Ervina Puspita Sari | B
1506689414

**LATIHAN MENAMBAHKAN DELETE**

Saat saya membuat fitur delete ini, saya mulai dengan mengedit script SQL @Delete pada kelas interface StudentMapper.java di mana di sini saya mengapus data dari table student ketika npm dalam url sama dengan npm student di database,. Selanjutnya saya membuat method void deleteStudent() yang berisi parameter param("npm") dari url.

Dalam StudentMapper.java :

```
  //DELETE
    @Delete ("DELETE from STUDENT WHERE npm = #{npm}")
    void deleteStudent(@Param("npm") String npm);
```

Kemudian, saya melengkapi method deleteStudent() yang ada di kelas StudentServiceDatabase.java dan menyambung ke kelas StudentService.java . Di sini saya menambahkan logging Slf4j yaitu log.info kemudian memanggil method deleteStudent() yang berisi parameter npm, yang sudah dibuat di StudentMapper.java tadi.

Dalam StudentService.java:

```
    //DELETE
    void deleteStudent (String npm);
```

Dalam StudentServiceDatabase.java:

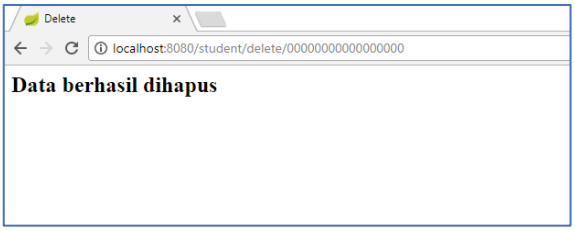
```
  //DELETE
    @Override
    public void deleteStudent (String npm)
    {
       log.info("student " + npm + " deleted");
       studentMapper.deleteStudent(npm);
    }
```

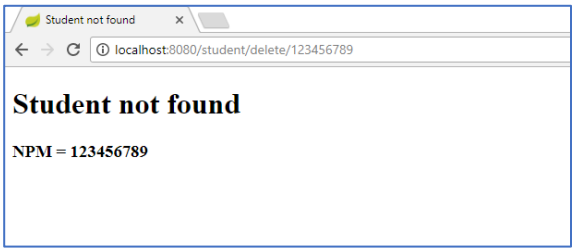
Terakhir, saya mengedit method delete pada kelas StudentController.java . Ketika data student ada, maka tampilan delete.html akan ditampilkan dan data berhasil dihapus. Namun ketika data student tidak ditemukan, maka akan dikembalikan ke not-found.html dan tidak ada data yang dihapus.

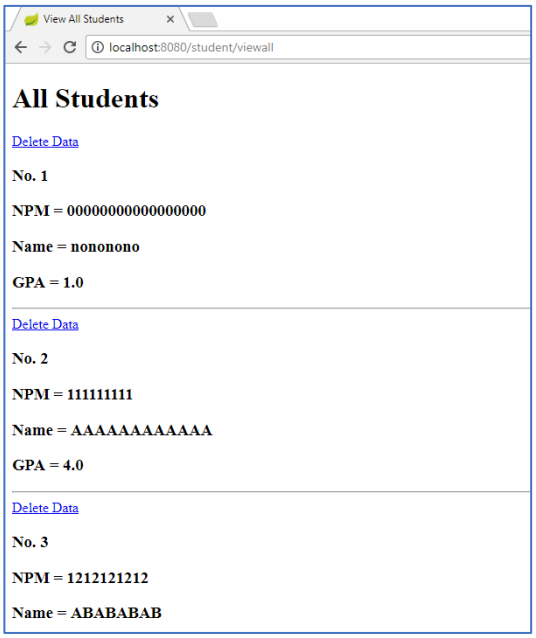
Dalam StudentController.java :

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

```
}
```




**LATIHAN MENAMBAHKAN UPDATE**

Urutan langkahnya kurang lebih sama ketika saya mengimplementasikan delete. Dimulai dengan mengedit updateStudent() pada StudentMapper.java

Dalam StudentMapper.java :

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

Selanjutnya membuat dan menambahkan updateStudent() pada StudentService.java dan StudentServiceDatabse.java yang memuat logging Slf4j

Dalam StudentService.java:

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

Dalam StudentServiceDatabase.java:

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

Ervina Puspita Sari | B
1506689414

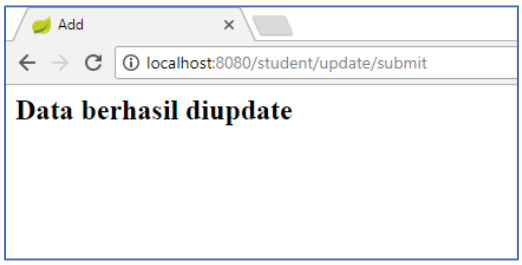
Selain itu saya juga membuat halaman html baru di antaranya form-update.html dan success-update.html yang mengadobsi dari form-add.html dan success-add.html. Di sini syntax input type dirubah, di mana npm dijadikan readonly=true karena merupakan key sehingga tidak bisa dirubah lagi isinya.

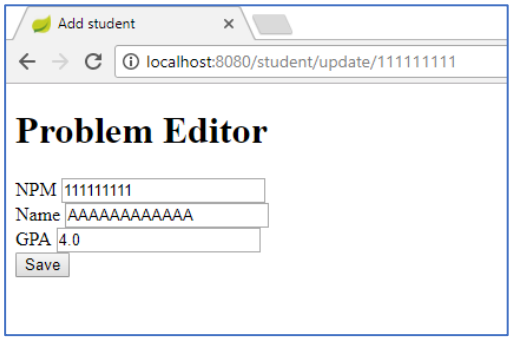
Terakhir, saya membuat controller. Di sini ada dua controller, yaitu controller yang mengurus form berisi data yang akan diupdate dan controller yang mengurus submit dari form tersebut.

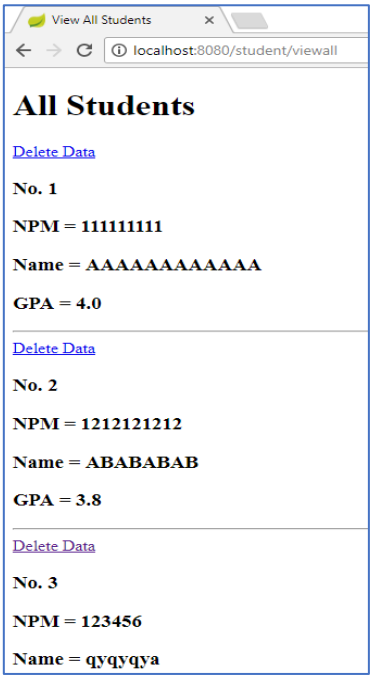
```
//UPDATE Untuk mengurus form
  @RequestMapping("/student/update/{npm}")
  public String update (Model model, @PathVariable(value = "npm") String npm) {
      StudentModel student = studentDAO.selectStudent (npm);
       if(student!= null) {
           model.addAttribute("student", student);
    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


   //UPDATESUBMIT, mengurus submit form
  @RequestMapping("/student/update/submit")
  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";
  }
```



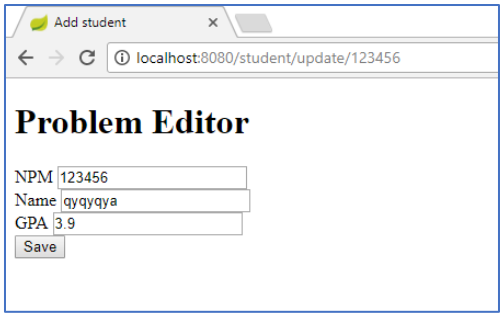



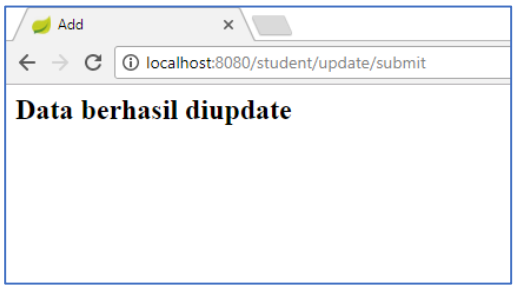
Ervina Puspita Sari | B
1506689414

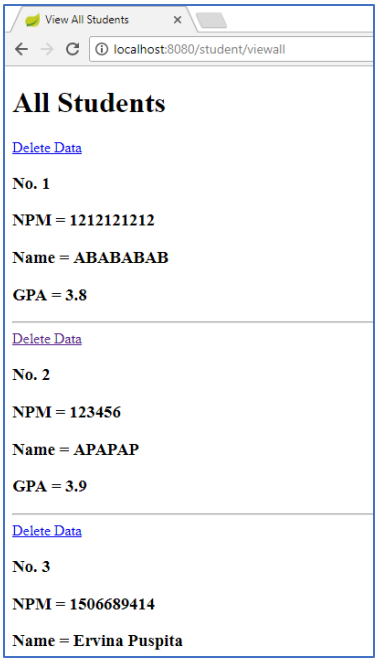
**LATIHAN MENG GUNAKAN OBJECT SEBAGAI PARAMETER**

Pada latihan ini, kembali membuat update namun dengan menggunakan StudentModel dan meniadakan RequestParam. Menggunakan RequestParam sesuai dengan jumlah variable sangatlah tidak efisien, terlebih jika jumlah variablenya banyak. Maka @ModelAttribute ini sangat disarankan karena lebih simple hanya dengan menambahkan suatu object yang terdiri dari berbagai variable. Saya membuatnya seperti ini, karena ini merupakan controller untuk mensubmit form, saya menggunakan @PostMapping.

```
//UPDATESUBMIT_STUDENTMODEL
 @PostMapping("/student/update/submit")
 public String updateSubmitStudentModel (@ModelAttribute StudentModel student)  {
     studentDAO.updateStudent(student);
     return "success-update";
 }
```



Ervina Puspita Sari | B
1506689414

PERTANYAAN

1. Saat saya menggunakan object sebagai parameter pada form post dan tidak terdapat attribute required pada input type nya, saya melakukan validasi input yang optional dan required dengan meletakannya pad akelas StudentModel.java di mana di atas private variable saya berikan @NotNull yang artinya variable tersebut tidak boleh kosong. Karena object Student terdiri dari variable npm, name, dan gpa, maka jika salah satunya tidak diisikan maka object tidak bisa dipost/ditambahkan. Validasi diperlukan.

```
1  package com.example.model;
2
3  import javax.validation.constraints.NotNull;
8
9  @Data
10 @AllArgsConstructor
11 @NoArgsConstructor
12 public class StudentModel
13 {
14
15     @NotNull
16     private String npm;
17
18     @NotNull
19     private String name;
20
21     @NotNull
22     private double gpa;
23
24 }
```

2. Karena saat submit form, kita harus mengirim data yang ada ke database, maka digunakanlah post. Sementara get digunakan untuk menampilkan data yang ada dari database. Perlu penanganan berbeda, yaitu pada parameternya, @PostMapping menggunakan **public String updateSubmitStudentModel (@ModelAttribute StudentModel student)** , dan @GetMapping **menggunakan public String update (Model model, @PathVariable(value = "npm") String npm). N**amun @PostMapping dan @GetMapping juga hampir memiliki syntax yang mirip.

3. Mungkin, seperti pada form. Kita get data kemudian post data dalam method delete sekaligus update.