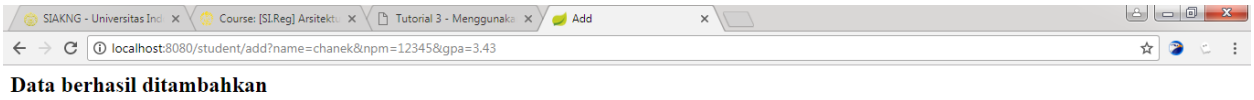
Iman Alfathan Yudhanto
1406623524
ADPAP-A
Latihan dan Write-up Tutorial

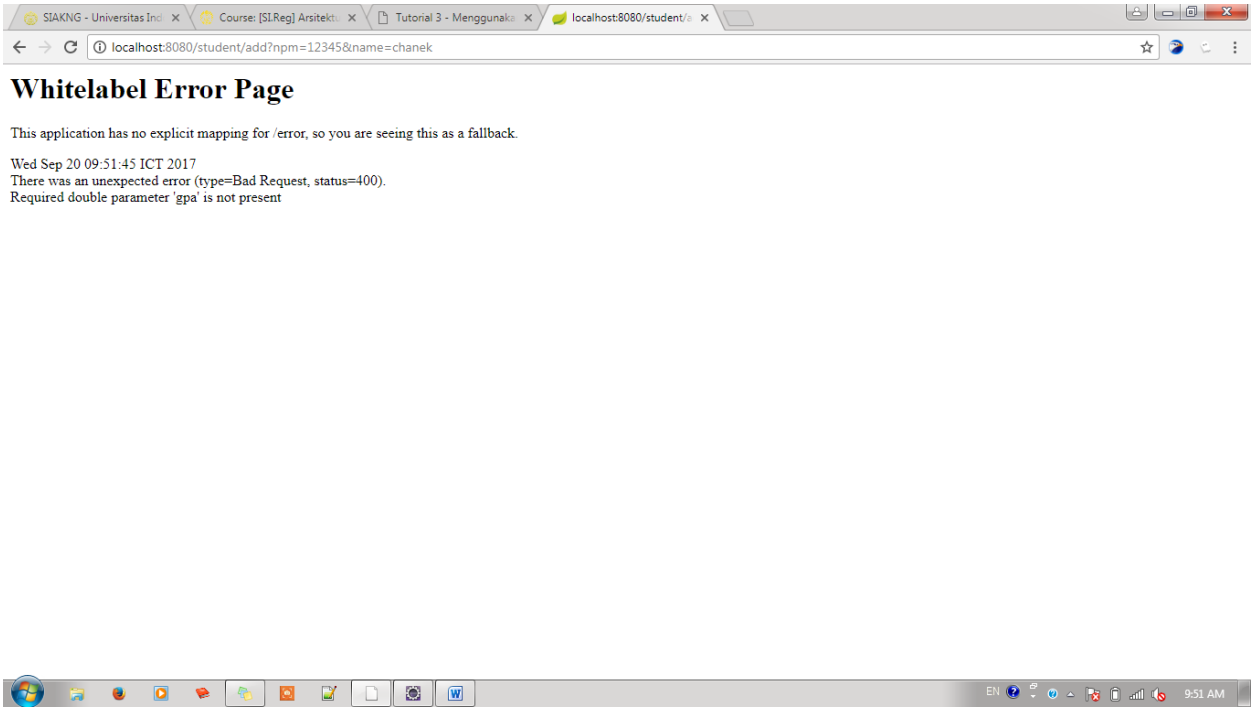
localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43



Pertanyaan 1: apakah hasilnya? Jika error, tuliskan penjelasan Anda.

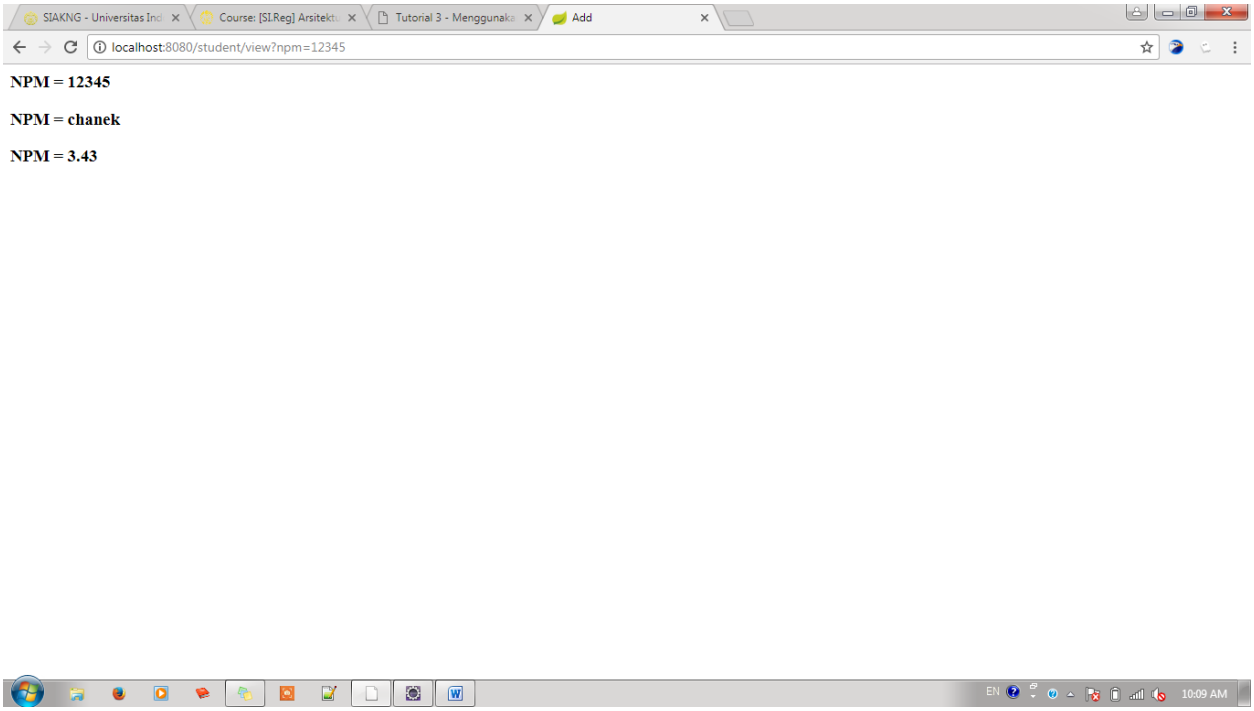
Tidak terjadi error. Hasilnya menampilkan add html yang dipanggil oleh *controller.*

localhost:8080/student/add?npm=12345&name=chanek



Pertanyaan 2: apakah hasilnya? Jika error, tuliskan penjelasan Anda.
Terjadi error karena variable gpa tidak ada. Penyebabnya adalah karena variable gpa kosong. Sedangkan pada controller terdapat required=true yang menandakan bahwa variable harus dimasukkan.

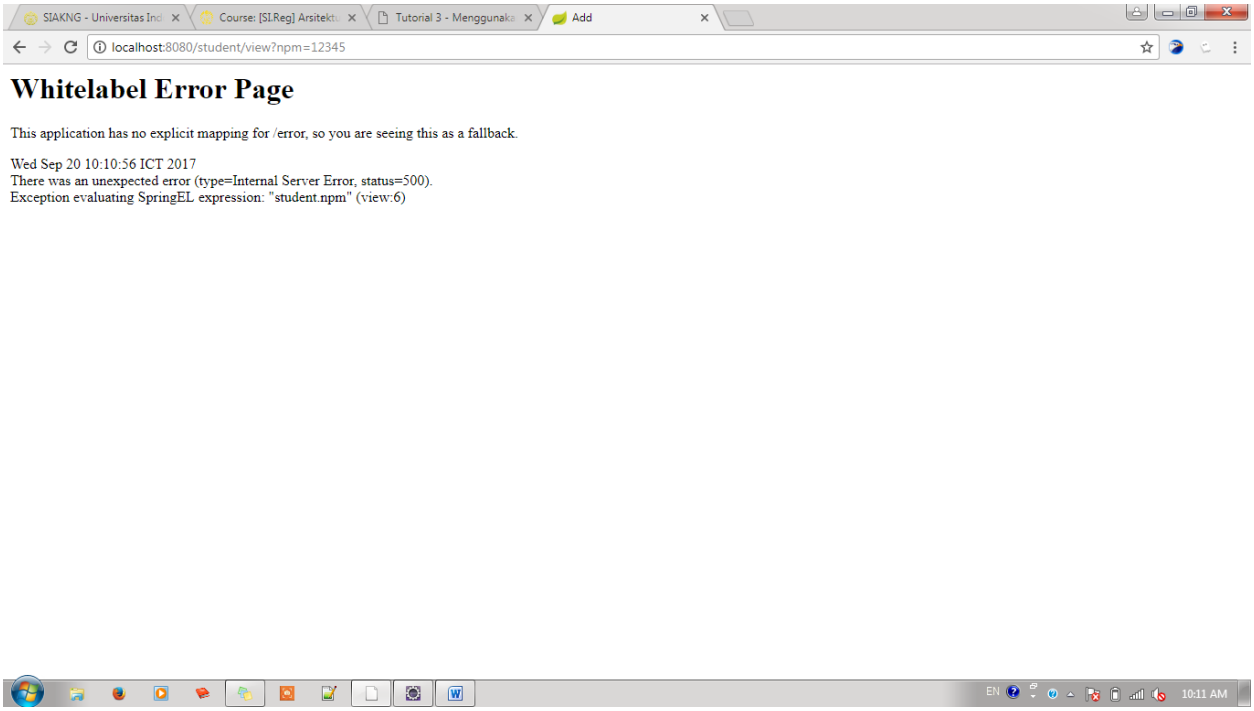
localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43, lalu buka localhost:8080/student/view?npm=12345



Pertanyaan 3: apakah data Student tersebut muncul? Jika tidak, mengapa?

Student tersebut dapat muncul karena npm yang tersimpan di memori ada.

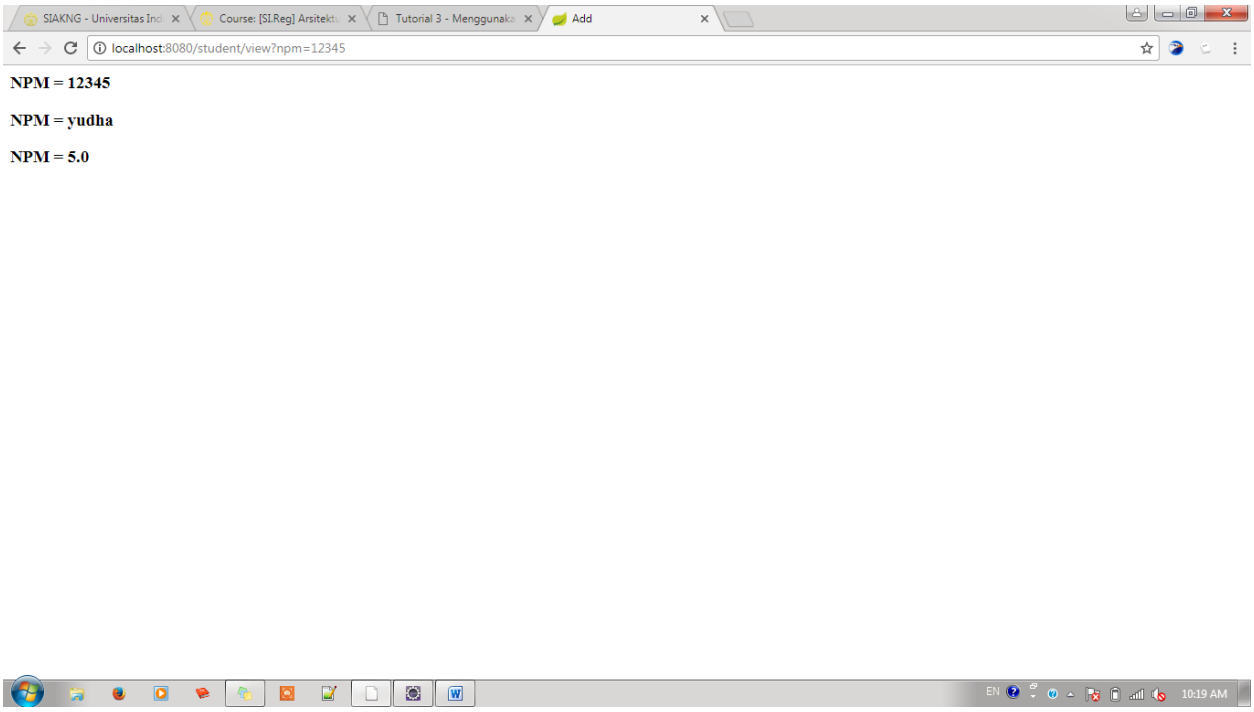
Coba matikan program dan jalankan kembali serta buka localhost:8080/student/view?npm=12345



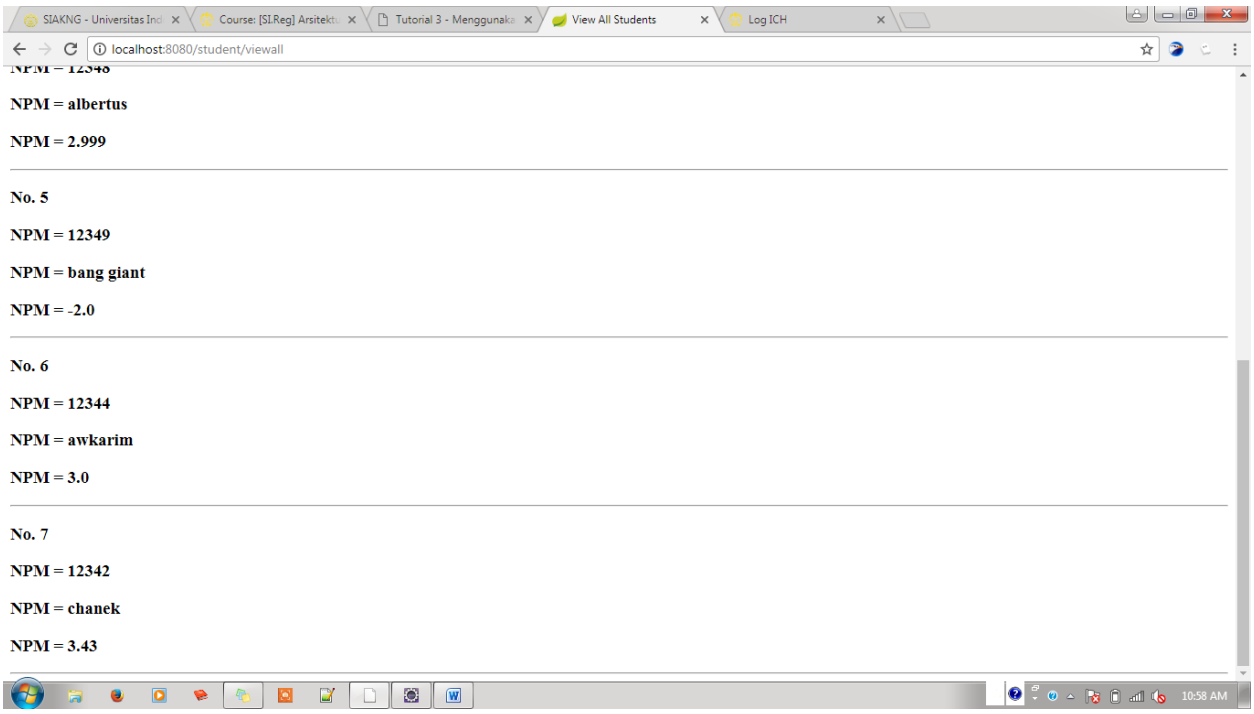
Pertanyaan 4: apakah data Student tersebut muncul? Jika tidak, mengapa?

Data student tidak muncul karena npm tersebut tidak ada di memori. Penyebabnya ketika program dimatikan, secara otomatis data yang tersimpan di memori akan hilang. Data hanya tersimpan jika program dijalankan. Ketika dijalankan lagi, data yang tersimpan di memori akan menjadi kosong.

Coba tambahkan data Student lainnya dengan NPM yang berbeda.



localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/viewall,

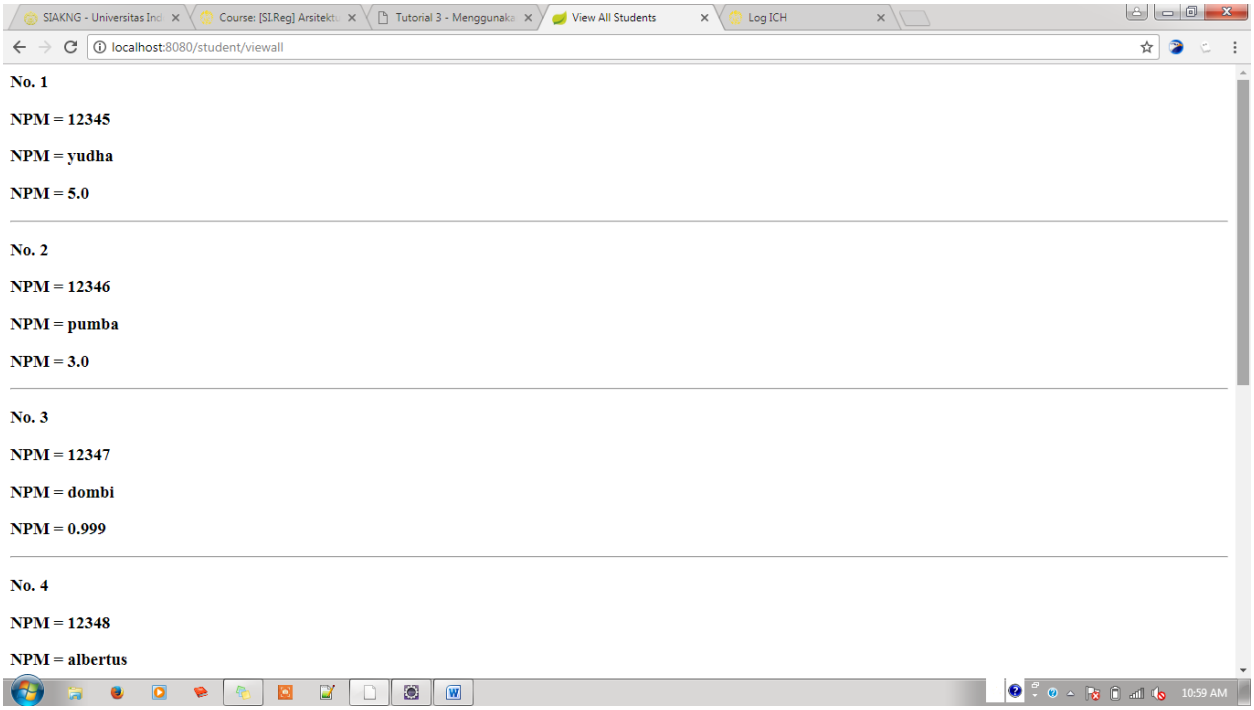


Pertanyaan 5: apakah data Student tersebut muncul?

Data student tersebut muncul karena data tersebut ada dan berhasil ter-*insert.*

*sebelumya sudah dimasukkan beberapa data sebelum melakukan penginputan di atas.

Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka localhost:8080/student/viewall,



Pertanyaan 6: Apakah semua data Student muncul?

Data student tersebut muncul karena data tersebut ada dan berhasil ter-*insert.*

Write-up Tutorial

Pada tutorial ini diajarkan mengendai dasar-dasar dari MVC tanpa menggunakan database. Back-end yang terlibat dalam tutorial ini adalah controller, service, dan model. Pada tutorial ini diajarkan bagaimana cara menambahkan data, melihat detail data, melihat semua data, dan menghapus data. Controller dan service berperan penting dalam tutorial ini. Controller berfungsi sebagai pengarah menuju url. Controller memanggil service agar dapat melakukan manipulasi data dan merepresentasikan jadi sebuah model.

Pada latihan, dibuatlah sebuah method view yang dapat menampilkan data jika ada dan men-*direct* ke halaman error jika data tidak ada.

Method pada controller:

```
@RequestMapping("/student/view/{npm}")
public String view2(@PathVariable String npm, Model model) {
	Student student = studentService.selectStudent(npm);
	model.addAttribute("student", student);
	System.out.println(student.getName() + " " + student.getNpm());
	if (student == null) {
		return "error";
	} else {

		return "view";
	}
}
```

Method pada service:

```
@Override
public Student selectStudent(String npm) {
	// Implement
	Student a = null;
	int size = studentList.size();
	for (int i = 0; i < size; i++) {
		if (studentList.get(i).getNpm().equals(npm)) {
			a = studentList.get(i);
			break;
		}
	}
	return a;

}
```

Pada studentcontroller, dibuat method seperti di atas. Cara kerja method adalah mendapatkan objek student dengan memanggil method select student berdasarkan npm pada service. Setelah itu, objek tersebut dimasukkan ke add attribute. Secara otomatis, mau null ataupun tidak method tersebut tetap tetap menyimpan model. Setelah itu, dilakukan validasi dari objek student. Validasi pertama mengecek apakah objek student null atau tidak. Jika null, maka akan ter-*direct* ke halaman error. Jika tidak null, maka akan ter-*direct* ke halaman view, yang mana akan menampilkan objek student berdasarkan npm yang dicari.

Pada inmemorystudentservice, dibuat method seperti di atas. Cara kerja method adalah membuat objek student yang kosong. Setelah itu, melakukan looping sebuah list yang berisikan objek

student. Jika objek student sesuai dengan apa yang ingin dicari, maka ia akan objek yang null berisi objek yang dicari. Setelah ketemu objek yang dicari, maka looping dihentikan. Dan method me-*return* objek student.

Method pada delete:

```
@RequestMapping("/student/delete/{npm}")
	public String delete(@PathVariable String npm, Model model) {
		Student student = studentService.selectStudent(npm);
		//studentService.addStudent(student);
		if (student == null) {
		return "error";
		}else{
			studentService.deleteStudent(npm);
			return "delete";
		}
	}
```

Method pada service:

```
	@Override
	public void deleteStudent(String npm) {
		// Implement
		for (int i = 0; i < studentList.size(); i++) {
			if (studentList.get(i).getNpm().equals(npm)) {
				System.out.println(studentList.get(i));
				studentList.remove(i);
				break;
			}
		}
	}
```

Pada studentcontroller, dibuat method seperti di atas. Cara kerja method adalah mendapatkan objek student dengan memanggil method select student berdasarkan npm pada service. Setelah itu, dilakukan validasi dari objek student. Validasi pertama mengecek apakah objek student null atau tidak. Jika null, maka akan ter-*direct* ke halaman error. Jika tidak null, maka akan dilakukan penghapusan. Caranya adalah memanggil delete student dari studentservice. Setelah menghapus, maka akan ter-*direct* ke halaman delete.

Pada inmemorystudentservice, dibuat method seperti di atas. Cara kerja method adalah melakukan looping sebuah list yang berisikan objek student. Jika objek student sesuai dengan apa yang ingin dicari, maka ia akan objek tersebut akan dihapus. Setelah ketemu objek yang dicari, maka looping dihentikan.