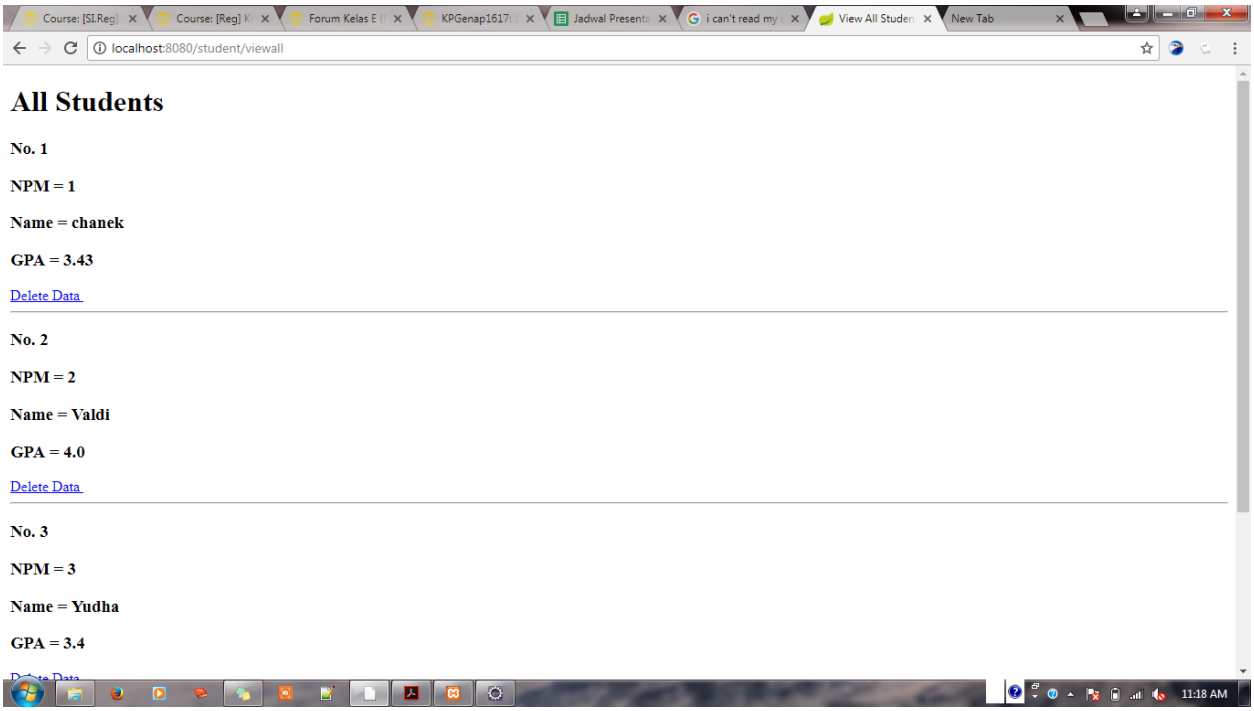
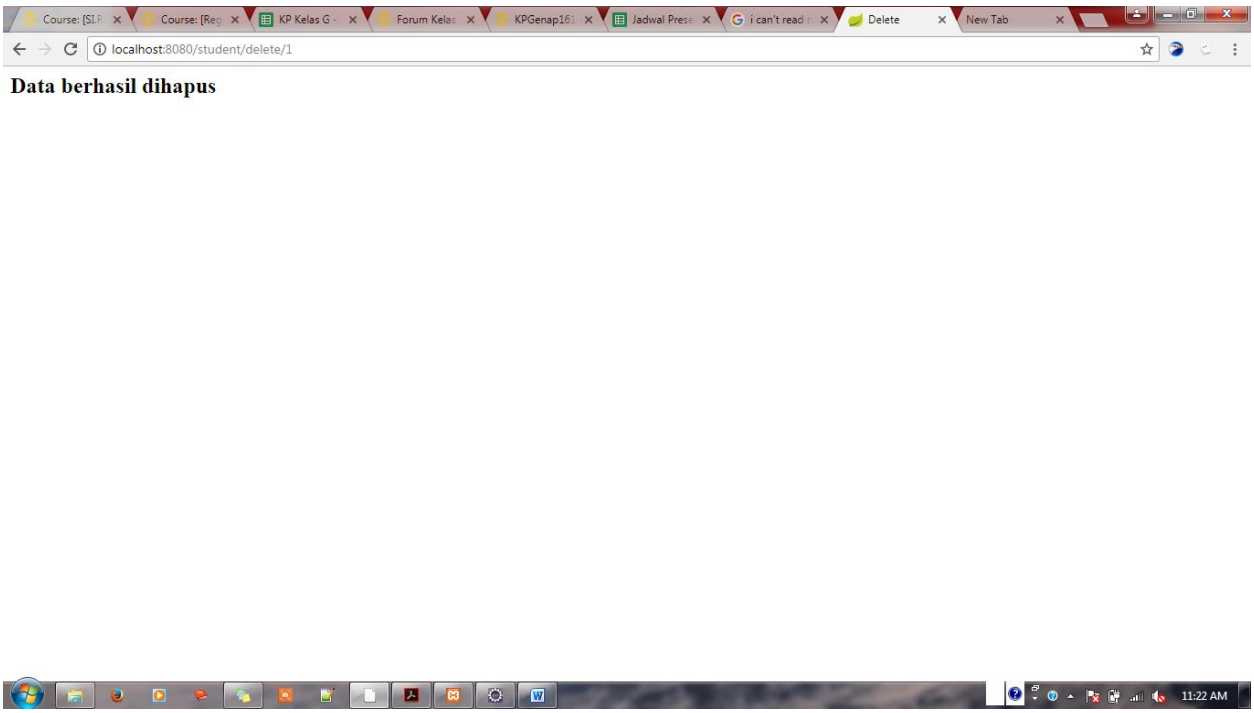
Iman Alfathan Yudhanto
1406623524
APAP-A
Write-up dan Penjelasan Tutorial 4

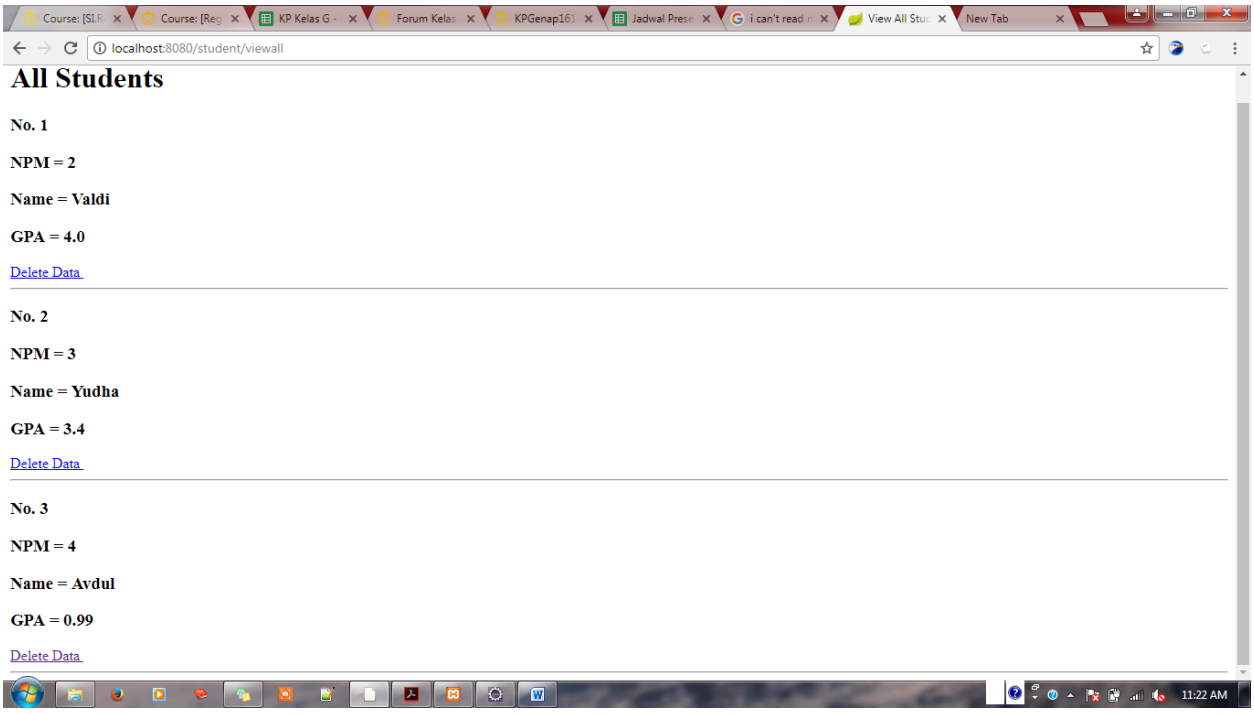
Latihan Menambahkan Delete



Sebelum melakukan delete



Saat melakukan delete



Setelah melakukan delete

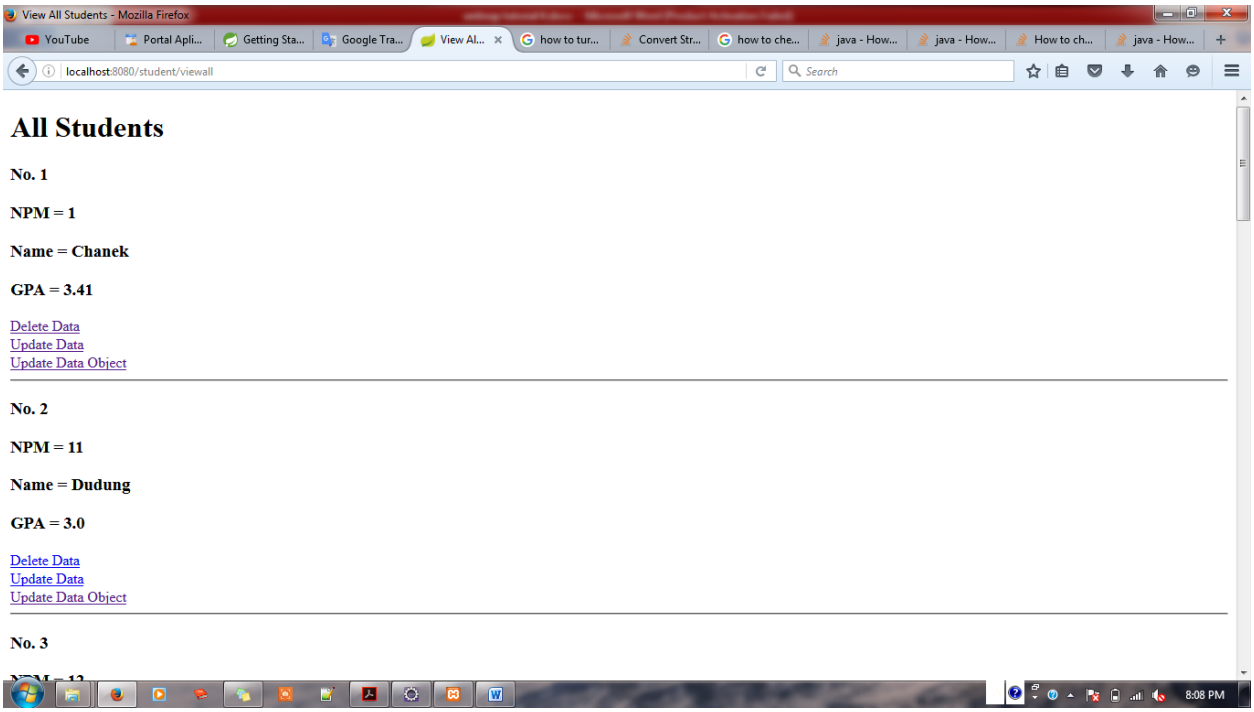
Penjelasan: Pada bagian ini, dilakukan alur logic pada bagian controller. Yang pertama adalah memanggil student yang ingin dihapus berdasarkan npm. Setelah itu melakukan validasi. Jika

npm student ada, maka dilakukan penghapusan dan me-return ke halaman berhasil dihapus. Ika tidak ditemukan, maka akan me-return ke halaman not-found. Kode pada controller seperti berikut:

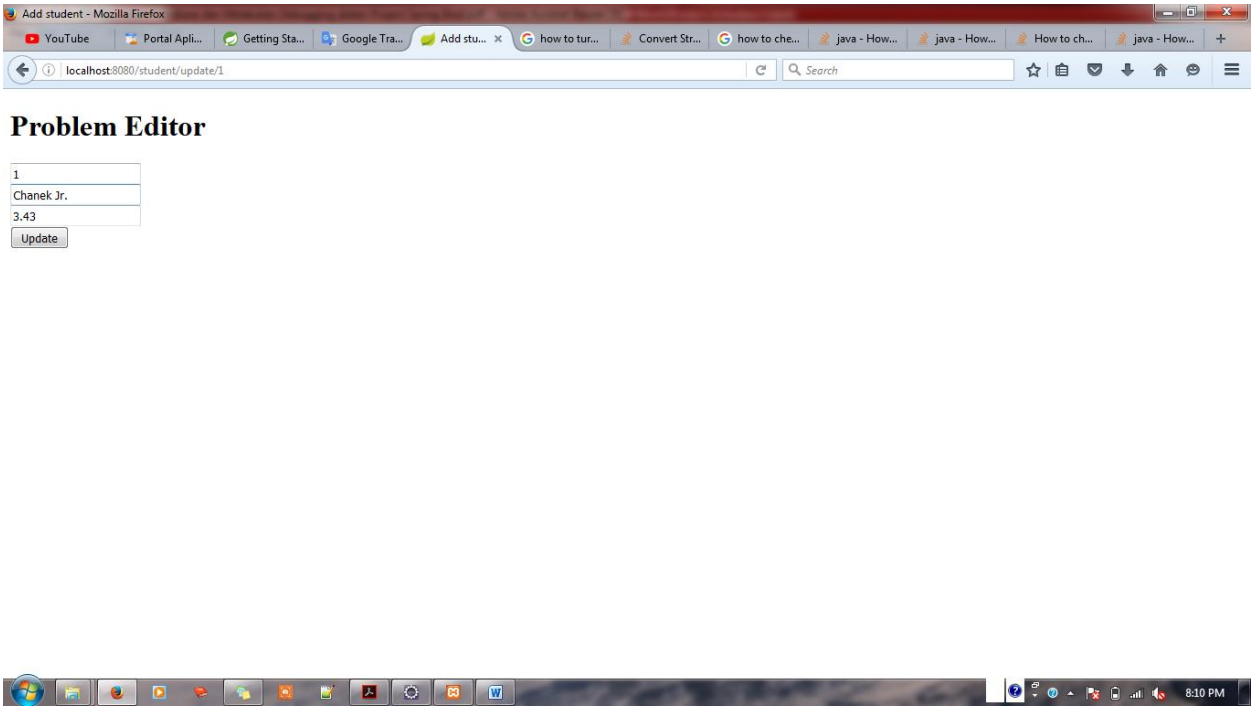
```
@RequestMapping("/student/delete/{npm}")
	public String delete(Model model, @PathVariable(value = "npm")
String npm) {
		StudentModel student = studentService.selectStudent(npm);

		if (student != null) {
			studentService.deleteStudent(npm);
			return "delete";
		} else {
			model.addAttribute("npm", npm);
			return "not-found";
		}
	}
```

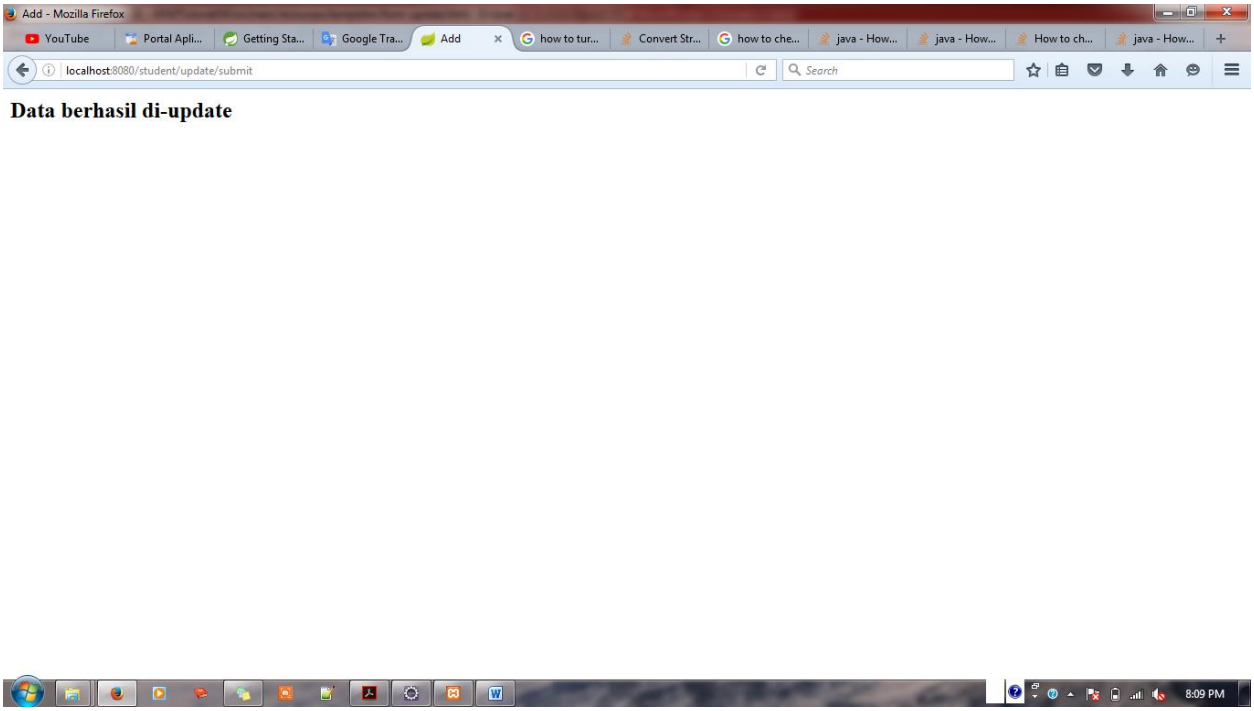
Latihan Menambahkan Update



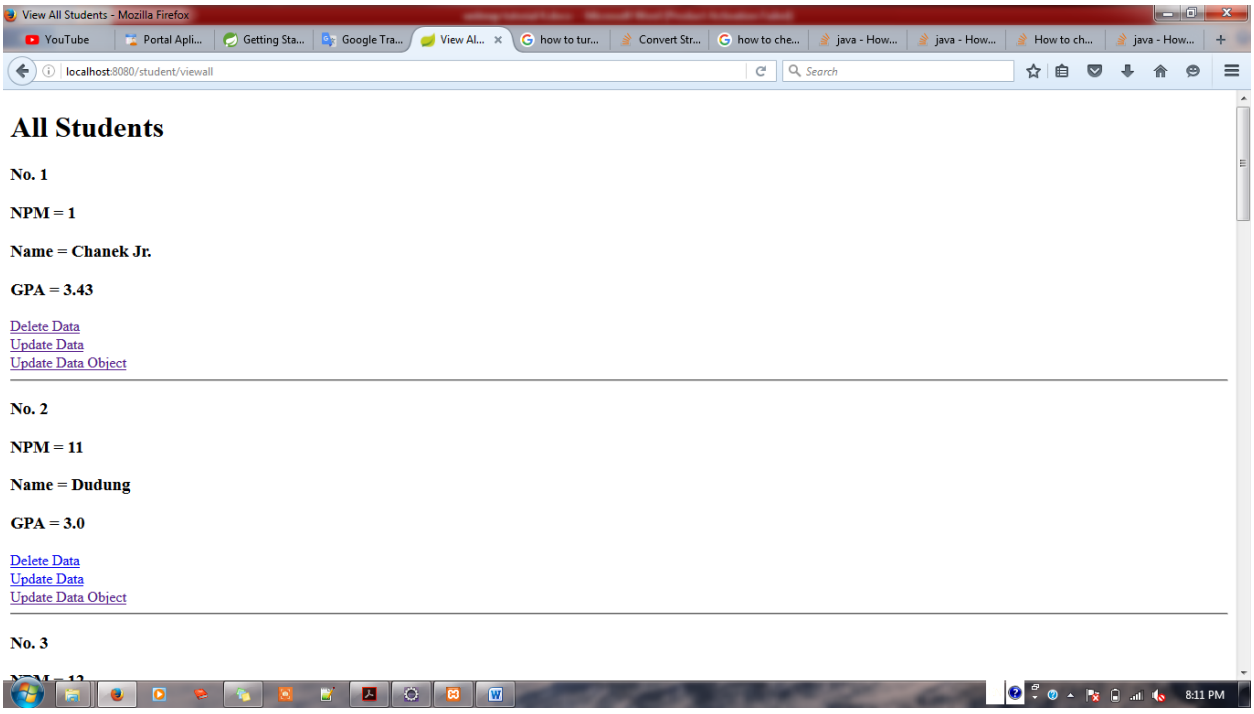
Sebelum melakukan update



Saat melakukan update. Melakukan perubahan pada name dan gpa.



Setelah melakukan update



Hasil dari melakukan update. Terjadi perubahan pada nama dan gpa pada npm 1.

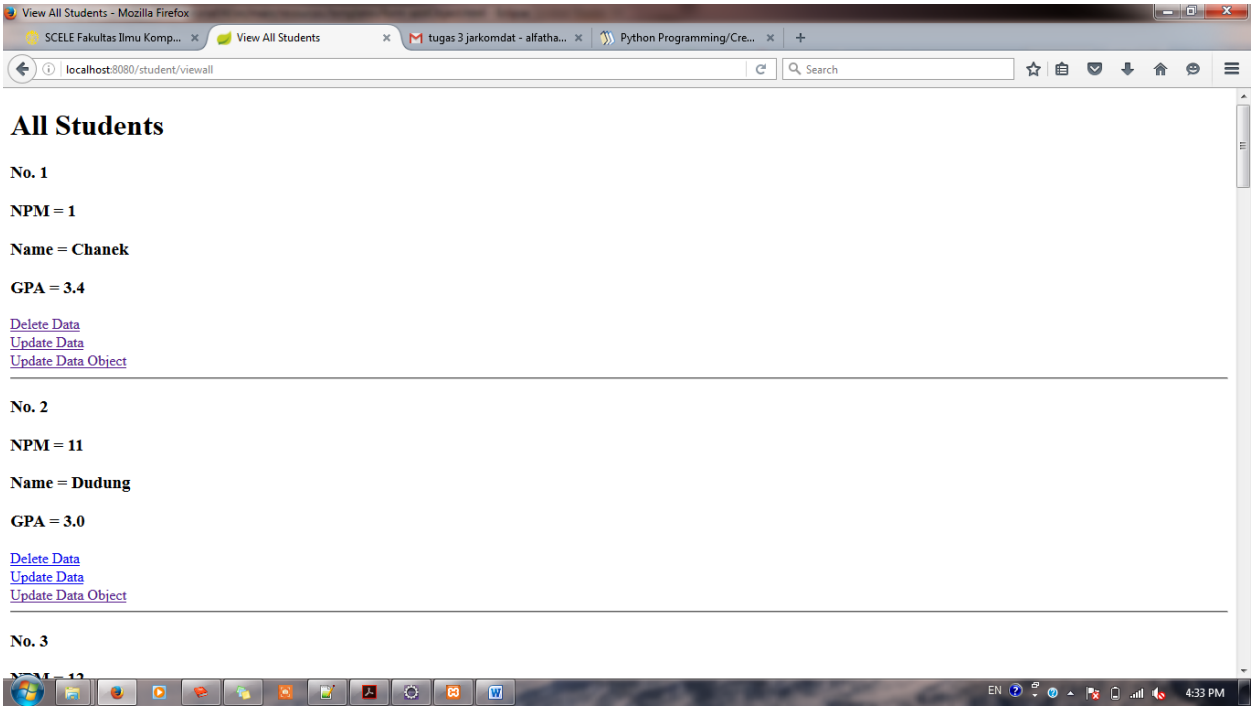
Penjelasan: Pada bagian ini, dilakukan alur logic pada bagian controller. Yang pertama adalah memanggil student yang ingin dihapus berdasarkan npm. Setelah itu melakukan validasi. Jika

npm student ada, maka akan melakukan add attribute agar bisa membawa data yang lama ke dalam front end, lalu me-return ke halaman form-update. Ika tidak ditemukan, maka akan me-return ke halaman not-found. Pada controller juga ditambahkan kode untuk melakukan submit. Yang pertama dilakukan adalah membuat objek student baru berdasarkan parameter, lalu memanggil method pada service untuk meng-update. Lalu me-return ke halaman berhasil meng-update. Selain itu, ada validasi pada method service jika gpa yang dimasukkan bernilai kosong. Jika bernilai kosong, maka halaman akan kembali ke halaman form-update. Jika tidak, maka akan melakukan update lalu me-return ke halaman success-update. Kode pada controller seperti berikut:

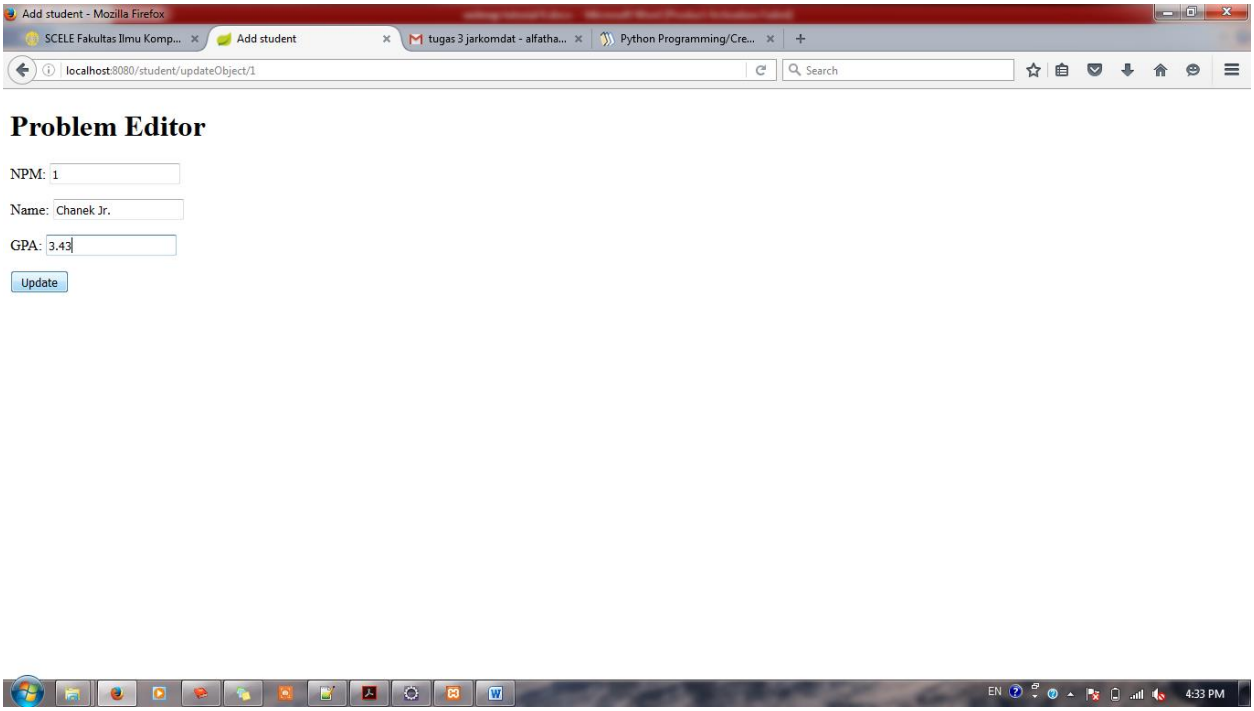
```
	@RequestMapping("/student/update/{npm}")
	public String update(Model model, @PathVariable(value = "npm") String npm) {
		StudentModel student = studentService.selectStudent(npm);
		if (student != null) {
			model.addAttribute("student", student);
			return "form-update";
		} else {
			model.addAttribute("npm",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

	@RequestMapping("/student/update/submit")
	public String updateSubmit(Model model, @RequestParam(value = "npm", required
= false) String npm,
			@RequestParam(value = "name", required = false) String name,
			@RequestParam(value = "gpa", required = false) String gpa)
	{
		StudentModel studentTemp = studentService.selectStudent(npm);
		if(gpa.equalsIgnoreCase("")){
			model.addAttribute("student", studentTemp);
			return "form-update";
		}else{
		StudentModel student = new StudentModel(npm, name,
Double.parseDouble(gpa));
		studentService.updateStudent(student);
		return "success-update";
		}
	}
```

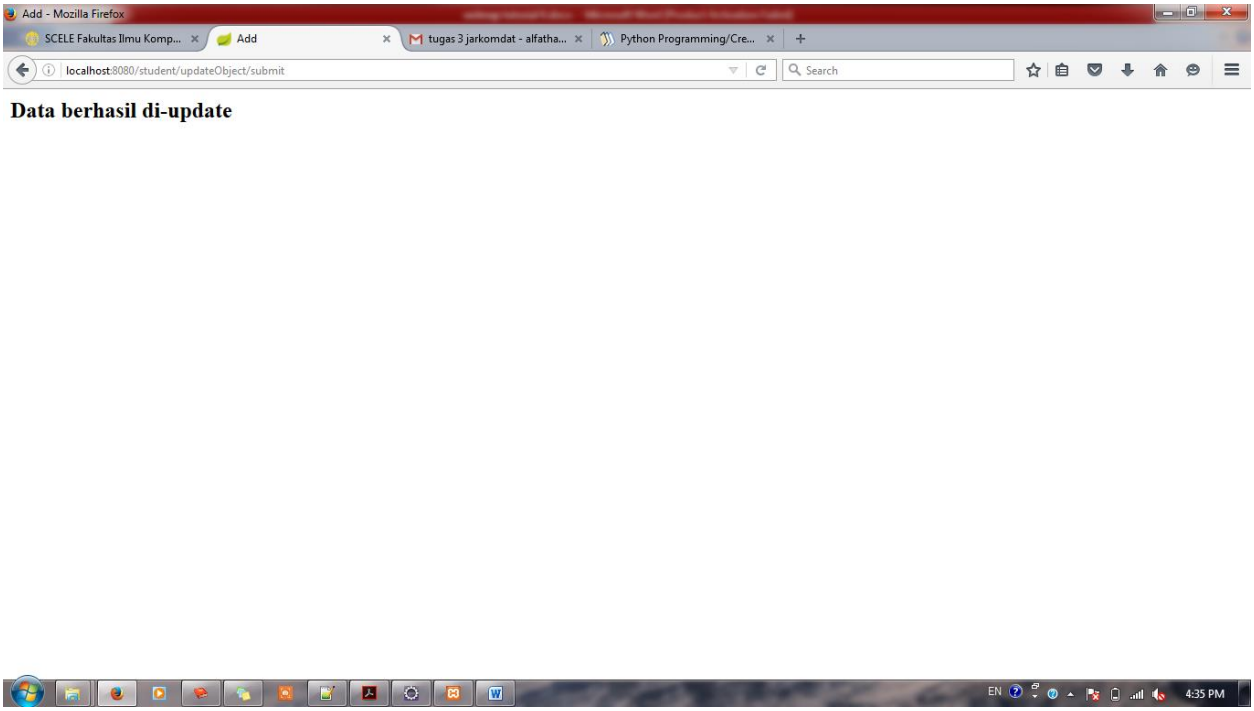
Latihan Menggunakan Object Sebagai Parameter



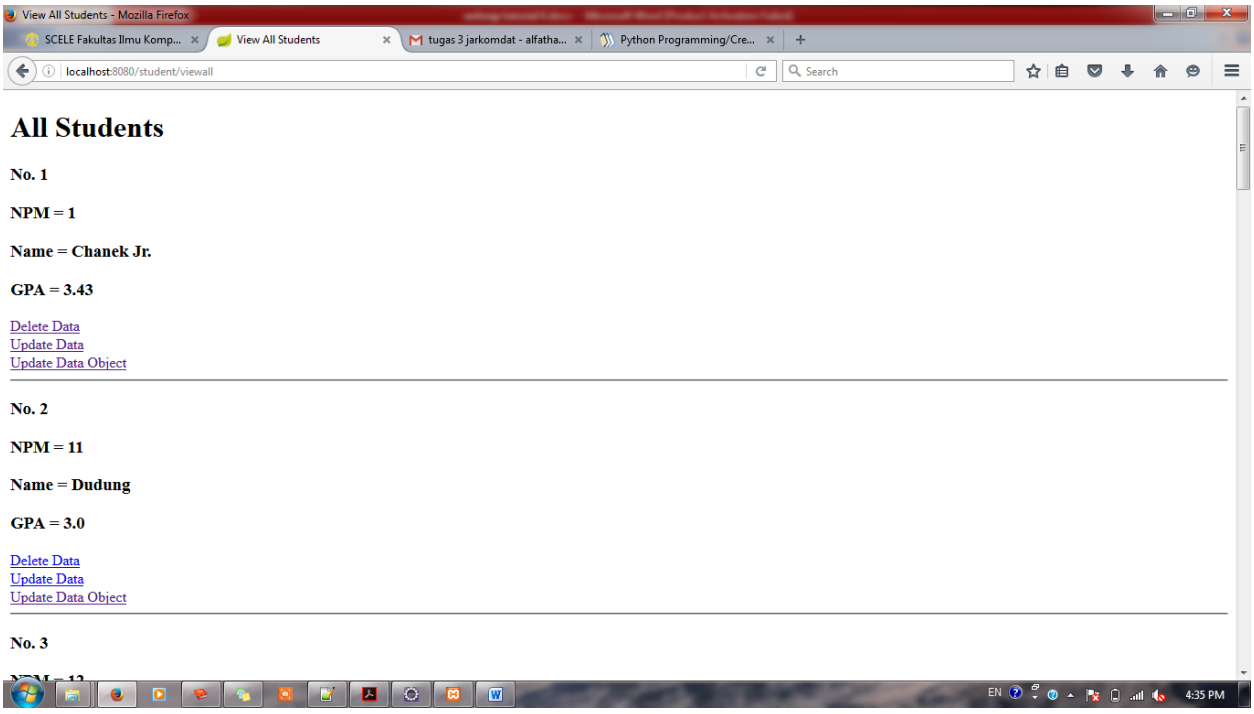
Sebelum melakukan update menggunakan objek.



Ketika melakukan update menggunakan objek.



Berhasil menggunakan update denga objek.



Hasil dari melakukan update. Terjadi perubahan pada nama dan gpa pada npm 1.

Penjelasan: Pada bagian ini, dilakukan alur logic pada bagian controller. Yang pertama adalah memanggil student yang ingin dihapus berdasarkan npm. Setelah itu melakukan validasi. Jika

npm student ada, maka akan melakukan add attribute agar bisa membawa data yang lama ke dalam front end, lalu me-return ke halaman form-update. Jika tidak ditemukan, maka akan me-return ke halaman not-found. Pada controller juga ditambahkan kode untuk melakukan submit. Yang membedakannya adalah parameternya yang berupa objek. Untuk itu, maka dilakukan perubahan pada bagian front-end dimana bagian tersebut langsung meresponnya sebagai variable. Yang pertama dilakukan adalah membuat objek student baru berdasarkan parameter, lalu memanggil method pada service untuk meng-update. Lalu me-return ke halaman berhasil meng-update. Selain itu, ada validasi pada method service jika gpa yang dimasukkan bernilai kosong. Jika bernilai kosong, maka halaman akan kembali ke halaman form-update. Jika tidak, maka akan melakukan update lalu me-return ke halaman success-update. Kode pada controller seperti berikut:

```
    @RequestMapping("/student/updateObject/{npm}")
    public String updateObject(Model model, @PathVariable(value = "npm") String npm)
{
        //model.addAttribute("greeting", new Greeting());
      StudentModel student = studentService.selectStudent(npm);
            if (student != null) {
                  /* System.out.println(student.getNpm());
                  System.out.println(student.getName());
                  System.out.println(student.getGpa()); */
                  model.addAttribute("student", student);
                  return "form-updateObject";
            } else {
                  model.addAttribute("npm", npm);
                  return "not-found";
            }
    }
     @RequestMapping(value = "/student/updateObject/submit", method =
RequestMethod.POST)
      public String updateObjectSubmit(Model model, @ModelAttribute StudentModel
student) {
      StudentModel studentTemp = studentService.selectStudent(student.getNpm());

      String gpa= Double.toString(studentTemp.getGpa());
      if(gpa.equalsIgnoreCase("")){
                  model.addAttribute("student", studentTemp);
                  return "form-update";
            }else{
            studentService.updateStudent(student);
            return "success-update";

            }
      }Berikut program pada bagian front-end:
<form action="#" th:action="@{/student/updateObject/submit}" th:object="${student}"
method="post">
            <div>
                  <p>NPM: <input type="text" readonly = "true" th:field="*{npm}"
/></p>
            </div>
```

```
        <div>
              <p>Name: <input type = "text"  th:field="*{name}" /></p>
        </div>
        <div>
              <p>GPA: <input type = "text"  th:field="*{gpa}" /></p>
        </div>

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

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? Apakah validasi diperlukan?
Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.
Jika tidak menggunakan objek sebagai parameter, maka validasi diperlukan sebelum memanggil method pada service karena parameter disebutkan secara langsung. Jika menggunakan object, maka validasi dapat dilakukan pada bagian controller,service, atau pada bagian front-end.

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?
Karena dengan menggunakan post method, maka method POST akan mengirimkan data atau nilai langsung ke action untuk ditampung. Berdasarkan yang saya coba pada program saya, belum perlu penanganan berbeda di *header* atau *body method*.

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
Satu method dapat menerima get dan post sekaligus.

Hal yang dipelajari: Yang saya pelajari dari tutorial ini adalah bagaimana cara mengakses database dan mengelola database dengan program yang telah dibuat. Mapper berfungsi dalam mengambil database. Lalu service berguna dalam mengelola data yang telah diambil oleh mapper. Controller berfungsi dalam mengarahkan ke url, selain itu controller membutuhkan service. Selain itu, saya mendapatkan hal menarik seperti objek sebagai parameter, cara tersebut lebih ringkas karena hanya butuh satu parameter. Front-end berperan penting dalam kasus ini. Pada bagian form, diberi th:field yang menjadikan mereka variable yang merujuk langsung ke objek.