
- Melengkapi method **deleteStudent** di class **StudentServiceDatabase** serta menambahkan log.info

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

- Melengkapi method **delete** pada **StudentController** beserta validasinya, pertama memanggil method selectStudent(npm) terlebih dulu dan mengecek apakah data mahasiswa yang akan dihapus ada

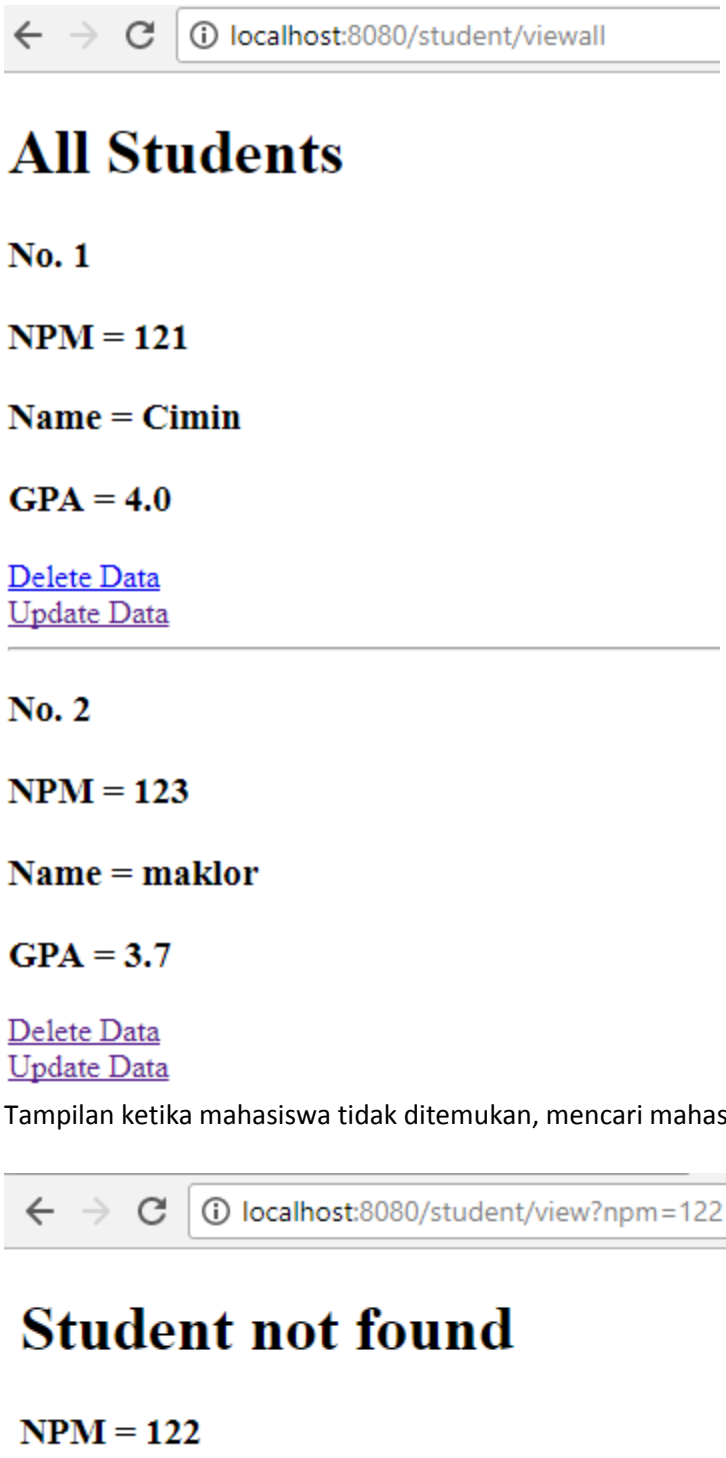
```
@RequestMapping("/student/delete/{npm}")
public String delete (Model model, @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent (npm);
        if(student != null) {
            studentDAO.deleteStudent (npm);

            return "delete";
        }
        else {
            return "not-found";
        }
}
```

- Tampilan saat method delete dijalankan adalah sebagai berikut (setelah ditambahkan data-data mahasiswa)
    - View All



- Delete (Menghapus Data Mahasiswa Cilor, NPM = 122)

- View All setelah salah satu data mahasiswa dihapus, tersisa 2 data



- Tampilan ketika mahasiswa tidak ditemukan, mencari mahasiswa dengan npm 122

2. Method Update

- Menambahkan Method **updateStudent** pada **StudentMapper**

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

- Menambahkan Method **updateStudent** pada interface **StudentService**

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

- Menambahkan Method **updateStudent** pada class **StudentServiceDatabase**

```
    @Override
    public void updateStudent(StudentModel student) {
        Log.info ("update student info");
        studentMapper.updateStudent(student);
```

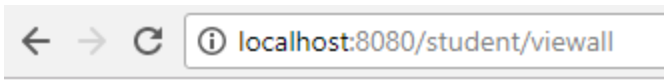
- Menambahkan link update data pada viewall.html

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

- Membuat **form-update.html** berisi form untuk pengisian update name dan gpa sedangkan npm diset read-only sehingga tidak dapat diubah
- Membuat **success-update.html,** berisi pesan sukses ketika update berhasil
- Menambahkan method **updateStudent** pada **StudentController,** merequest mapping ke /studet/update/{npm}, memberikan validasi dan mengembalikan hasil sesuai validasi
- Menambahkan method **updateSubmit** 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 {
            model.addAttribute("npm", npm);
            return "not-found";
        }
}

@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";
}
```

- Tampilan ketika method update dijalankan adalah sebagai berikut :
    - ViewAll



All Students

No. 1

NPM = 121

Name = Cimin

GPA = 4.0

Delete Data
Update Data

No. 2

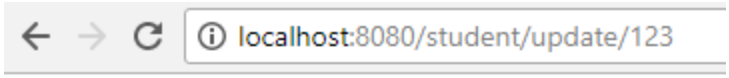
NPM = 123

Name = maklor

GPA = 3.7

Delete Data
Update Data

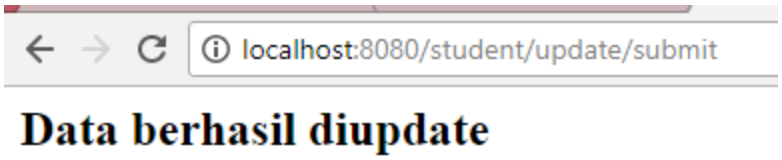
- Update



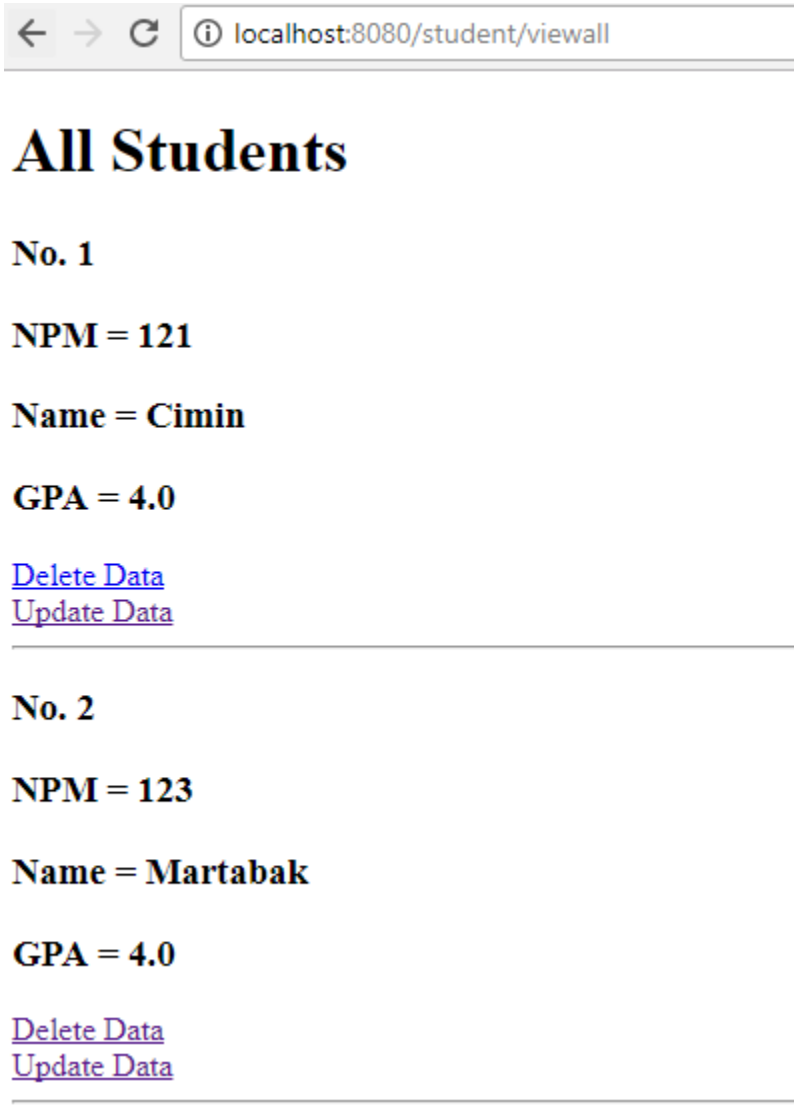
Update Student Data

NPM 123
Name maklor
GPA 3.7
Update

-



- Tampilan ViewAll setelah update dilakukan (Maklor berubah menjadi Martabak)



3. Method Object Sebagai Parameter

`Mengubah updateSubmit pada StudentController sehingga hanya menerima parameter student.`

```
@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 {
            model.addAttribute("npm", npm);
            return "not-found";
        }
}

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

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

Mengubah **Form-update.html** dengan menambahkan :

```
th:action="@{/student/update/submit}" th:object="${student}">
```

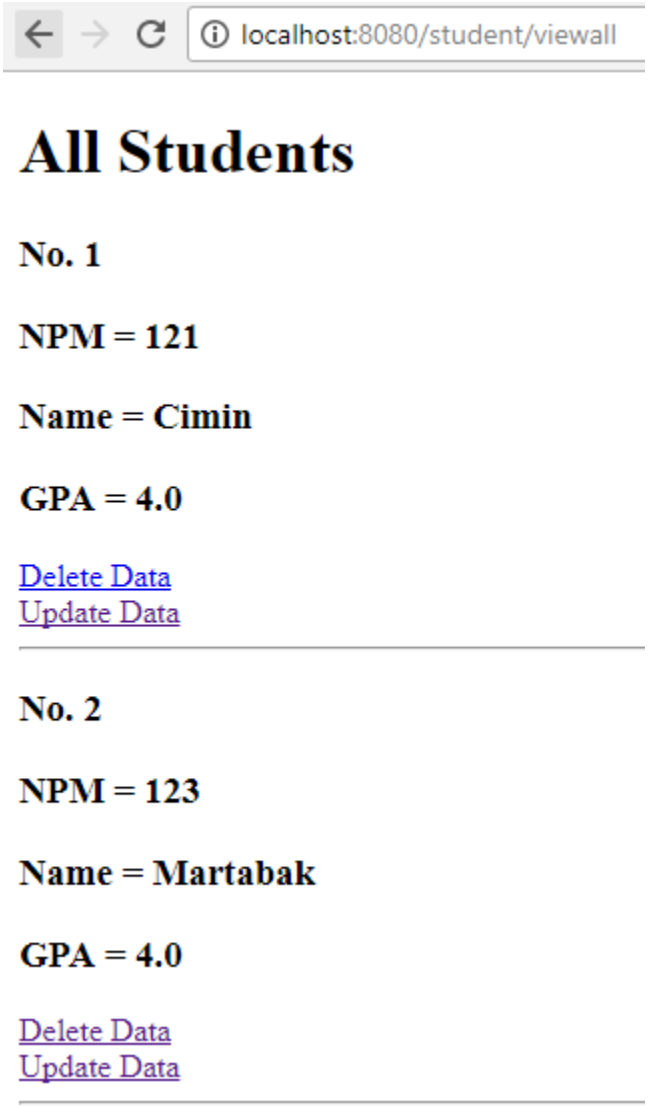
da nth:field=”*{[nama_field]}”, code selengkapnya sebagai berikut :

```
<!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 Data</h1>

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

        <div>
            <button type="submit" name="action" value="update">Update</button>
        </div>
    </form>

</body>

</html>
```

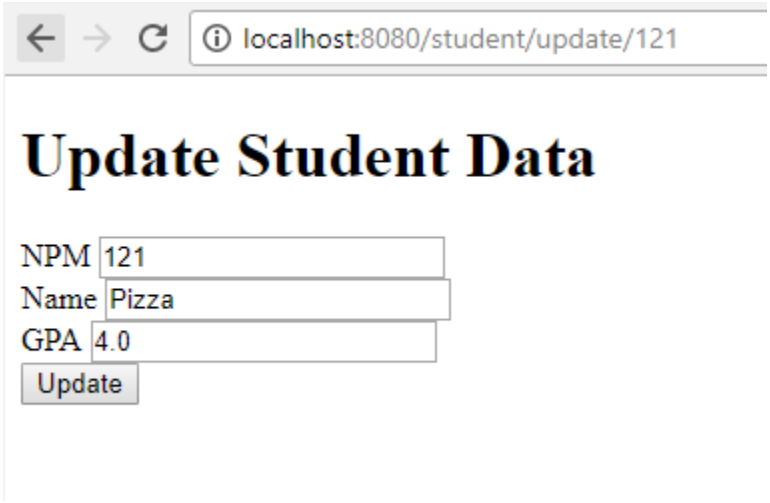
Tampilan saat dijalankan adalah sebagai berikut :

Adzkia Husnul Abidat - 1506757680

- View All



- Update



- Tampilan View All setelah diupdate