Ruth Nolytha Putri Sari
1606955006

RIngkasan Materi Tutorial MVC dan Service

Pada tutorial kali ini membahas konsep MVC dan service yang digunakan. Model utama yang dijadikan objek adalah Student yang memiliki npm, nama, dan gpa. Atribut ini didefinisikan dalam class StudentModel.java dan memiliki method setter dan getter untuk mengaksesnya. Controller yang digunakan pada tutorial kali ini dimiliki oleh student dan diberi nama StudentController.java yang tugasnya untuk merespond HTTP requests dan mengembalikan view. View dan controller menggunakan teknologi Thymeleaf dan Spring Boot.

Pada tutorial 3 ini juga menggunakan service yang menjadi mediator antara controller dengan database. StudentService.java merupakan interface dari layanan yang ada yang berisi kontrak apa saja yang bisa dilakukan. Implementasi method yang ada pada interface tersebut dilakukan pada class InMemoryStudentService.java namun dalam pengerjaannya pada tutorial ini tidak mengakses database sama sekali tapi menggunakan list yang berisi objek student yang ditambahkan menggunakan method addStudent yang akan dibuat.

Membuat Class Model

```
package com.example.tutorial3.model;

public class StudentModel {
	private String name;
	private String npm;
	private double gpa;

	public StudentModel(String npm, String name, double gpa) {
		this.name = name;
		this.npm = npm;
		this.gpa = gpa;
	}

	public void setName(String name) {
		this.name = name;
	}

	public void setNpm(String npm) {
		this.npm = npm;
	}

	public void setGpa(Double gpa) {
		this.gpa = gpa;
	}

	public String getName() {
		return this.name;
	}

	public String getNpm() {
		return this.npm;
	}
```

```
	public Double getGpa() {
		return this.gpa;
	}

}
```

Membuat Service

```
package com.example.tutorial3.service;

import java.util.List;

import com.example.tutorial3.model.StudentModel;

public interface StudentService {
	StudentModel selectStudent(String npm);

	List<StudentModel> selectAllStudents();

	void addStudent(StudentModel student);

}
```

- class InMemoryStudentService

```
package com.example.tutorial3.service;

import java.util.List;

import com.example.tutorial3.model.StudentModel;

import java.util.ArrayList;

public class InMemoryStudentService implements StudentService {
	private static List<StudentModel> studentList = new
ArrayList<StudentModel>();

	@Override
	public StudentModel selectStudent(String npm) {
		//implement
		if(npm != null) {
			for(int i=0; i<studentList.size(); i++) {
				if(studentList.get(i).getNpm() == npm) {
					return studentList.get(i);
				}
			}
			return null;
		}
		else
			return null;
	}

	@Override
	public List<StudentModel> selectAllStudents(){
		return studentList;
	}

	@Override
```

```
        public void addStudent(StudentModel student) {
                studentList.add(student);
        }
}
```

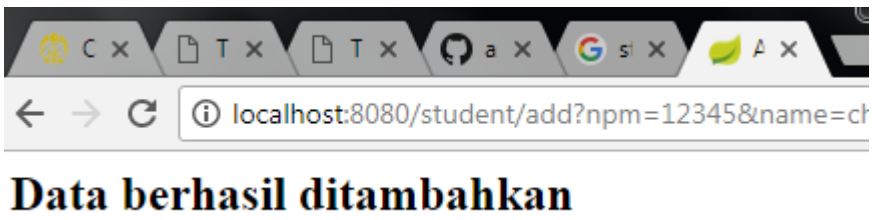
- method selectStudent

```
public class InMemoryStudentService implements StudentService {
        private static List<StudentModel> studentList = new
ArrayList<StudentModel>();

        @Override
        public StudentModel selectStudent(String npm) {
                //implement
                if(npm != null) {
                        for(int i=0; i<studentList.size(); i++) {
                                if(studentList.get(i).getNpm()).equalsIgnoreCase(npm))
{
                                        return studentList.get(i);
                                }
                        }
                        return null;
                }
                else
                        return null;
        }
```

Membuat Controller dan Fungsi Add

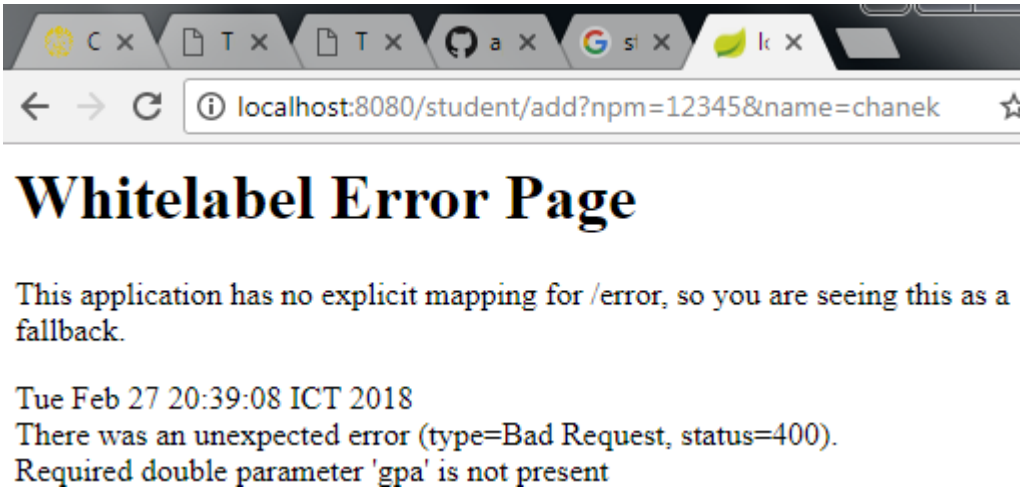
```
@Controller
public class StudentController {
        private final StudentService studentService;

        public StudentController() {
                studentService = new InMemoryStudentService();
        }

        @RequestMapping("/student/add")
        public String add(@RequestParam(value = "npm", required = true) String npm,
@RequestParam(value = "name", required = true) String name, @RequestParam(value =
"gpa", required = true) double gpa) {
                StudentModel student = new StudentModel(npm, name, gpa);
                studentService.addStudent(student);
                return "add";
        }
```

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



Pertanyaan 1: apakah hasilnya? Jika error, tuliskan penjelasan Anda.
Jawaban : Hasil tidak error. Objek student dengan npm, nama dang pa seperti yang tertera pada parameter url telah berhasil ditambahkan kedalam studentList

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



Pertanyaan 2: apakah hasilnya? Jika error, tuliskan penjelasan Anda
Jawaban: Hasil error, karena gpa merupakan RequestParam yang required. Nilai gpa harus di pass melalui pemanggilan url pada browser agar objek mahasiswa baru dapat ditambahkan kedalam studentList

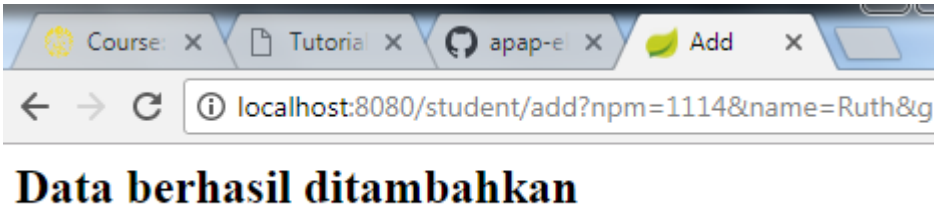
Method View by NPM

```
	@RequestMapping("/student/view")
	public String view(Model model, @RequestParam(value = "npm", required =
true) String npm) {
		StudentModel student = studentService.selectStudent(npm);
		model.addAttribute("student", student);
		return "view";
	}
}
```

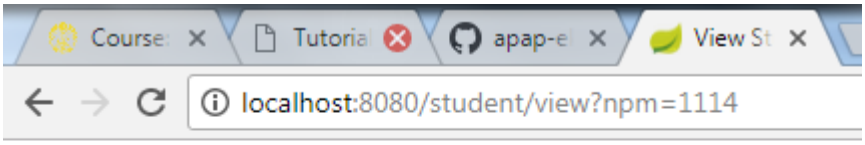
View.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
	<title>View Student by NPM</title>
</head>
<body>
	<h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
	<h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
	<h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
</body>
</html>
```

- Jalankan program dan buka
http://localhost:8080/student/add?npm=1114&name=Ruth&gpa=4 lalu buka



localhost:8080/student/view?npm=1114



NPM = 1114

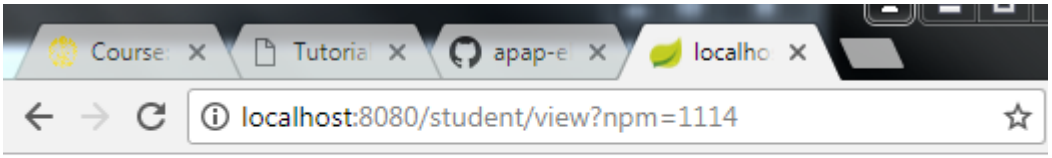
Name = Ruth

GPA = 4.0

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

Jawaban: Data student dengan npm 1114 yang di pass melalui url browser muncul karena pada controller dipanggil method selectStudent yang ada pada studentService untuk mengembalikan objek student dengan npm 1114 dan ditampilkan pada view.

- Coba matikan program dan jalankan kembali serta buka localhost:8080/student/view?npm=1114



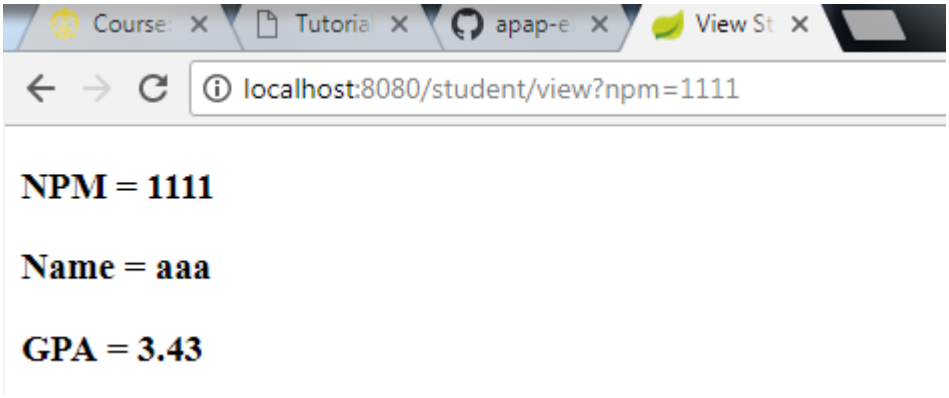
Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Feb 27 21:29:17 ICT 2018
There was an unexpected error (type=Internal Server Error, status=500).
Exception evaluating SpringEL expression: "student.npm" (view:7)

Pertanyaan 4: apakah data Student tersebut muncul? Jika tidak, mengapa?
Jawaban : Data student tidak muncul, karena setelah program dimatikan, isi studentList yang tadinya telah berisi daftar student yang ditambahkan, tidak ada lagi. studentList akan kembali kosong ketika program di-restart. Kecuali jika service mengakses data student yang disimpan dalam database, maka ketika program di-restart pun, data sudent tidak akan hilang.

- Coba tambahkan data Student lainnya dengan NPM yang berbeda.

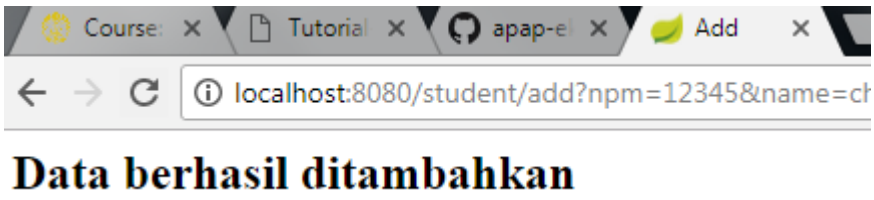


Method View All

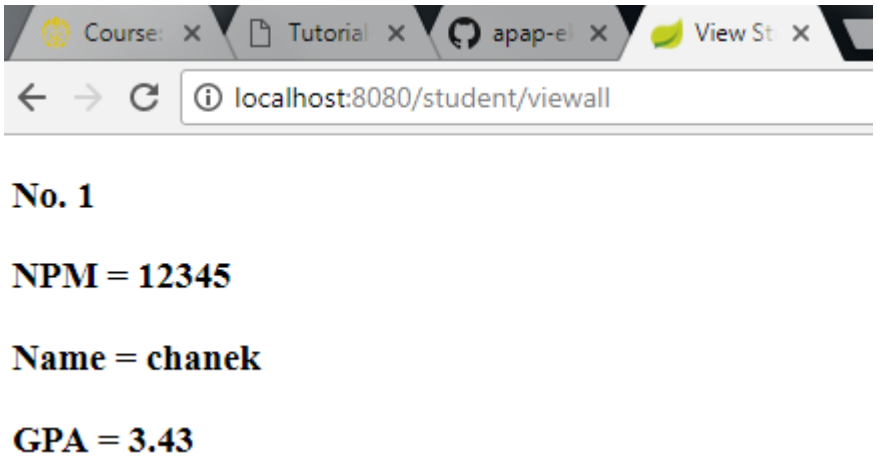
```
@RequestMapping("/student/viewall")
	public String viewAll(Model model) {
		List<StudentModel> students = studentService.selectAllStudents();
		model.addAttribute("students", students);
		return "viewall";
	}
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
	<title>View Student by NPM</title>
</head>
<body>
	<div th:each="student, iterationStatus: ${students}">
	<h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
	<h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
	<h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
	<h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
	<hr/>
	</div>
</body>
</html>
```

- Jalankan program dan buka localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43

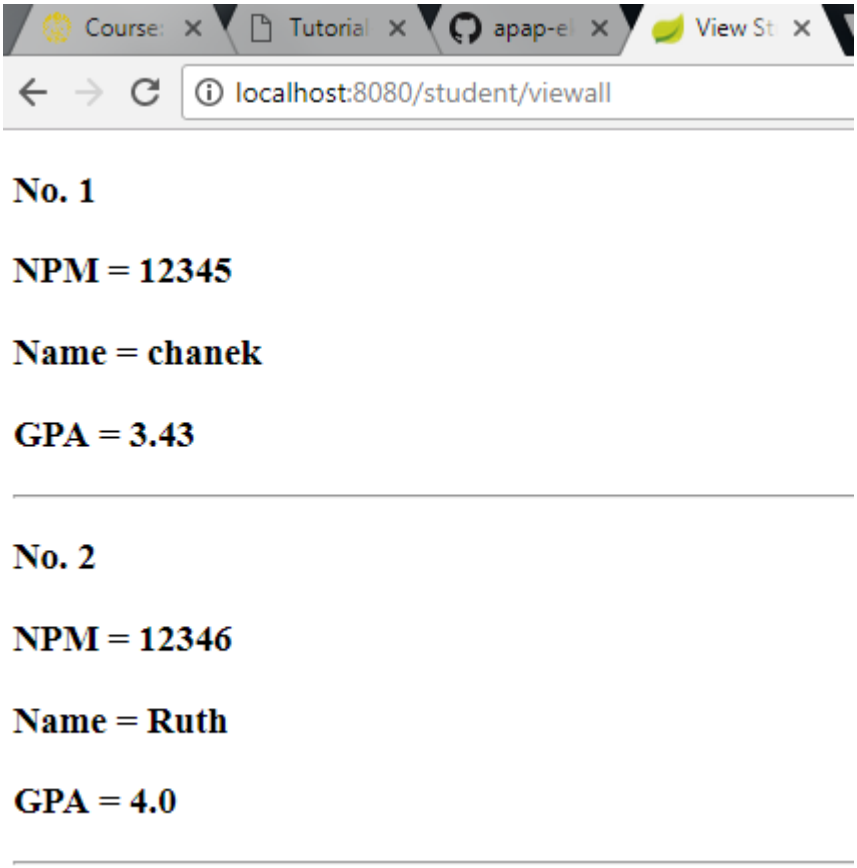


- lalu buka localhost:8080/student/viewall



Pertanyaan 5: apakah data Student tersebut muncul?
Jawaban: Ya. Karena viewAll mengembalikan semua objek student yang terdapat pada studentList. Begitu pula dengan student yang baru ditambahkan, akan ditampilkan datanya ketika viewAll dipangil.

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



Pertanyaan 6: Apakah semua data Student muncul?
Jawaban : Semua data student yang disimpan dalam studentList akan muncul pada viewAll. Hal ini karena method yang dijalankan saat url tersebut dipanggil mengembalikan semua isi studentList yang berisi objek student.

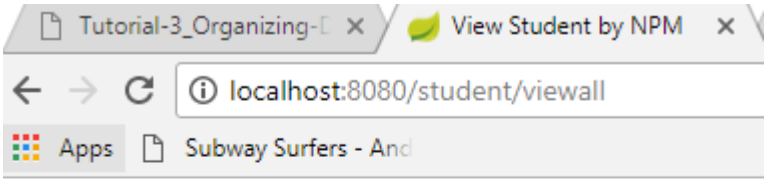
Latihan

1. Pada StudentController tambahkan sebuah method view Student dengan menggunakan Path Variable. Misalnya, Anda ingin memasukkan data seorang Student yang memiliki NPM 14769, untuk melihat data yang baru dimasukkan tersebut dapat mengakses halaman localhost:8080/student/view/14769.

```
@RequestMapping({"/student/view/", "student/view/{npm}"})
	public String viewNpm(@PathVariable Optional<String> npm, Model
model) {
		if(npm.isPresent()) {
			StudentModel student =
studentService.selectStudent(npm.get());
			if(student == null) {
				model.addAttribute("npm", npm);
				return "notFound";
			} else {
				if(npm.get().equalsIgnoreCase(student.getNpm())){
					model.addAttribute("student", student);
					return "view";
				}
				else {
```

```
                        model.addAttribute("npm", npm);
                        return "notFound";
                    }
                }
            } else {
                model.addAttribute("npm", npm);
                return "notFound";
            }
        }
```

Method ini menggunakan anotasi PathVariable sebagai cara agar parameter yang di pass melalui url browser bisa bersifat Optional. Ketika user ingin melihat data student dengan npm yang benar/salah/null akan bisa diatasi karena path url nya tidak harus diisi dengan String npm. Ketika npm yang di pass dari url browser tidak cocok pada data student yang ada pada studentList, maka user akan dialihkan ke halaman error page: notFound.html. Tapi ketika npm yang di pass cocok dan ada pada objek student di studentList, maka halaman akan menampilkan data student dengan npm yang dimaksud.



No. 1

NPM = 1111

Name = Ruth

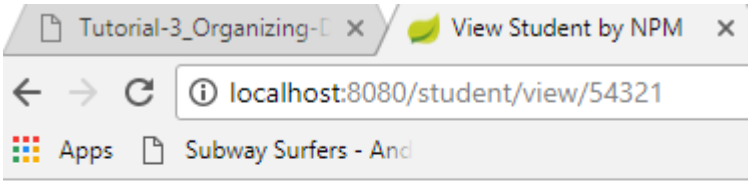
GPA = 4.0

No. 2

NPM = 54321

Name = Rita

GPA = 4.0

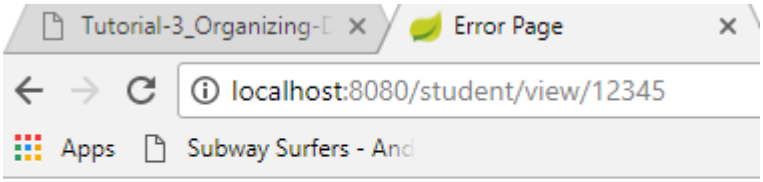


NPM = 54321

Name = Rita

GPA = 4.0

Jika nomor NPM tidak diberikan atau tidak ditemukan kembalikan halaman error yang berisi informasi bahwa nomor NPM kosong atau tidak ditemukan.



Student's NPM Not Found

Tambahkan fitur untuk melakukan delete Student berdasarkan NPM. Misalnya, setelah melakukan add Student pada soal nomor 1, cobalah untuk melakukan delete data tersebut dengan mengakses halaman localhost:8080/student/delete/14769. Tampilkan sebuah halaman yang memberikan informasi bahwa data tersebut telah berhasil dihapus.

- Menambahkan kontrak deleteStudent pada interface studentService

```
public interface StudentService {
    StudentModel selectStudent(String npm);

    List<StudentModel> selectAllStudents();

    void addStudent(StudentModel student);

    void deleteStudent(String npm);

}
```

Hal ini dilakukan agar service delete student bisa dibuat implementasinya dan digunakan ketika dibutuhkan

- Membuat implementasi deleteStudent dengan parameter npm

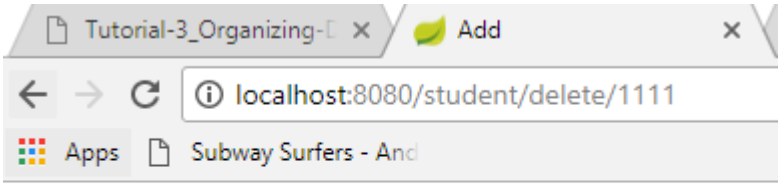
```java
@Override
public void deleteStudent(String npm) {
    for(int i=0; i<studentList.size(); i++) {
        if((studentList.get(i).getNpm()).equalsIgnoreCase(npm)) {
            studentList.remove(i);
        }
    }
}
```

Hal ini dilakukan untuk mengimplementasikan method yang ada pada interface studentService agar bisa digunakan ketika dibutuhkan. Student dengan npm yang sesuai pada parameter method akan di hapus dari list dengan menggunakan fungsi remove yang telah disediakan Java.

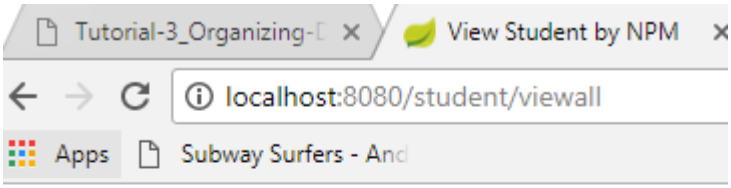
- Membuat controller untuk deleteStudent

```java
@RequestMapping({"/student/delete/", "student/delete/{npm}"})
public String delete(@PathVariable Optional<String> npm, Model model) {
    if(npm.isPresent()) {
        StudentModel student = studentService.selectStudent(npm.get());
        if(student == null) {
            model.addAttribute("npm", npm);
            return "notFound";
        } else {
            if(npm.get().equalsIgnoreCase(student.getNpm())){
                studentService.deleteStudent(npm.get());
                return "delete";
            }
            else {
                model.addAttribute("npm", npm);
                return "notFound";
            }
        }
    } else {
        model.addAttribute("npm", npm);
        return "notFound";
    }
}
```

Mirip dengan viewByNpm, method delete pada controller ini juga menggunakan npm sebagai parameter untuk mencari student yang ingin dihapus pada studentList. Method ini menggunakan anotasi PathVariable sebagai cara agar parameter yang di pass melalui url browser bisa bersifat Optional. Ketika user ingin menghapus data student dengan npm yang benar/salah/null akan bisa diatasi karena path url nya tidak harus diisi dengan String npm yang tepat. Ketika npm yang di pass dari url browser tidak cocok pada data student yang ada pada studentList, maka user akan dialihkan ke halaman error page: notFound.html. Tapi ketika npm yang di pass cocok dan ada pada objek student di studentList, maka halaman akan menghapus data student dengan npm yang dimaksud dari studentList.



Data berhasil dihapus



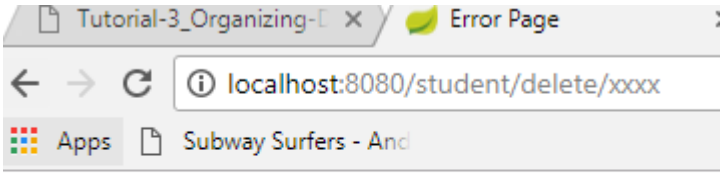
No. 1

NPM = 54321

Name = Rita

GPA = 4.0

Jika nomor NPM tidak diberikan atau tidak ditemukan kembalikan halaman error yang berisi informasi bahwa nomor NPM kosong atau tidak ditemukan dan proses delete dibatalkan.



Student's NPM Not Found