Tutorial 3

Menggunakan Model dan Service pada Project Spring Boot

Write-up tutorial 3:

Pada tutorial ke-3 kali ini, saya mempelajari bagaimana membuat sebuah program dengan menggunakan model dan service pada spring boot dengan menggunakan ArrayList sebagai sistem penyimpanan data. Saya mempelajari bagaimana menambahkan dan menghapus data-data dari dan ke ArrayList tersebut yang nantinya juga akan ditampilkan pada suatu halaman web. Selain itu, saya juga mempelajari bahwa dengan menggunakan ArrayList, jika program dimatikan maka akan membuat data yang ada pada ArrayList tersebut terhapus semua.

4. **Jalankan program dan buka**

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

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

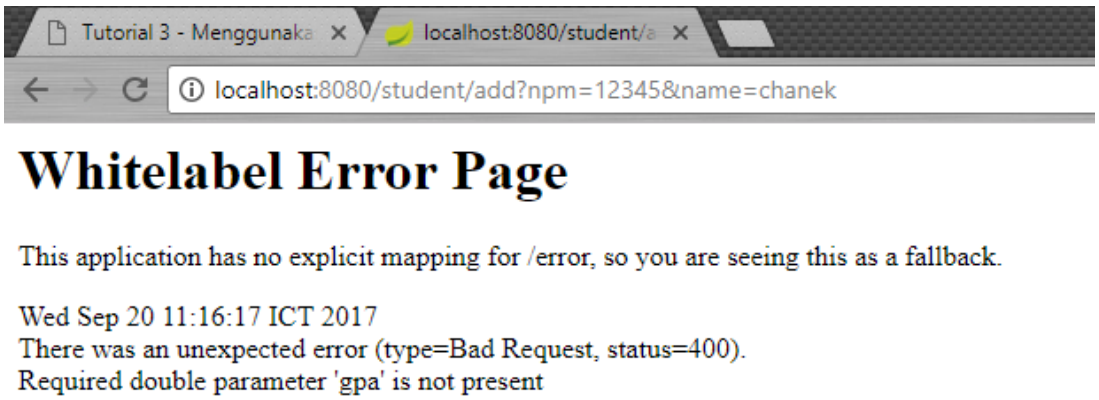
   Halaman menampilkan tampilan sebagai berikut yang menandakan bahwa data student berhasil ditambahkan.

Data berhasil ditambahkan

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

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

   Halaman menampilkan tampilan sebagai berikut. Hal ini terjadi karena atribut gpa sebelumnya sudah diset *required*, karena tidak terdapat nilai untuk atribut tersebut maka terjadi error.



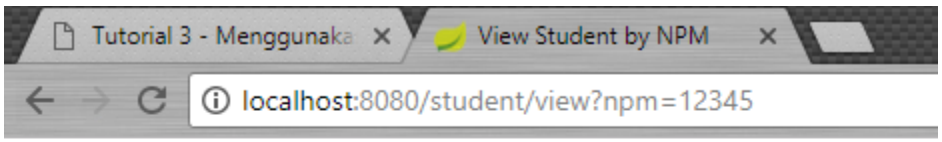
3. **Jalankan program dan buka**

   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?

   Data berhasil masuk dan muncul.

Data berhasil ditambahkan



NPM = 12345

Name = chanek

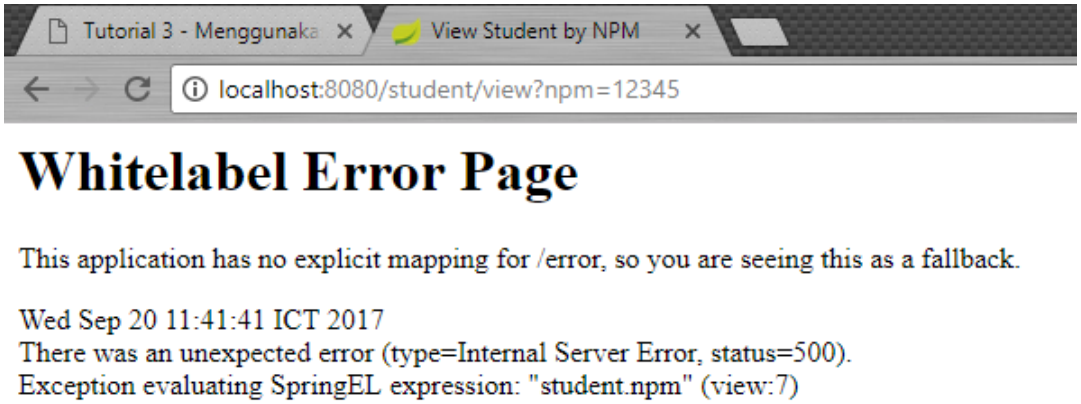
GPA = 3.43

4. **Coba matikan program dan jalankan kembali serta buka**

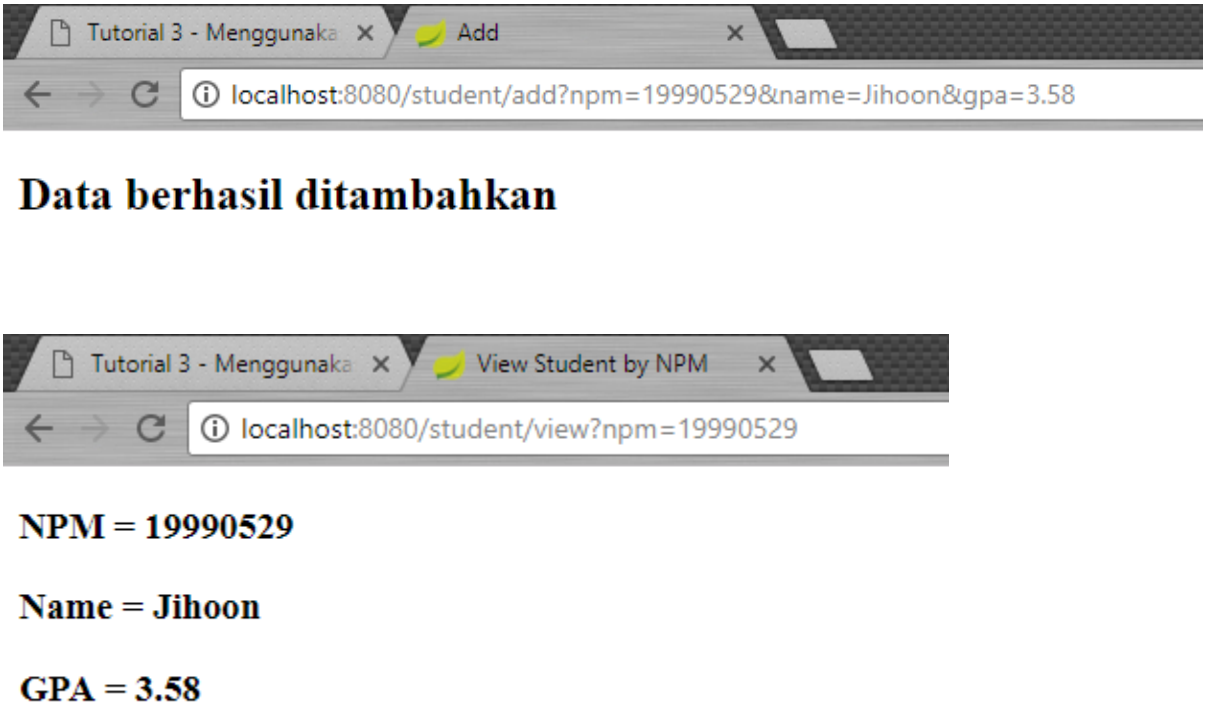
   localhost:8080/student/view?npm=12345

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

Halaman menampilkan error karena tidak ada objek student yang dikembalikan, karena tidak ditemukan. Hal ini dapat terjadi karena pada tutorial ini, data disimpan menggunakan ArrayList yang isinya akan di-*reset* kembali setiap program dijalankan.



5. **Coba tambahkan data Student lainnya dengan NPM yang berbeda.**



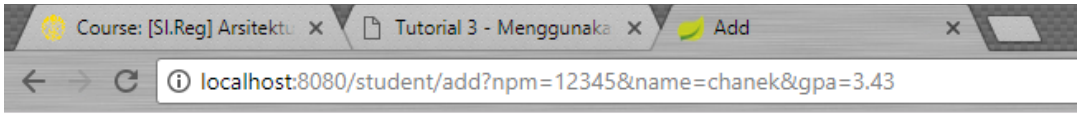
3. **Jalankan program dan buka**

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

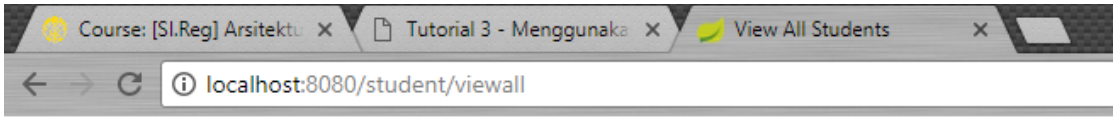
   localhost:8080/student/viewall,

   Pertanyaan 5: apakah data Student tersebut muncul?

   Data berhasil masuk dan muncul.



Data berhasil ditambahkan



No. 1

NPM = 12345

Name = chanek

GPA = 3.43

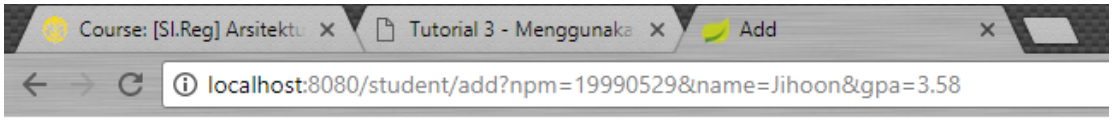
---

4. **Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka**

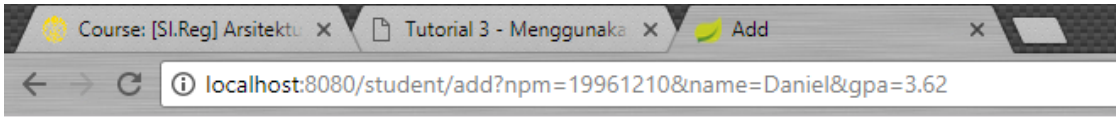
   localhost:8080/student/viewall,

   Pertanyaan 6: Apakah semua data Student muncul?

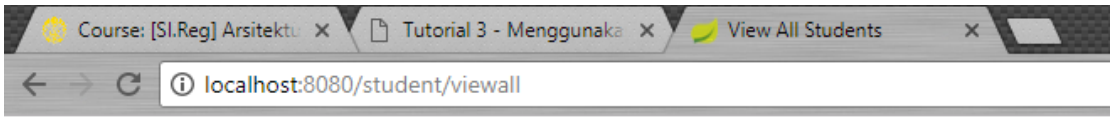
   Halaman berhasil memunculkan seluruh data Student yang ada.



Data berhasil ditambahkan



Data berhasil ditambahkan



No. 1

NPM = 12345

Name = chanek

GPA = 3.43

No. 2

NPM = 19990529

Name = Jihoon

GPA = 3.58

No. 3

NPM = 19961210

Name = Daniel

GPA = 3.62

Method selectStudent

```
@Override
public StudentModel selectStudent(String npm) {
    for(int ii = 0; ii < studentList.size(); ii++) {
        if(studentList.get(ii).getNpm().equals(npm)) {
            return studentList.get(ii);
        }
    }
    return null;
}
```

Method selectStudent digunakan untuk mencari objek student berdasarkan npm yang diberikan. Maka dari itu, yang saya lakukan adalah melakukan *loop* dari ArrayList yang berisi semua data student yang ada dan jika objek student dengan npm yang sama ditemukan, method akan mengembalikan objek student tersebut dan jika tidak ditemukan, method akan mengembalikan nilai null.

Tahap-Tahap Pembuatan Fitur Latihan No. 1

1. Membuat method viewPath di controller, yaitu method yang berfungsi untuk melihat data mahasiswa dengan menggunakan *path variable*. Method ini bekerja dengan cara melihat apakah *user* memberikan atau tidak nilai atribut npm terlebih dahulu. Jika *user* tidak memberikan, maka halaman akan langsung di-*redirect* ke halaman errorviewnpmempty.html yang berisi keterangan bahwa tidak ada NPM yang dimasukkan. Jika *user* memberikan nilai dari atribut npm, maka method akan menggunakan method selectStudent yang sudah dibuat sebelumnya untuk dicek apakah objek student dengan npm yang diberikan ada atau tidak. Jika ada, maka halaman akan di-*redirect* ke view.html yang akan menampilkan data dari objek student tersebut dan jika tidak, maka halaman akan di-*redirect* ke halaman errorviewnpmnotfound.html yang berisi keterangan bahwa objek student dengan NPM yang diberikan tidak ditemukan.

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

2. Setelah itu, hal yang saya lakukan adalah membuat kedua halaman error yang sudah disebutkan sebelumnya yaitu halaman errorviewnpmnotfound.html dan errorviewnpmempty.html.

errorviewnpmnotfound.html

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Student Not Found</title>
</head>
<body>
    <h3 th:text="'Mahasiswa dengan NPM ' + ${npm} + ' tidak ditemukan.'">Student with NPM Not Found</h3>
</body>
</html>
```

errorviewnpmempty.html

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Student Not Found</title>
    </head>
    <body>
        <h2>NPM Kosong. Masukkan NPM untuk melihat data mahasiswa.</h2>
    </body>
</html>
```

Tahap-Tahap Pembuatan Fitur Latihan No. 2

1. Untuk membuat fitur *delete*, hal yang saya lakukan adalah menambah method removeStudent ke kelas *interface* yang nantinya berfungsi untuk menghapus data suatu mahasiswa.

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

import java.util.List;

public interface StudentService {
    StudentModel selectStudent(String npm);

    List<StudentModel> selectAllStudents();

    void addStudent(StudentModel student);

    void removeStudent(StudentModel student);
}
```

2. Lalu, implementasi method removeStudent tersebut di kelas InMemoryStudentService.java.

```java
@Override
public void removeStudent(StudentModel student) {
    studentList.remove(student);
}
```

3. Setelah itu, saya membuat method delete di *controller*, yaitu method yang berfungsi untuk menghapus data mahasiswa dengan menggunakan *path variable*. Method ini bekerja dengan cara melihat apakah *user* memberikan atau tidak nilai atribut npm terlebih dahulu. Jika *user* tidak memberikan, maka halaman akan langsung di-*redirect* ke halaman errordeletenpmempty.html yang berisi keterangan bahwa tidak ada NPM yang dimasukkan dan proses *delete* dibatalkan. Jika *user* memberikan nilai dari atribut npm, maka method akan menggunakan method selectStudent yang sudah dibuat sebelumnya untuk dicek apakah objek student dengan npm yang diberikan ada atau tidak. Jika ada, maka objek student tersebut akan dihapus dengan menggunakan method removeStudent yang telah dibuat sebelumnya, lalu halaman akan di-*redirect* ke delete.html yang akan menampilkan bahwa data mahasiswa tersebut berhasil dihapus. Jika tidak, maka halaman akan di-*redirect* ke halaman errordeletenpmnotfound.html yang berisi keterangan bahwa objek student dengan NPM yang diberikan tidak ditemukan dan proses *delete* dibatalkan.

```java
@RequestMapping(value = {"/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) {
            studentService.removeStudent(student);
            return "delete";
        } else {
            model.addAttribute("npm", npm.get());
            return "errordeletenpmnotfound";
        }
    } else {
        return "errordeletenpmempty";
    }
}
```

3. Setelah itu, hal yang saya lakukan adalah membuat halaman-halaman keterangan yang akan ditampilkan sesuai kriteria-kriteria yang telah disebutkan sebelumnya yaitu delete.html (jika proses *delete* berhasil), errordeletenpmempty.html (jika npm tidak diberikan) dan errordeletenpmnotfound.html (jika mahasiswa dengan npm yang diberikan tidak ditemukan).

delete.html

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Delete</title>
    </head>
    <body>
        <h2>Data berhasil dihapus</h2>
    </body>
</html>
```

errordeletenpmempty.html

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Student Not Found</title>
    </head>
    <body>
        <h2>NPM Kosong. Proses delete dibatalkan.</h2>
    </body>
</html>
```

errordeletenpmnotfound.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Student Not Found</title>
</head>
<body>
    <h3 th:text="'Mahasiswa dengan NPM ' + ${npm} + ' tidak ditemukan.'">Student with NPM Not Found</h3>
    <h3>Proses delete dibatalkan.</h3>
</body>
</html>
```