Nama : Muhammad Kamal
NPM : 1606954905

Tutorial 04 APAP

1. Method untuk delete

- Method pada studentmapper untuk delete

```
@Delete("Delete from student where npm = #{npm}")
void deleteStudent(@Param("npm") String npm);
```

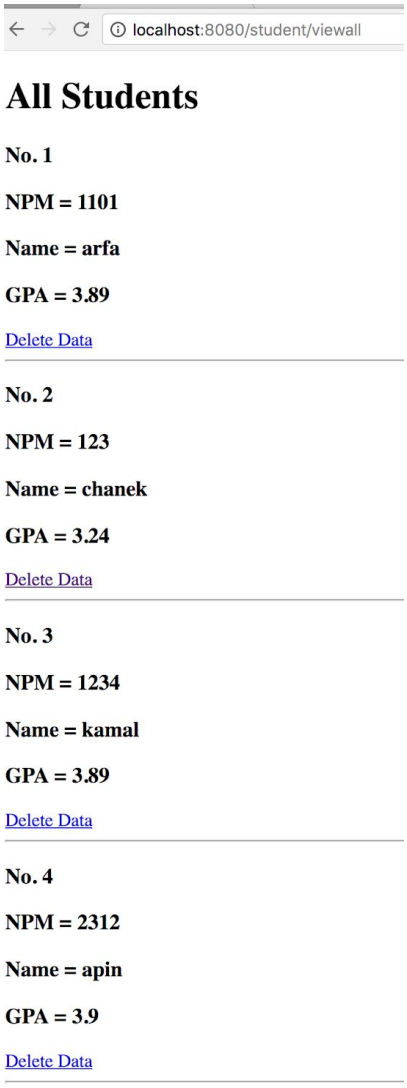
- Method delete student pada service class

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

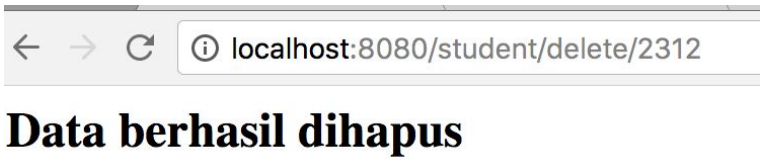
- Pada controller tambahkan pengecekan, berikut

```
@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";
    }

}
```

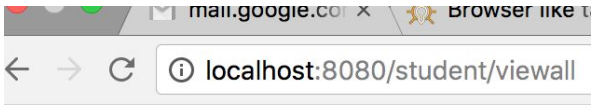
- Hasil dari tambah student, lakukan view all



- Lakukan panggilan delete, dengan klik link di view all, akan tampil sebagai berikut



Selanjutnya lakukan view all setelah delete



All Students

No. 1

NPM = 1101

Name = arfa

GPA = 3.89

Delete Data

No. 2

NPM = 123

Name = chanek

GPA = 3.24

Delete Data

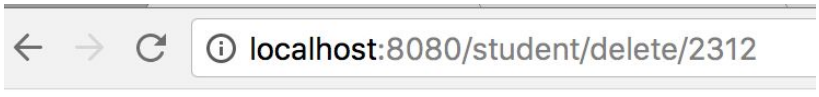
No. 3

NPM = 1234

Name = kamal

GPA = 3.89

Delete Data



Student not found

NPM = 2312

2. Method menambahkan update

- Method update pada studentmapper

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

- Method update student pada interface studentservice

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

- Implementasi method update student pada studentservicedatabase

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

- Tambahkan link pada view all untuk mengarahkan ke form edit untuk diupdate

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

- Tambahkan view form-update untuk melakukan edit/update

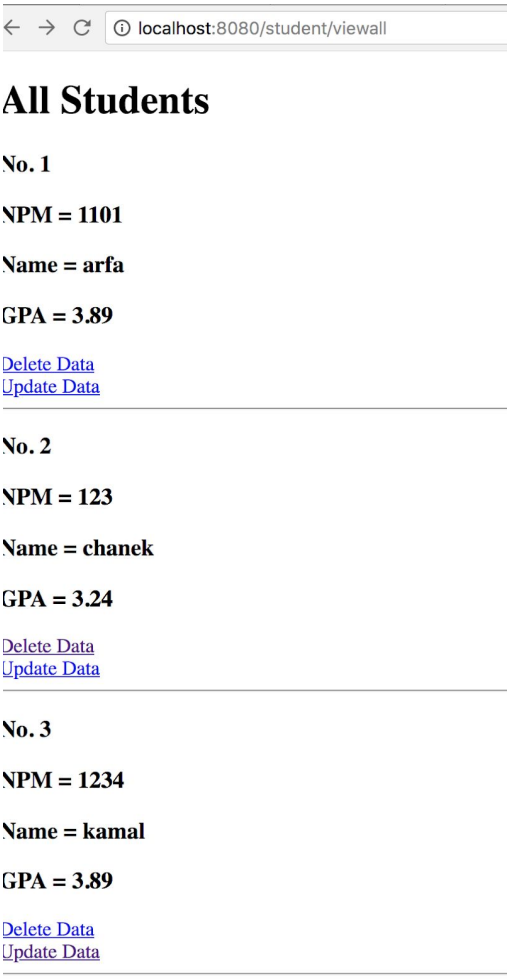
```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:th="http://www.thymeleaf.org">
<head>

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

<body>

        <h1 class="page-header">Problem Editor</h1>

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

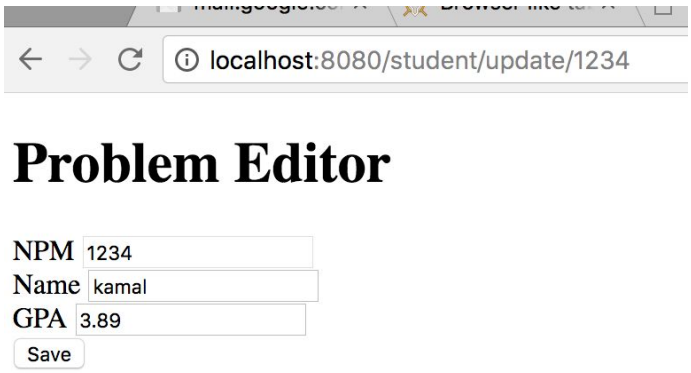
```
                        <label for="gpa">GPA</label> <input type="text" name="gpa"
th:value="${student.gpa}" />
                </div>

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

</body>

</html>
```

- Tambahkan method di studentcontroller untuk akses update student, sebagai berikut

```
@RequestMapping("/student/update/{npm}")
  public String updatePath (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";
     }
  }
```

Method di atas, akan mengarahkan data berdasarkan select student yang ada di db, dan akan ditampilkan di view form-update, jika tidak ditemukan, akan tampil view not-found, berikut tampilan view all dan form update setelah dipilih





- Tambahkan method dengan tipe request post untuk melakukan update data di 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)
  {
                StudentModel student = new StudentModel (npm, name, gpa);
      studentDAO.updateStudent(student);

      return "success-update";
  }
```

Berikut hasil setelah melakukan save update submit

Data berhasil diubah/diupdate

Latihan menggunakan object

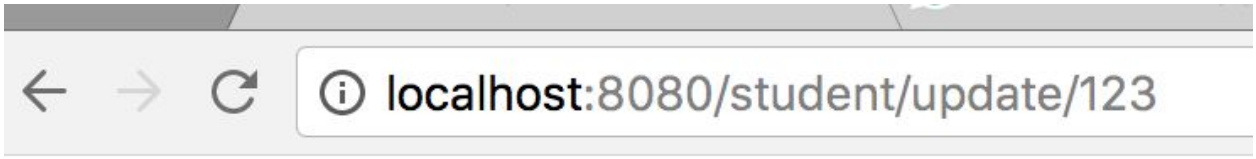
1. Tambahkan 1 html baru yang akan dipanggil ketika view berdasarkan npm, html ini tidak merubah html file “form-update” tetapi membuat baru untuk melihat perbedaan sintaks ketika melakukan post dengan menggunakan model, html ini bernama “form-update-object”, berikut isi htmlnya

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

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

Lalu di studentcontroller lakukan perubahan pada method updateSubmit sebagai berikut

```
@PostMapping(value = "/student/update/submit")
  public String updateSubmit(@ModelAttribute StudentModel student)
  {
                //StudentModel student = new StudentModel (npm, name, gpa);
                if (student != null)
                {
      studentDAO.updateStudent(student);
                }
      return "success-update";
  }
```

Pada method diatas, hanya memerlukan modelattribute yaitu studentmodel, tidak seperti menggunakan requestparam dimana harus memanggil semua atribut yang dikirim dari HTML, berikut tampilan form update dan setelah melakukan update



Problem Editor

NPM 123
Name chanek1
GPA 3.24
Save

Data berhasil diubah/diupdate

No. 1

NPM = 1101

Name = arfa

GPA = 3.89

Delete Data
Update Data

No. 2

NPM = 123

Name = chanek1343

GPA = 3.24

Delete Data
Update Data

No. 3

NPM = 1234

Name = kamal2233

GPA = 3.89

Delete Data
Update Data

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.
   Jawaban:
   Ya, diperlukan validasi di backend, dengan cara mengecek apakah model yang dikirim bernilai null atau tidak, seperti pada contoh updateSubmit method yang telah dirubah di atas

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?

   Jawaban:
   Jika menggunakan POST method, data yang dikirimkan tidak akan terlihat pada brwoser dan bersifat sensitif, selain itu bisa mengirimkan data dengan jumlah yang lebih besar dibandingkan GET method.
   Akan diperlukan penanganan berbeda apabila POST yang dikirim, berbentuk object atau requestmapping.

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   Jawaban:
   Bisa, tambahkan atribute menggunakan {}, misal -> method = { RequestMethod.GET, RequestMethod.POST }