Membuat controller dan fungsi Add

Ringkasan: pada bagian ini kita mempelajari bagaiamana membuat fungsi untuk menambahkan data kedalam model. Kita akan mempelajari bagaimana konsep MVC diimplementasikan. Langkah – langkahnya adalah sebagai berikut:

1. Buat class StudentModel seperti berikut ini :

```
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 name;
    }

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

    public String getNpm() {
        return npm;
    }

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

    public double getGpa() {
        return gpa;
    }

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

2. Buatlah service dan letakkan di package com.example.tutorial3.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); }
```

Service ini berfungsi sebagai wadah untuk mendefinisikan method-method apa saja yang dapat dilakukan untuk memanipulasi kelas student.

3. Buatlah InMemoryStudentservice pada package yang sama

```
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) {
        for (int i=0;i<studentList.size();i++) {
            if (studentList.get(i).getNpm().equals(npm)) {
                return studentList.get(i);
            }
        }
        return null;
    }

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

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

Kelas InMemoryStudentService merupakan implementasi dari interface StudentService.

4. Buatlah controller di pacakge com.example.tutorial3.controller. Tambahkan method add yang menerima parameter name, npm, dan gpa dengan menggunakan request method GET. Buat kelas studentController dengan isi sebagai berikut:

```
package com.example.tutorial3.controller;

import java.util.List;
import java.util.Optional;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
```

```
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.example.tutorial3.service.InMemoryStudentService;
import com.example.tutorial3.service.StudentService;
import com.example.tutorial3.model.StudentModel;

@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";
	}
}
```

5. Berikutnya pada direktori resources/templates tambahkan file add.html

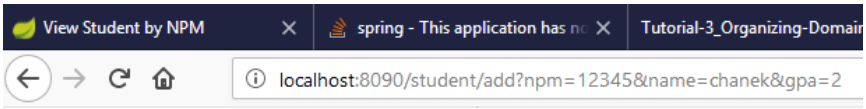
```
<!DOCTYPE html>
<html>
    <head>
        <title>Insert title here</title>
    </head>
<body>
    <h2>Data berhasil ditambahkan</h2>
</body>
</html>
```

6. Jalankan program dan buka

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

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

   Jawab: Hasilnya data berhasil ditambahkan. Tidak terjadi error

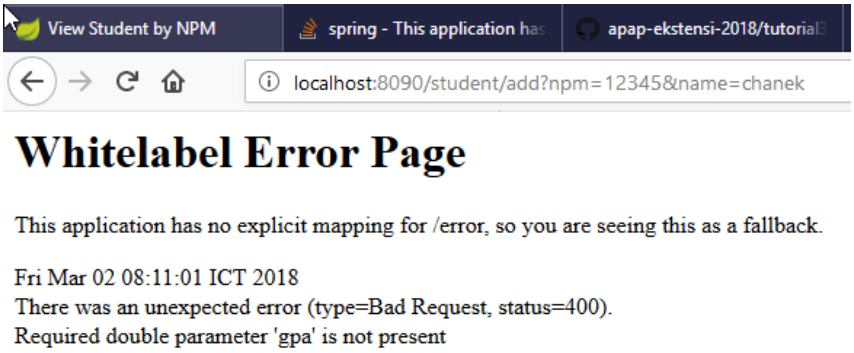


Data berhasil ditambahkan

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

   Pertanyaan 2: apakah hasilnya? Jika error, tuliskan penjelasan Anda.

   Jawab: Hasilnya terjadi error karena parameter gpa tidak disediakan.



Menampilkan data berdasarkan NPM

Ringkasan: pada bagian ini kita akan belajar menggunakan perintah pencarian untuk mencari salah satu data dari mahasiswa.

1. Pada class StudentController tambahkan method seperti berikut ini:

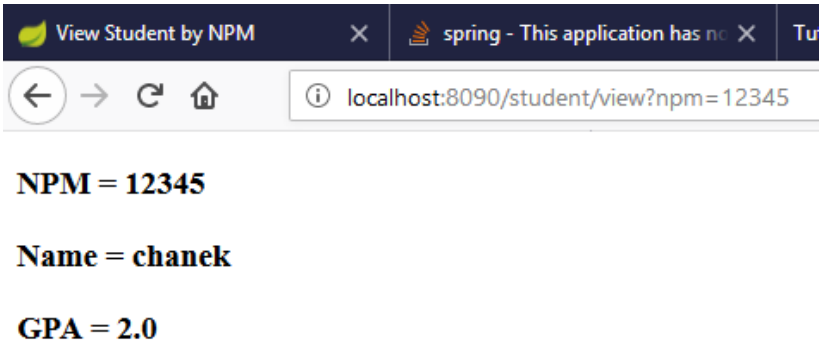
```
@RequestMapping("/student/view")
public String view(Model model, @RequestParam(value="npm", required=true) String npm) {
    StudentModel student = studentService.selectStudent(npm);
    model.addAttribute("student", student);
    return "view";
}
```

2. Lalu pada direktori resources/templates tambahkan view.html dengan isi seperti berikut:

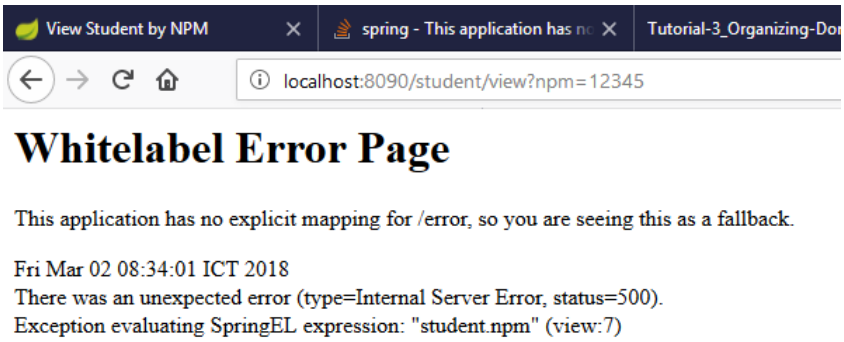
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Student by NPM</title>
    </head>
<body>
    <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
    <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
    <h3 th:text="'GPA = ' + ${student.gpa}">Student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

3. Jalankan program dan buka
   localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/view?npm=12345

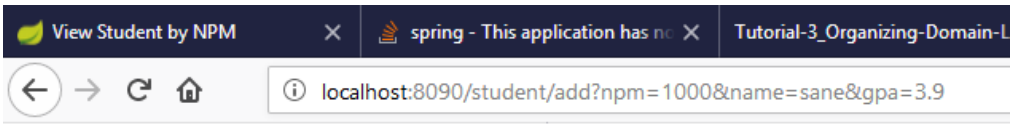
   Pertanyaan 3: apakah data Student tersebut muncul?jika tidak, mengapa?
   Muncul

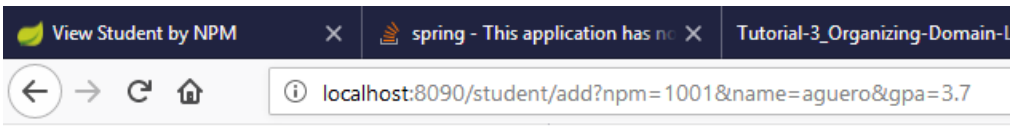


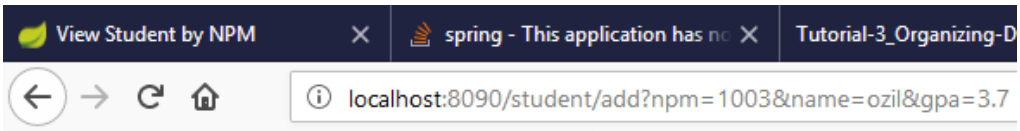
4. Coba matikan program dan jalankan kembali serta buka
   localhost:8080/student/view?npm=12345
   Pertanyaan 3: apakah data Student tersebut muncul?jika tidak, mengapa?
   Tidak muncul, karena data disimpan di list. List hanya menampung data di memory. Ketika program dihentikan, data yang ada di list akan terhapus.



5. Tambahkan data student lainnya dengan npm yang berbeda.







Method View All

Ringakasan: pada tutorial ini kita akan menampilkan semua data yang telah dimasukkan ke dalam list.

1. Pada class studentController tambahkan method berikut

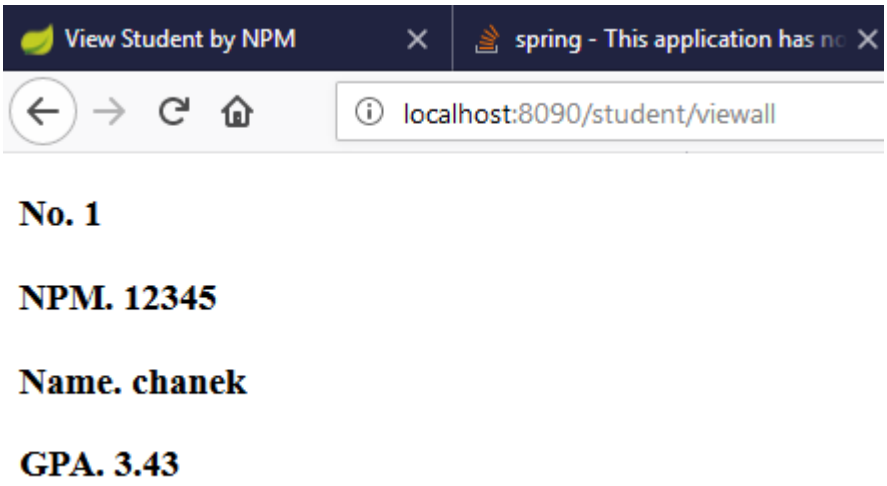
```
@RequestMapping("/student/viewall")
public String viewAll(Model model) {
    List<StudentModel> students = studentService.selectAllStudents();
    model.addAttribute("students", students);
    return "viewall";
}
```

2. Selanjutnya pada direktori resources/templates tambahkan viewall.html dengan isi sebagai berikut:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View All</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>
    </div>
</body>
</html>
```

3. Jalankan program dan buka
   localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/viewall

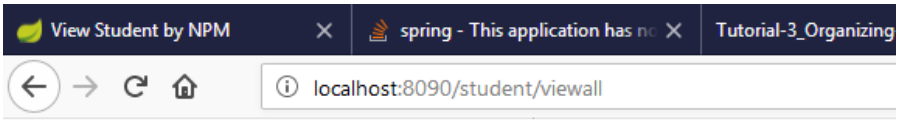
   Pertanyaan 5: apakah data Student tersebut muncul?jika tidak, mengapa?
   Muncul



4. Coba tambahkan data student lainnya dengan npm yang berbeda, lalu buka

Localhost:8080/student/viewall,

Pertanyaan 6: apakah semua data student muncul
muncul



No. 1

NPM. 12345

Name. chanek

GPA. 3.43

No. 2

NPM. 12346

Name. chanek

GPA. 3.43

No. 3

NPM. 12347

Name. david

GPA. 5.0

No. 4

NPM. 8888

Name. ramirez

Latihan

1. pada studentController tambahkan sebuah *method view* Student dengan menggunakan **Path Variable**. Format path variable nya adalah localhost:8080/student/view/12345. Jika npm tidak diberikan atau tidak ditemukan kembalikan halaman error yang berisi informasi bahwa nomor NPM kosong atau tidak diberikan.
   Langkah – langkah:
   a. tambahkan method untuk menampilkan data mahasiswa dengan format url localhost:8080/student/view/12345

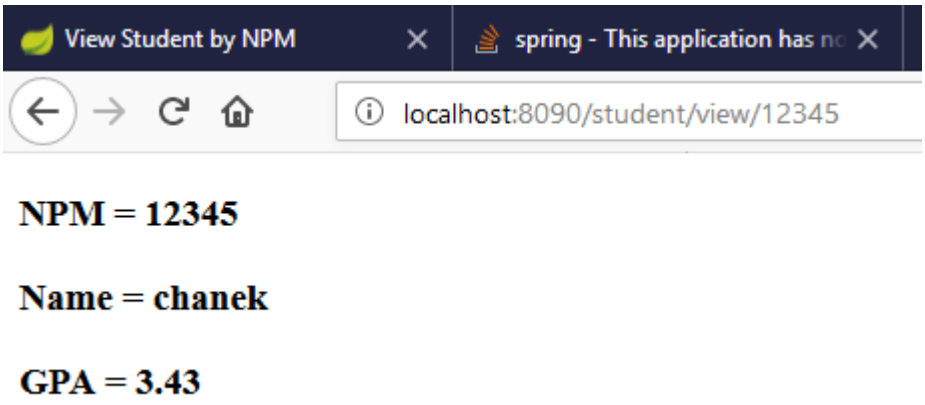
```
@RequestMapping("/student/view/{npm}")
public String view2(@PathVariable Optional<String> npm, Model model) {

    StudentModel student = studentService.selectStudent(npm.get());
    if(student == null) {
        return "kosong";
    }
    model.addAttribute("student", student);
    return "view";
}
```

Pada method ini jika tidak ada data npm pada model student maka akan mengembalikan view kosong.html

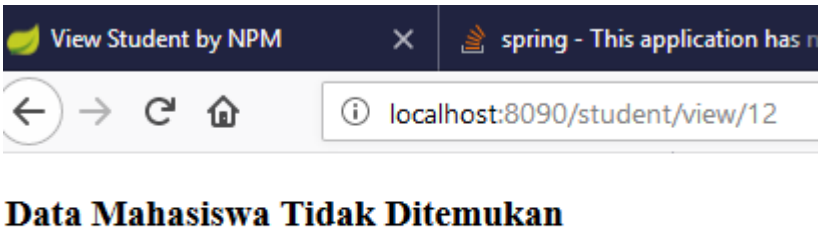
b. buatlah view kosong.html dengan isi seperti berikut ini.

```
<!DOCTYPE html>
<html>
    <head>
        <title>View Student by NPM</title>
    </head>
<body>
    <h3>Data Mahasiswa Tidak Ditemukan</h3>
</body>
</html>
```

c. tambahkan data dan cobalah di browser.



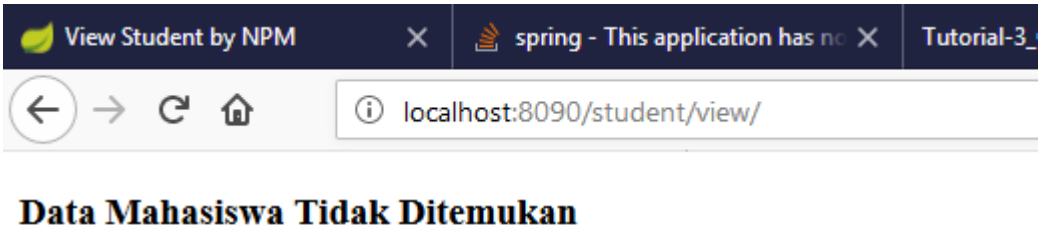
d. apabila npm yang diberikan salah maka akan menampilkan



e. untuk menampilkan data npm yang tidak disediakan maka kita perlu mengubah method pada route /student/view/ yang sudah ada menjadi seperti berikut ini.

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

f. selanjutnya coba testing pada browser.



Data Mahasiswa Tidak Ditemukan

2.

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.

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.

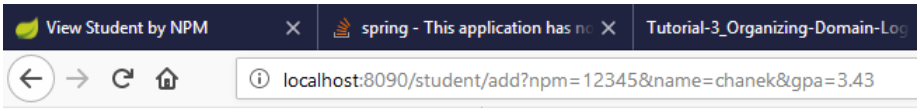
a. Buat method di controller

```
@RequestMapping("/student/delete/{npm}")
public String delete(@PathVariable Optional<String> npm, Model model) {

    StudentModel student = studentService.selectStudent(npm.get());
    if(student == null) {
        return "kosong";
    }
    studentService.removeStudent(student);
    return "hapus";
}

@RequestMapping("/student/delete")
public String delete(Model model) {
    return "kosong";
}
```

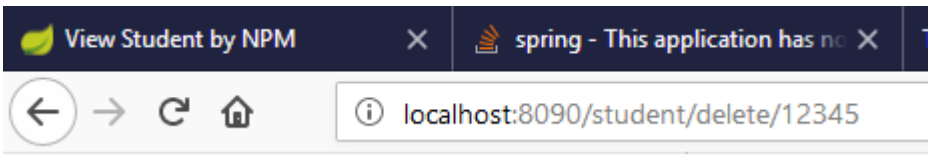
b. Tambahkan view berhasil di hapus.

```
<!DOCTYPE html>
<html>
    <head>
        <title>View Student by NPM</title>
    </head>
<body>
    <h3>Data berhasil dihapus</h3>
</body>
</html>
```

c. Coba tes di browser, dengan menambahkan data terlebih dahulu.

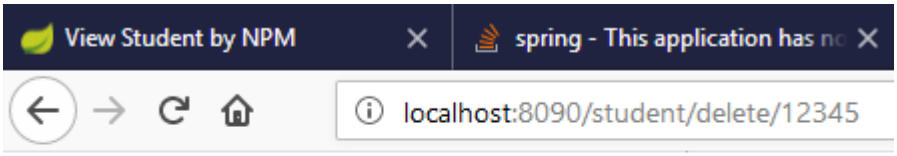


Data berhasil ditambahkan



Data berhasil dihapus

d. Jika data tidak ditemukan maka akan menampilkan halaman error



Data Mahasiswa Tidak Ditemukan