Ramadhana Ibnu Akbar
1606954975-EXT

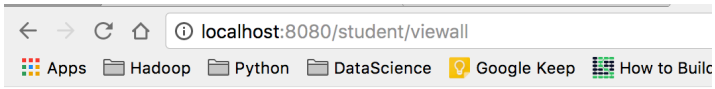
TUTORIAL 4 APAP

1. Method Delete

```
@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";
    }
}
```

Pada method delete yang pertama dilakukan adalah memeriksa apakah NPM tersebut ada di database. Ini dilakukan dengan memanggil class StudentService dan menggunakan method selectStudent. Selanjutnya jika NPM tersebut ada, maka akan memanggil method deleteStudent dari StudentService dengan paramaeter npm. Pada method deleteStudent ini akan memanggil method deleteStudent dari class StudentMapper dimana method ini yang menjadi penghubung dengan database dan semua query juga dituliskan di method ini.



All Students

Delete Data Update Data

No. 1

NPM = 11111

Name = test2

GPA = 4.0

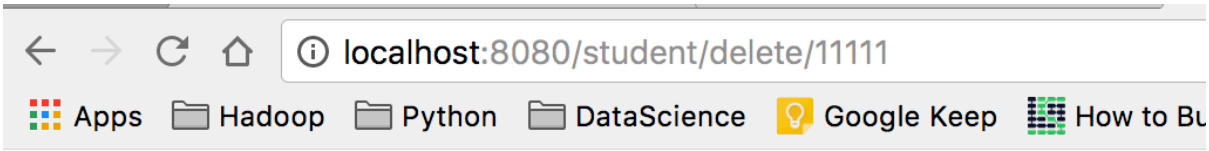
Delete Data Update Data

No. 2

NPM = 12345

Name = chanek

GPA = 3.5



Data berhasil dihapus

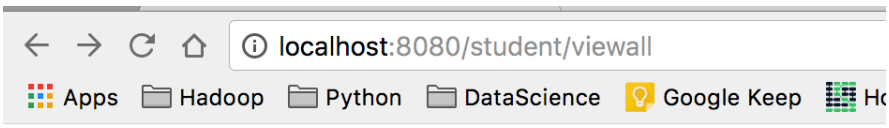
2. Method Update

```
@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";
}
```

Untuk update data terdiri dari dua method, method update untuk menentukan data mana yang akan di update sedangkan untuk method updateSubmit untuk mengganti data lama dengan yang baru. Pada method Update prosesnya sama dengan method Delete, perbedaanya hanya setelah npm diperiksa apakah ada di database atau tidak. Jika npm ada di database maka akan menuju ke view form-update.html yaitu form untuk mengubah data student dengan parameter npm yang akan diubah. Selanjutnya pada form-update akan ditangani oleh method updateSubmit. Pada method updateSubmit untuk update data menggunakan method updateStudent pada class StudentService dengan 3 parameter npm, name, dan gpa. Selanjutnya akan dilanjutkan oleh method updateStudent pada class StudentMapper.



All Students

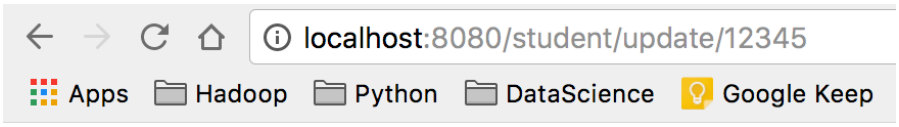
Delete Data Update Data

No. 1

NPM = 12345

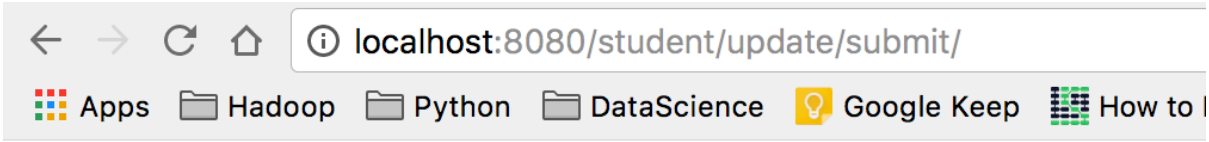
Name = chanek

GPA = 3.5



Problem Editor

NPM 12345
Name rama
GPA 3.5
Save



Data Successfully Updated!

3. Object sebagai Parameter

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

Class StudentController

```
<form action="/student/update/submit/" method="post" 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>
```

form-update.html

Dengan object sebagai parameter, tidak perlu lagi menggunakan @RequestParam untuk setiap kolom-kolom database cukup dengan melempar parameter object student ke view. Sebagai gantinya di view form-update.html pada tags <form></form> ditambahkan th:object="${student}" sebagai model object yang menyimpan data dari StudentController.

Pertayaan

1. Ya diperlukan untuk melihat nilainya sudah sesuai atau tidak atau null atau tidak.
2. Dengan menggunakan POST akan lebih aman, karena data yang dikirim tidak akan muncul di url pada browser. Dan tidak perlu penanganan berbeda pada body.
3. Mungkin, sebernarnya dengan menggunakan anotasi @RequestMapping secara default untuk mapping HTTP request dengan GET ataupun POST. Untuk lebih spesifik bisa menggunakan @GetMapping untuk method GET dan @PostMapping untuk method POST.