Nama : Winda Dumaria Simanjuntak
NPM : 1706107075

Tutorial 3
Menggunakan Model dan Service pada Project Spring Boot

Ringkasan dari materi

- Konsep MVC (model-view-controller) dapat diterapkan di Spring Boot.
- Dalam penerapan MVC ada sebutan service yang berfungsi sebagai mediator antara controller dan database. Service bisa saja merupakan interface yang mendefinisikan method-method apa saja yang dapat dilakukan untuk memanipulasi model.
- Pada service layer, disimpan business logic yang digunakan untuk mengolah data yang terdapat dalam database. Pengolahan ini meliputi kalkulasi data, manipulasi user input kedalam database, dsbnya. Pada praktikum kali ini belum berhubungan dengan database, penyimpanan data dalam ArrayList tapi sudah ada pengolahan data di dalam service layer.

Membuat Class Model

1. Buatlah package com.example.tutorial3.model dan class StudentModel di dalam package tersebut sesuai instruksi tutorial.
2. Tambahkan method constructor, setter, dan getter ke dalam class StudentModel seperti berikut.

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

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

	public String getNPM () {return this.npm;}

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

	public double getGPA () {return this.gpa;}

	public void setGPA (double gpa) {this.gpa=gpa;}
}
```

.

Nama : Winda Dumaria Simanjuntak
NPM : 1706107075

Membuat Service

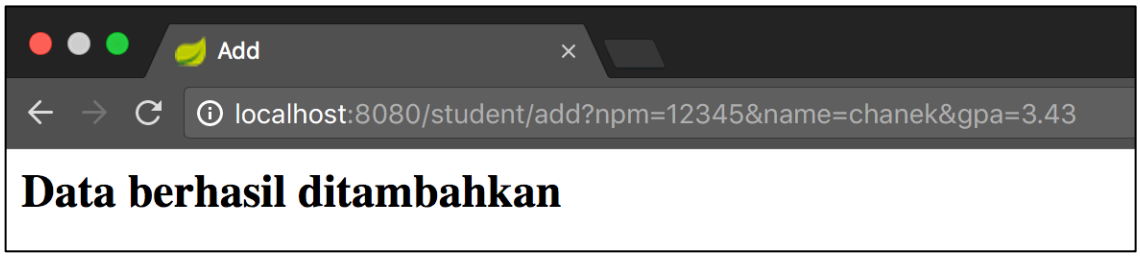
1. Buatlah package com.example.tutorial3.service dan interface StudentService di dalam package tersebut sesuai instruksi tutorial.
2. Buatlah class InMemoryStudentService.java yang mengimplementasikan interface StudentService yg sudah dibuat tadi. Berikut spesifikasi dari class InMemoryStudentService.

```
package com.example.tutorial3.service;
import java.util.ArrayList;
import java.util.List;

import com.example.tutorial3.model.StudentModel;

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

	@Override
	public StudentModel selectStudent(String npm) {
		StudentModel student=null;
		for(StudentModel a: studentList) {
			if(a.getNPM().equalsIgnoreCase(npm))
				student=a;
		}
		return student;
	}

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

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

	@Override
	public StudentModel deleteStudent(String npm) {
		StudentModel student = student = selectStudent(npm);
		if(student !=null)
			studentList.remove(student);
		return student;
	}

}
```

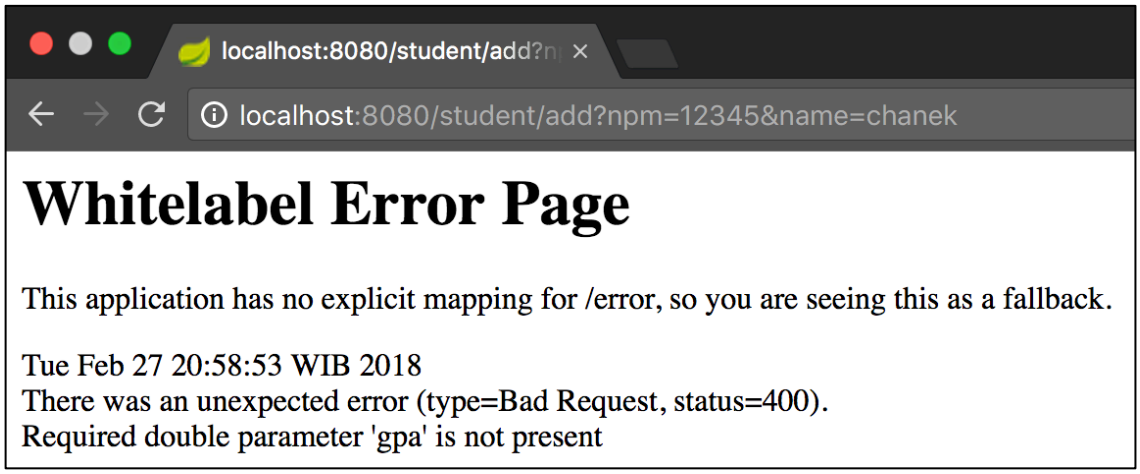
Nama : Winda Dumaria Simanjuntak
NPM : 1706107075

Membuat Controller

1. Buatlah package com.example.tutorial3.controller dan class StudentService di dalam package tersebut sesuai instruksi tutorial. Lalu pada directory resources/templates tambahkan add.html
2. Jalankan program dan buka
   a. localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43
      Pertanyaan 1: apakah hasilnya? Hasilnya adalah data berhasil ditambahkan dengan parameter yang dimasukkan melalui URL tersebut seperti berikut ini .

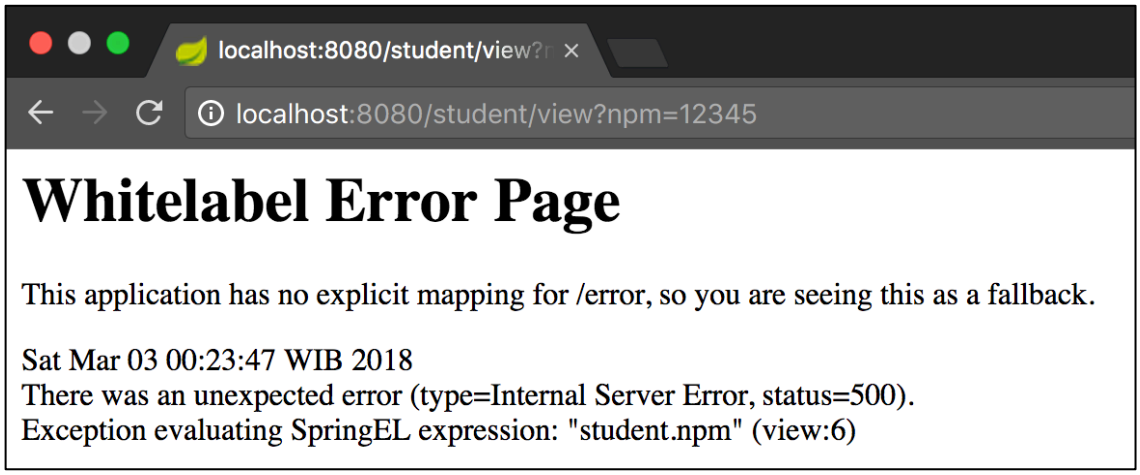


   b. localhost:8080/student/add?npm=12345&name=chanek
      Pertanyaan 2: apakah hasilnya? Hasilnya adalah error karena kurang memasukkan parameter gpa pada URL tersebut padahal parameter gpa sifatnya required.



3. Tambahkan method view by NPM pada class StudentController dan pada directory resources/templates tambahkan view.html sesuai instruksi tutorial.
4. Jalankan program dan buka
   a. 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? Tidak, karena saat mau menampilkan data atribut dari student yg baru saja ditambahkan tidak diperbolehkan karena private. Berikut tampilan yang akan muncul ketika melakukn hal di atas.

Nama : Winda Dumaria Simanjuntak
NPM : 1706107075

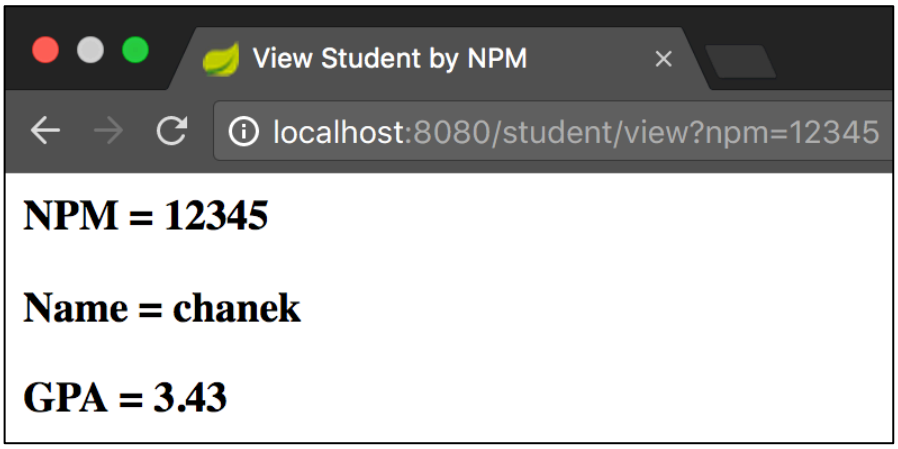


b. Coba matikan program dan jalankan kembali serta buka localhost:8080/student/view?npm=12345 Pertanyaan 4: apakah data Student tersebut muncul? Tidak, alasannya sama seperti point 4a sebelumnya.

c. Untuk mengatasi hal tersebut kita bisa memanfaatkan method getter yang sudah dibuat pada class Student untuk bisa menampilkan datanya pada view.html seperti berikut.

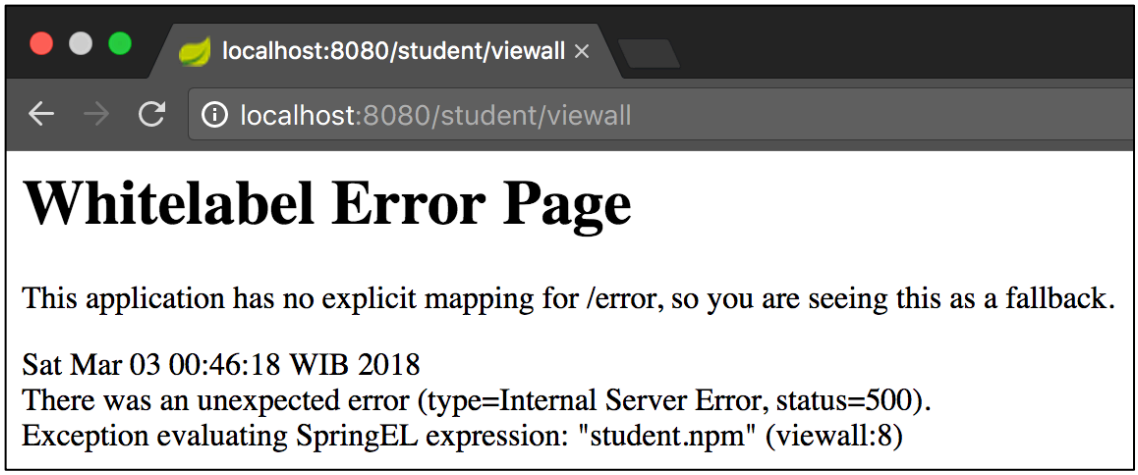
```
<html>
	<head>
		<title>View Student by NPM</title>
	</head>
	<body>
		<h3 th:text="'NPM = '+${student.getNPM()}">Student NPM</h3>
		<h3 th:text="'Name = '+${student.getName()}">Student Name</h3>
		<h3 th:text="'GPA = '+${student.getGPA()}">Student GPA</h3>
	</body>
</html>
```

Coba matikan dan ulangi lagi langkah pada point 4a maka data student yang ditambahkan berhasil ditampilkan seperti berikut ini.



Nama : Winda Dumaria Simanjuntak
NPM : 1706107075

5. Tambahkan method viewAll pada class StudentController dan pada directory resources/templates tambahkan viewall.html sesuai instruksi tutorial.
6. Jalankan program dan buka
   a. localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/viewall
   Pertanyaan 3: apakah data Student tersebut muncul? Tidak, alasannya sama seperti point 4a sebelumnya. Berikut tampilan yang akan muncul ketika melakukn hal di atas.

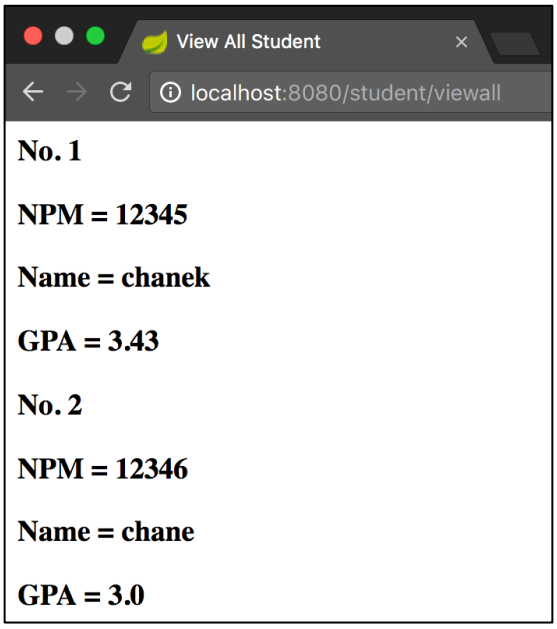


   b. Coba tambahkan data Student lainnya dengan NPM yg berbeda, lalu buka localhost:8080/student/viewall
   Pertanyaan 3: apakah data semua Student tersebut muncul? Tidak, alasannya sama seperti point 4a sebelumnya.
   c. Untuk mengatasi hal tersebut kita bisa memanfaatkan method getter yang sudah dibuat pada class Student untuk bisa menampilkan datanya pada view.html seperti berikut.

```
<html>
	<head>
		<title>View All Student</title>
	</head>
	<body>
		<div th:each="student,iterationStatus: ${students}">
			<h3 th:text="'No. '+${iterationStatus.count}">No.1</h3>
			<h3 th:text="'NPM = '+${student.getNPM()}">Student NPM</h3>
			<h3 th:text="'Name = '+${student.getName()}">Student Name</h3>
			<h3 th:text="'GPA = '+${student.getGPA()}">Student GPA</h3>
		</div>
	</body>
</html>
```

Coba matikan dan ulangi lagi langkah pada point 5a, 5b maka semua data student yang ditambahkan berhasil ditampilkan seperti berikut ini.

Nama : Winda Dumaria Simanjuntak
NPM : 1706107075



Latihan

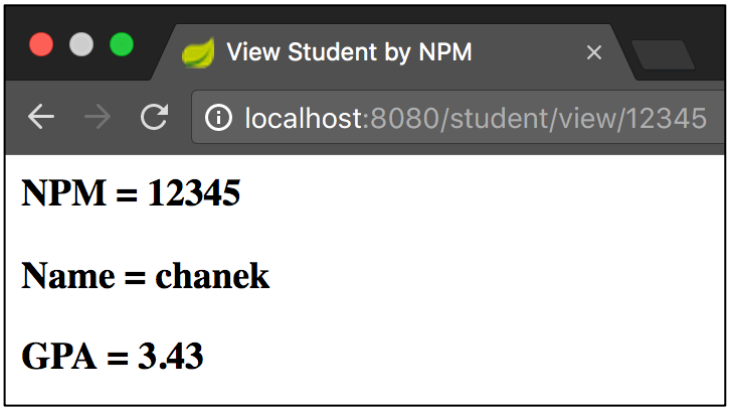
1. Tambahkan method view Student dengan menggunakan Path Variable dengan langkah berikut ini.
   a. Tambah method viewStudent pada class StudentController seperti berikut.

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

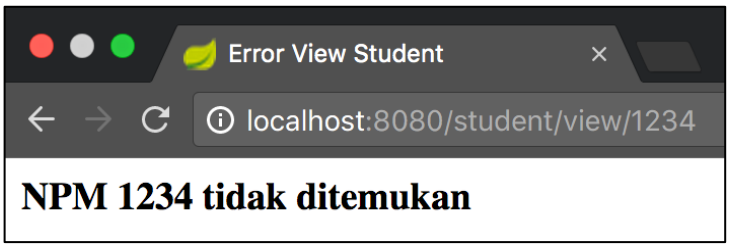
   b. Tambahkan view error_view.html untuk menampilkan hasil dari proses view student tidak berhasil seperti berikut.

```
<html>
	<head>
		<title>Error View Student</title>
	</head>
	<body>
		<h3 th:text="'NPM '+${npm}+' tidak ditemukan'">Student NPM</h3>
	</body>
</html>
```

c. Coba jalankan dan buka localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/view/12345
Berikut tampilan dari langkah di atas:



d. Coba jalankan dan buka localhost:8080/student/view/1234. Karna NPM 1234 belum ditambahkan maka akan muncul tampilan seperti berikut.



2. Tambahkan method delete dengan menggunakan Path Variable dengan langkah berikut ini.
   a. Tambah method delete pada class StudentController seperti berikut.

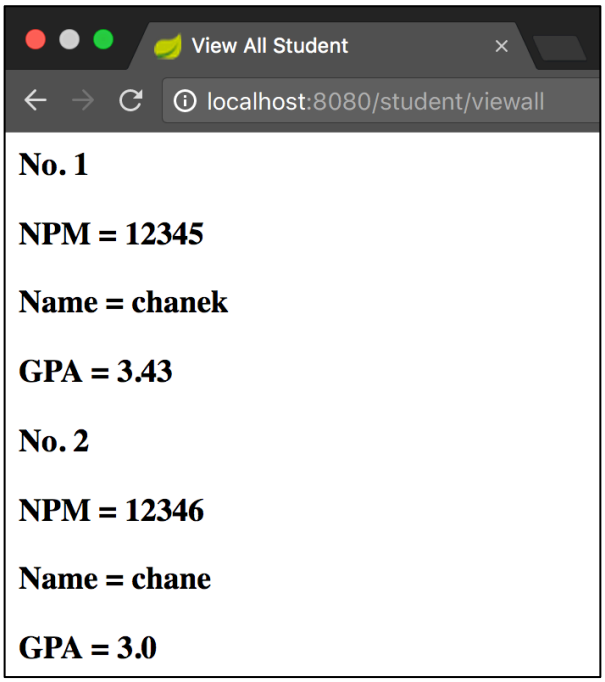
```
@RequestMapping("/student/delete/{npm}")
	public String delete(Model model, @PathVariable Optional<String> npm) {
		if(npm.isPresent()) {
			StudentModel student =
studentService.deleteStudent(npm.get());
			if(student != null) {
				model.addAttribute("npm", npm.get());
				return "delete";
			}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

   b. Tambahkan view delete.html dan error_delete.html untuk menampilkan hasil dari proses delete student berhasil dan tidak berhasil seperti berikut.

```
<html>
	<head>
		<title>Delete Student by NPM</title>
	</head>
	<body>
		<h3 th:text="'Data Student dengan NPM '+${npm}+' berhasil
dihapus'">Student NPM</h3>
	</body>
</html>
```

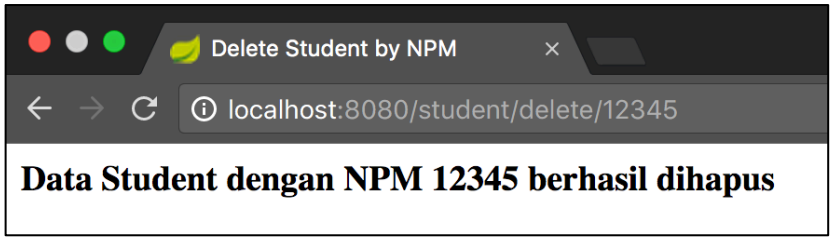
```
<html>
	<head>
		<title>Error Delete Student</title>
	</head>
	<body>
		<h3 th:text="'NPM '+${npm}+' tidak ditemukan. Proses delete
dibatalkan'">Student NPM</h3>
	</body>
</html>
```

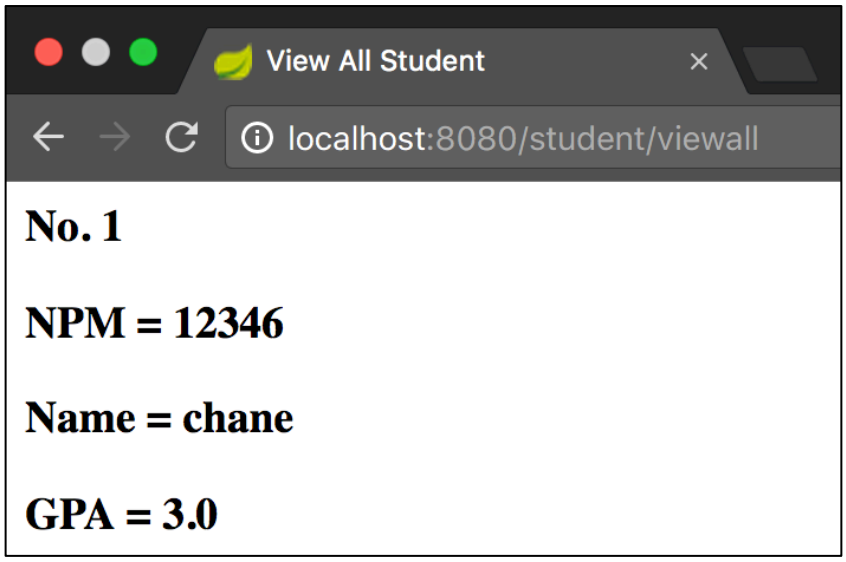
c. Coba jalankan dan buka localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/add?npm=12346&name=chane&gpa=3.0, kemudian localhost:8080/student/viewall
Berikut tampilan dari langkah di atas:



d. Coba jalankan dan buka localhost:8080/student/delete/12345 lalu buka localhost:8080/student/viewalll

Berikut hasil dari langkah tersebut





e. Coba jalankan dan buka localhost:8080/student/delete/1234. Karna NPM 1234 belum ditambahkan maka akan muncul tampilan seperti berikut.