

WRITE UP

Tutorial 3 - Menggunakan Model dan Service pada Project Spring Boot

Pada tutorial kali ini, kita diajarkan menggunakan Java untuk melakukan programming berbasis web. Framework yang digunakan adalah Spring. Disini kita belajar memanfaatkan konsep MVC (Model, View, Controller) dalam framework Spring. Saya rasa disini caranya sedikit mirip dengan framework PHP yang umumnya kita gunakan dalam mata kuliah PPW. Namun, saya menyadari terdapat beberapa perbedaan pada tutorial ini. Kita baru diajarkan cara memanfaatkan penggunaan inMemory (belum connect ke database), dan juga kita diajarkan konsep adanya service untuk wrap up operasi-operasi yang kita butuhkan untuk manipulasi data yang kita miliki.

Poin-point Tutorial part 1

Latihan Membuat Class Model (StudentModel.java)

```
1 package com.example.tutorial3.model;
2
3 public class StudentModel {
4     private String name;
5     private String npm;
6     private double gpa;
7
8     public StudentModel(String name, String npm, double gpa) {
9         this.name = name;
10        this.npm = npm;
11        this.gpa = gpa;
12    }
13
14    public String getName() {
15        return name;
16    }
17
18    public void setName(String name) {
19        this.name = name;
20    }
21
22    public String getNpm() {
23        return npm;
24    }
25
26    public void setNpm(String npm) {
27        this.npm = npm;
28    }
29
30    public double getGpa() {
31        return gpa;
32    }
33
34    public void setGpa(double gpa) {
35        this.gpa = gpa;
36    }
37 }
38
```

Latihan membuat interface service (StudentService.java)

```
StudentControll  view.html  StudentService.  ⌕
1 package com.example.tutorial3.service;
2 import java.util.List;
3
4
5
6 public interface StudentService {
7     StudentModel selectStudent (String npm);
8
9     List<StudentModel> selectAllStudents();
10    void addStudent(StudentModel student);
11
12    void delete(String npm);
13 }
14
```

Latihan membuat implementasi StudentService yang hanya berjalan didalam memory (belum connect database)

```

InMemoryStudent StudentControll StudentService. StudentModel.java
1 package com.example.tutorial3.service;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import com.example.tutorial3.model.StudentModel;
7
8 public class InMemoryStudentService implements StudentService {
9     private static List<StudentModel> studentList = new ArrayList<StudentModel>();
10
11     @Override
12     public StudentModel selectStudent(String npm) {
13         // TODO Auto-generated method stub
14         for(int i = 0 ; i < studentList.size(); i++) {
15             if(studentList.get(i).getNpm().equals(npm)) {
16                 return studentList.get(i);
17             }
18         }
19         return null;
20     }
21
22     @Override
23     public List<StudentModel> selectAllStudents() {
24         // TODO Auto-generated method stub
25         return studentList;
26     }
27
28     @Override
29     public void addStudent(StudentModel student) {
30         // TODO Auto-generated method stub
31         studentList.add(student);
32     }
33
34     @Override
35     public void delete(String npm) {
36         for(int i = 0 ; i < studentList.size(); i++) {
37             if(studentList.get(i).getNpm().equals(npm)) {
38                 studentList.remove(i);

```

Implementasi method selectSudent(String npm)

Implementasi bisa dilakukan menggunakan looping terhadap List dari studentList. Looping dilakukan hingga menemukan student dengan property npm yang sama seperti parameter. Jika sampai akhir looping data tidak ditemukan, maka akan return null.

```

@Override
public StudentModel selectStudent(String npm) {
    // TODO Auto-generated method stub
    for(int i = 0 ; i < studentList.size(); i++) {
        if(studentList.get(i).getNpm().equals(npm)) {
            return studentList.get(i);
        }
    }
    return null;
}

```

Latihan membuat controller (StudentController.java), serta implementasi fungsi add() untuk menambah data.

```

1 package com.example.tutorial3.controller;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import org.springframework.stereotype.Controller;
7 import org.springframework.ui.Model;
8 import org.springframework.web.bind.annotation.PathVariable;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RequestParam;
11
12 import com.example.tutorial3.model.StudentModel;
13 import com.example.tutorial3.service.InMemoryStudentService;
14 import com.example.tutorial3.service.StudentService;
15
16 @Controller
17 public class StudentController {
18     private final StudentService studentService;
19
20     public StudentController() {
21         studentService = new InMemoryStudentService();
22     }
23
24     @RequestMapping("/student/add")
25     public String add(@RequestParam(value = "npm", required = true) String npm,
26                     @RequestParam(value = "name", required = true) String name,
27                     @RequestParam(value = "gpa", required = true) double gpa ) {
28         StudentModel student = new StudentModel(name, npm, gpa);
29         studentService.addStudent(student);
30         return "add";
31     }
32
33 }
34

```

Membuat File HTML untuk menampilkan bahwa proses add berhasil.

Nama htmlnya add.html karena pada controller kita sudah return "add"; yang artinya nanti akan dicari template dengan nama add.html

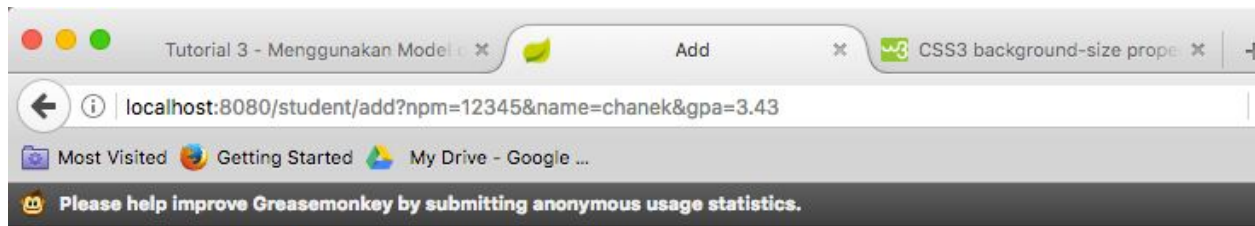
```

1 <html>
2 <head>
3     <title> Add </title>
4 </head>
5 <body>
6     <h2> Data berhasil ditambahkan </h2>
7 </body>
8 </html>
9

```

PERTANYAAN 1: localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43
apakah hasilnya? Jika error, tuliskan penjelasan Anda

Ternyata tidak error, data berhasil ditambahkan seperti ekspektasi method ini sesungguhnya.

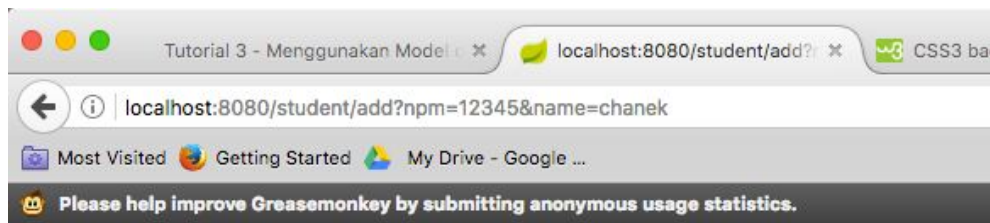


Data berhasil ditambahkan

PERTANYAAN 2: localhost:8080/student/add?npm=12345&name=chanek
apakah hasilnya? Jika error, tuliskan penjelasan Anda

Hasilnya error, hal itu dikarenakan kita sudah mendefinisikan di dalam parameter method tersebut bahwa semua parameter required = true. Pada kali ini kita tidak mengirimkan parameter gpa yang menyebabkan error.

```
25 public String add(@RequestParam(value = "npm", required = true )String npm,  
26                  @RequestParam(value = "name", required = true ) String name,  
27                  @RequestParam(value = "gpa", required = true ) double gpa ) {
```



Whitelabel Error Page

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

Fri Sep 22 15:53:39 WIB 2017

There was an unexpected error (type=Bad Request, status=400).

Required double parameter 'gpa' is not present

Sekarang kita lanjutkan,

Implementasi method untuk dapat menampilkan data mahasiswa berdasarkan parameter npm yang dikirimkan


```
36 @RequestMapping("/student/view")
37 public String view(Model model, @RequestParam(value = "npm", required = true) String npm) {
38
39     StudentModel student = studentService.selectStudent(npm);
40     model.addAttribute("student", student);
41     return "view";
42 }
```

Membuat file html view.html untuk menampilkan data student yang sebelumnya dikirimkan dari controller jika ditemukan.

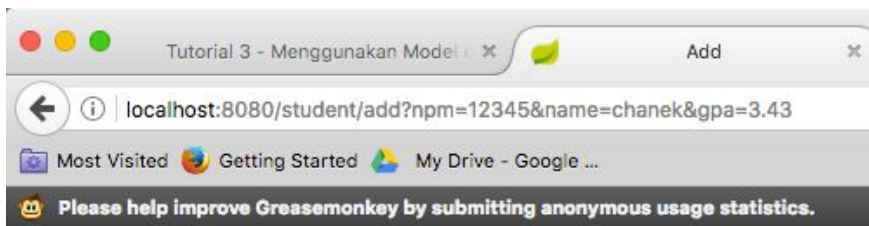
```
InMemoryStudent StudentControll add.html view.html StudentModel.java 8
1 <!DOCTYPE html>
2 <html xmlns:th= "http://www.thymeleaf.org">
3 <head>
4 <title>View Student by NPM </title>
5 </head>
6 <body>
7     <h3 th:text= "'NPM = ' + ${student.npm}"> Student NPM </h3>
8     <h3 th:text= "'Name = ' + ${student.name}"> Student Name </h3>
9     <h3 th:text= "'GPA = ' + ${student.gpa}"> Student GPA </h3>
10 </body>
11 </html>
```

PERTANYAAN 3: localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43

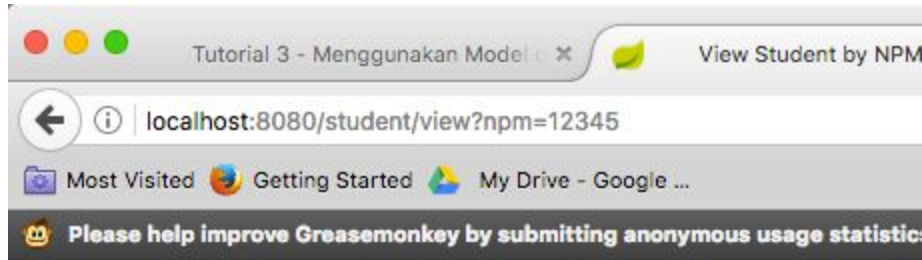
lalu buka localhost:8080/student/view?npm=12345

apakah data Student tersebut muncul? Jika tidak, mengapa?

Setelah di add, lalu di view dengan url tersebut, ternyata data student muncul, tidak ada error, sesuai ekspektasi



Data berhasil ditambahkan



NPM = 12345

Name = chanek

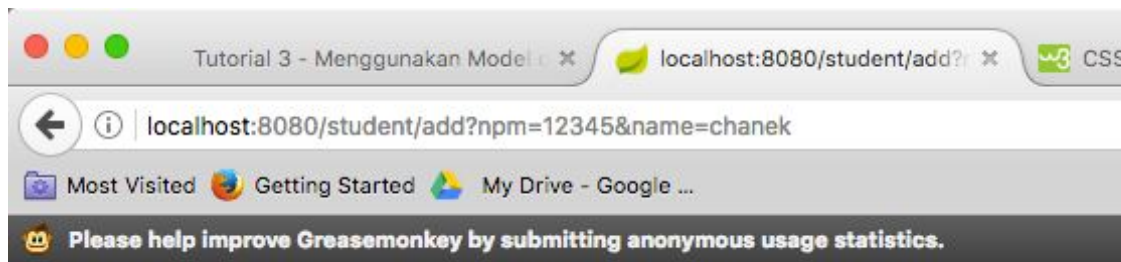
GPA = 3.43

PERTANYAAN 4: Coba matikan program dan jalankan kembali serta buka

localhost:8080/student/view?npm=12345

apakah data Student tersebut muncul? Jika tidak, mengapa?

Ternyata sekarang malah jadi error. Ini terjadi karena data yang sebelumnya sudah di add ke dalam list menjadi hilang (karena tadi dimatikan programnya). Ini karena kita belum benar-benar implementasi koneksi database, sesuai nama servicenya, kita masih menggunakan inMemory. Jadi, ketika mesin dimatikan, sesuai ekspektasi data hilang kembali dan menjadi error di halaman ini (karena kita tidak handle jika npm tidak ditemukan).



Whitelabel Error Page

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

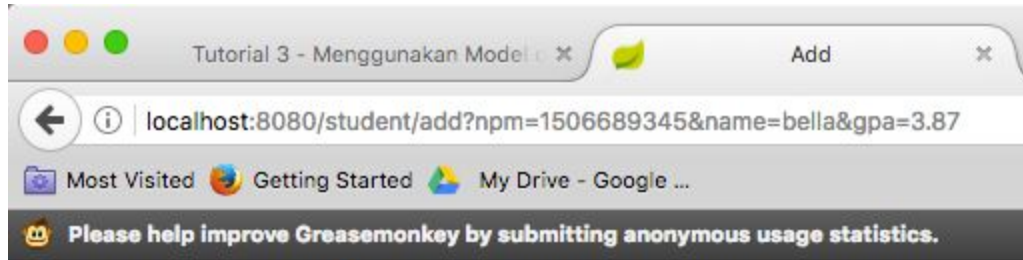
Fri Sep 22 15:53:39 WIB 2017

There was an unexpected error (type=Bad Request, status=400).

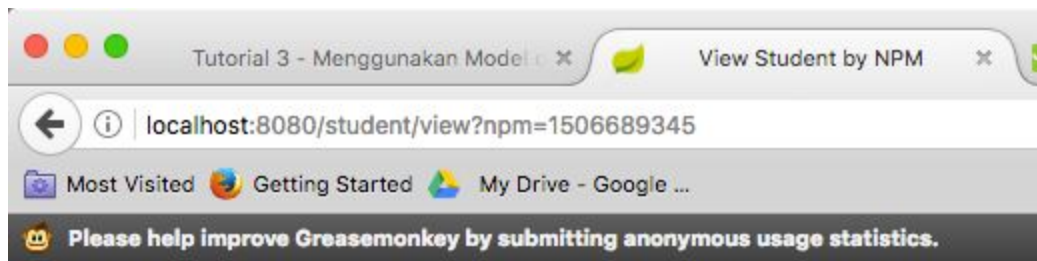
Required double parameter 'gpa' is not present

PERTANYAAN 5: Coba tambahkan data Student lainnya dengan NPM yang berbeda

Sesuai ekspektasi, setelah di add student bernama bella dengan npm 1506689345, saat di view, data nya muncul.



Data berhasil ditambahkan



NPM = 1506689345

Name = bella

GPA = 3.87

Sekarang kita lanjutkan lagi,

Membuat method viewAll() untuk tampilkan seluruh data yang ada

```
44 @RequestMapping("/student/viewall")
45 public String viewAll(Model model) {
46     List<StudentModel> students = studentService.selectAllStudents();
47     model.addAttribute("students", students);
48     return "viewall";
49 }
```

Membuat file html untuk menerima response yang dikirimkan dari controller untuk viewAll.

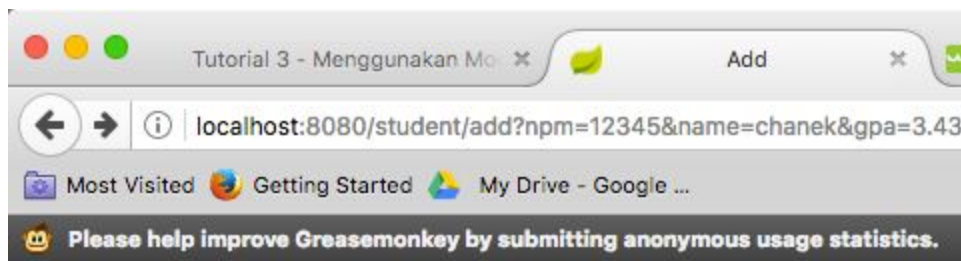

```
InMemoryStudent StudentControll add.html view.html viewall.html » 8
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>View All Students </title>
5 </head>
6 <body>
7 <div th:each="student, iterationStatus: ${students}">
8 <h3 th:text="No. ' + ${iterationStatus.count}">No. 1</h3>
9 <h3 th:text="'NPM. ' + ${student.npm}">Student NPM</h3>
10 <h3 th:text="'Name. ' + ${student.name}">Student Name</h3>
11 <h3 th:text="'GPA. ' + ${student.gpa}"> Student GPA</h3>
12 <hr/>
13 </div>
14 </body>
15 </html>
```

PERTANYAAN 5: localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43

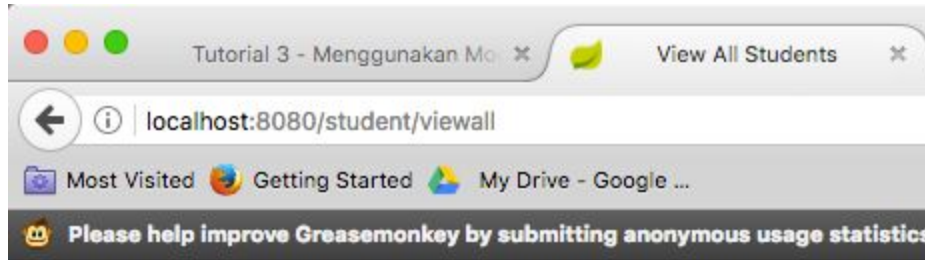
lalu buka localhost:8080/student/viewall

apakah data Student tersebut muncul?

Ternyata, setelah data di add dengan url tersebut, dan kemudian dibuka viewAll, datanya muncul sesuai ekspektasi kita



Data berhasil ditambahkan



No. 1

NPM. 12345

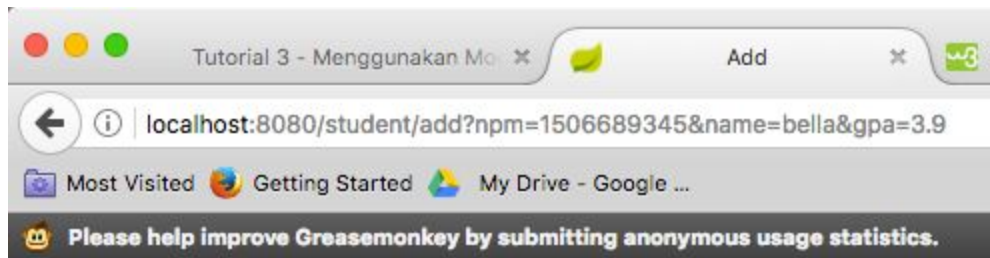
Name. chanek

GPA. 3.43

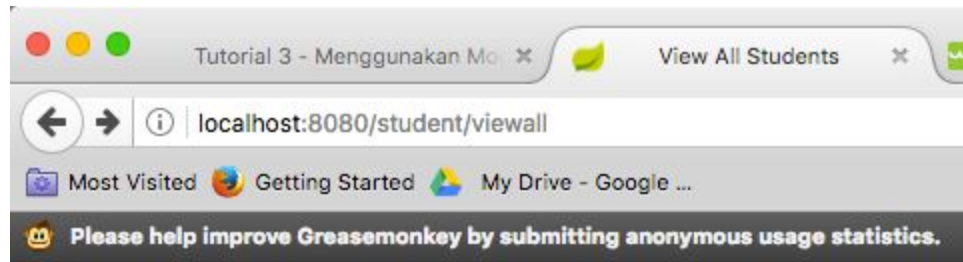
PERTANYAAN 6: Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka localhost:8080/student/viewall,

Apakah semua data Student muncul?

Sekarang kita input student dengan nama bella. Ternyata jadi muncul dibawah datanya chanek.



Data berhasil ditambahkan



No. 1

NPM. 12345

Name. chanek

GPA. 3.43

No. 2

NPM. 1506689345

Name. bella

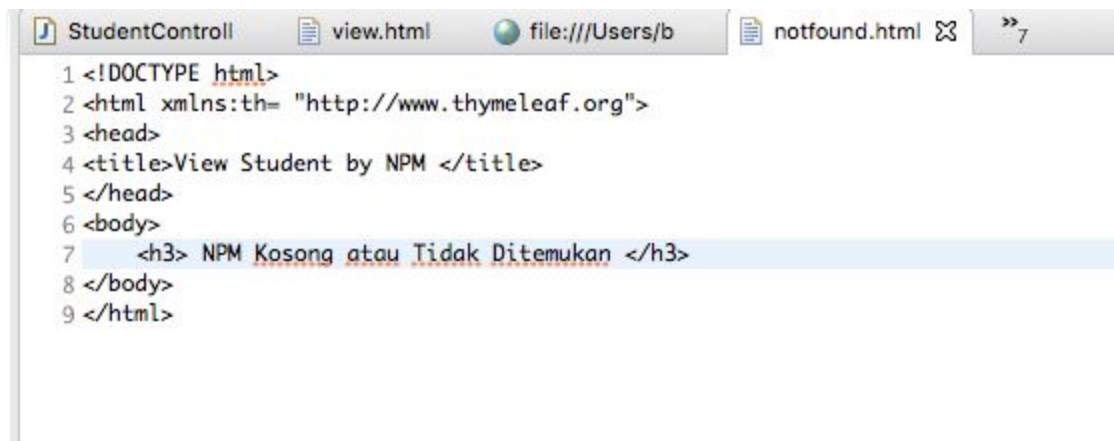
GPA. 3.9

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`.

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

Pertama kita buat file `html` bernama "`notfound.html`" untuk handle kasus dimana `npm` tidak dikirimkan maupun tidak ditemukan di data list kita.

A screenshot of a code editor window. The title bar shows three tabs: 'StudentControll', 'view.html', and 'notfound.html'. The 'notfound.html' tab is active. The code in the editor is as follows:

```
1 <!DOCTYPE html>
2 <html xmlns:th= "http://www.thymeleaf.org">
3 <head>
4 <title>View Student by NPM </title>
5 </head>
6 <body>
7   <h3> NPM Kosong atau Tidak Ditemukan </h3>
8 </body>
9 </html>
```

Berikutnya kita implementasi method `view` dengan `PathVariable` di `Controller`. Kita akan gunakan `Optional` untuk membuat variable `npm` menjadi tidak wajib dikirimkan, serta membuat 2 request mapping berbeda untuk handle dimana `npm` tidak dikirimkan (berdasarkan tutorial 2).

Kita juga akan cek jika `npm` present di variable yang dikirimkan, maka akan diproses dan jika ditemukan, kita akan tampilkan di `view.html` data studentnya. Selain daripada itu (`npm` tidak dikirim, atau tidak ditemukan saat coba `selectStudent`), maka lempar ke `notfound.html`.

Kita akan memanfaatkan `view.html` yang sebelumnya saat tutorial sudah kita buat.

```

51 @RequestMapping(value= {"{/student/view", "/student/view/{npm}"})
52 public String viewPathVariable(Model model, @PathVariable Optional<String> npm) {
53     StudentModel student = null;
54
55     if(npm.isPresent()) {
56         student = studentService.selectStudent(npm.get());
57         if(student == null) {
58             return "notfound";
59         }
60     } else {
61         return "notfound";
62     }
63     model.addAttribute("student", student);
64     return "view";
65 }
66 }
67

```

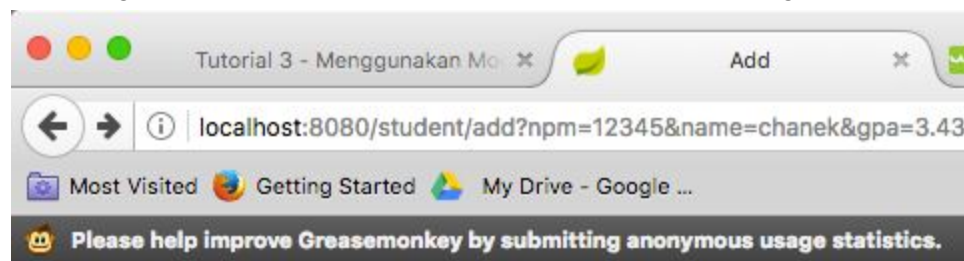
Ternyata hal ini membuat request mapping bentrok dengan /student/view yang sebelumnya sudah kita buat saat tutorial, kita comment dulu code yang tadi :D.

```

35 // TODO: di comment karena di soal latihan juga perlu request mapping /student/view
36 // @RequestMapping("/student/view")
37 // public String view(Model model, @RequestParam(value = "npm", required = true) String npm) {
38 //
39 //     StudentModel student = studentService.selectStudent(npm);
40 //     model.addAttribute("student", student);
41 //     return "view";
42 // }
43

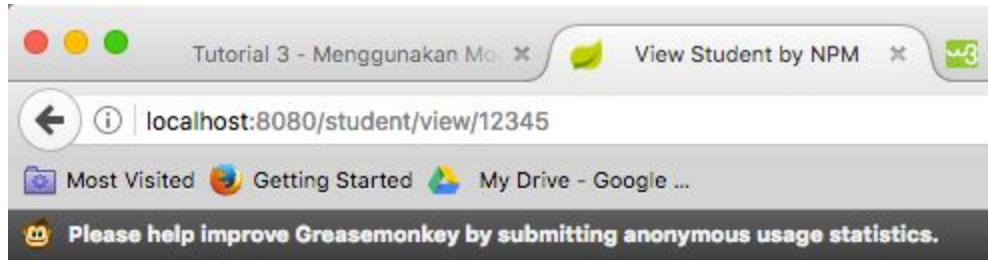
```

Sekarang kita coba memasukkan data Chanek kembali dengan npm 12345



Data berhasil ditambahkan

Lalu kita coba buka /student/view/12345 untuk test hasil kerja kita. Dan ternyata berhasil!! Data berhasil ditampilkan.

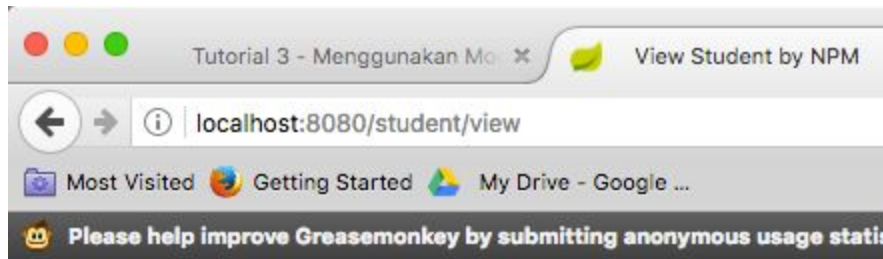


NPM = 12345

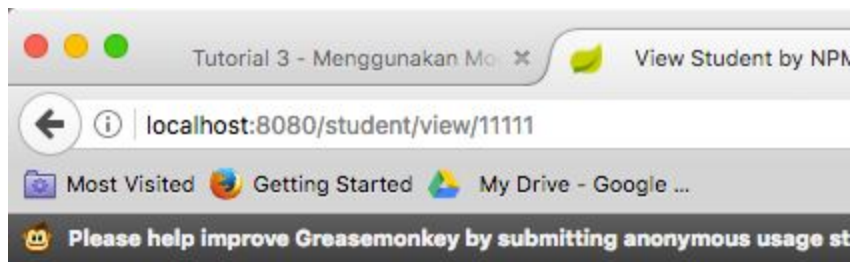
Name = chanek

GPA = 3.43

Kita coba kasus kasus untuk jika npm tidak ditemukan, maupun tidak dikirimkan.



NPM Kosong atau Tidak Ditemukan



NPM Kosong atau Tidak Ditemukan

Dengan begini, soal nomor 1 sudah terpenuhi semuanya.

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.

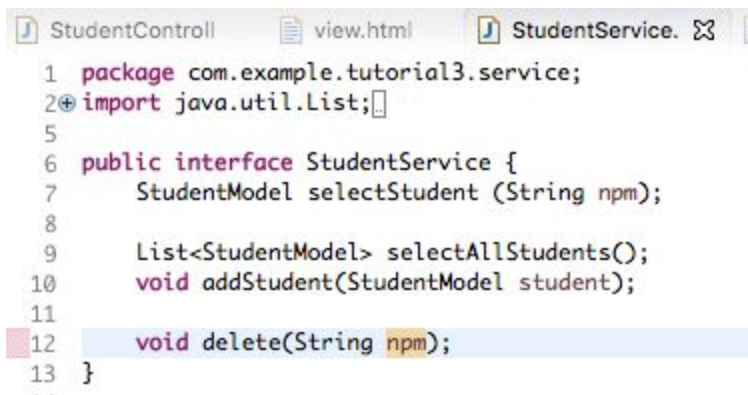
Kita akan memanfaatkan notfound.html yang sebelumnya untuk menampilkan pesan error jika npm tidak dikirimkan atau ternyata tidak ditemukan dalam data kita.

Sekarang kita hanya perlu buat html untuk menampilkan jika delete berhasil, bernama deletesuccess.html. Html ini akan menerima parameter npm yang dikirimkan controller untuk tampilkan pesan sukses.



```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>View Student by NPM </title>
5 </head>
6 <body>
7 <h3 th:text="'Proses Delete terhadap data NPM ' + ${npm} + ' Berhasil Dilakukan'"> Proses
8 </body>
9 </html>
```

Karena service yang kita buat saat tutorial tidak memungkinkan kita bisa melakukan delete(String npm), maka kita perlu buat dulu di interface service dan implementasinya.



```
1 package com.example.tutorial3.service;
2 import java.util.List;
3
4
5
6 public interface StudentService {
7     StudentModel selectStudent (String npm);
8
9     List<StudentModel> selectAllStudents();
10    void addStudent(StudentModel student);
11
12    void delete(String npm);
13 }
```

Saat kita akan delete, seperti halnya dengan get, kita cari dulu yang npmnya sesuai, lalu pakai method yang sudah disediakan List oleh java yaitu remove() untuk me remove student tersebut.

```

33
34 @Override
35 public void delete(String npm) {
36     for(int i = 0 ; i < studentList.size(); i++) {
37         if(studentList.get(i).getNpm().equals(npm)) {
38             studentList.remove(i);
39         }
40     }
41
42
43 }

```

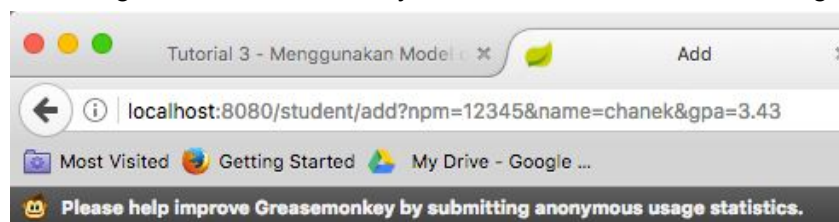
Setelah semua selesai baru kita buat method di controller dengan PathVariable sebagai media pengiriman parameter. Perhatikan methodnya mirip seperti nomor 1, kita buat 2 kemungkinan request mapping, dan kita cek apakah npm dikirimkan atau tidak. Selain itu juga kita perlu cek apakah ada student dengan npm tersebut dengan cara panggil selectStudent(String npm) sesuai dengan npm yang dikirimkan. Jika semuanya aman, barulah kita akan lakukan proses delete() serta tampilkan pesan berhasil. Jika tidak, maka kita perlu tampilkan notfound.html seperti biasa.

```

69 @RequestMapping(value= {"/student/delete", "/student/delete/{npm}"})
70 public String delete(Model model, @PathVariable Optional<String> npm) {
71     StudentModel student = null;
72
73     if(npm.isPresent()) {
74         student = studentService.selectStudent(npm.get());
75         if(student == null) {
76             return "notfound";
77         }
78         else {
79             studentService.delete(npm.get());
80             model.addAttribute("npm", npm.get());
81             return "deletesuccess";
82         }
83     } else {
84         return "notfound";
85     }
86 }
87
88
89

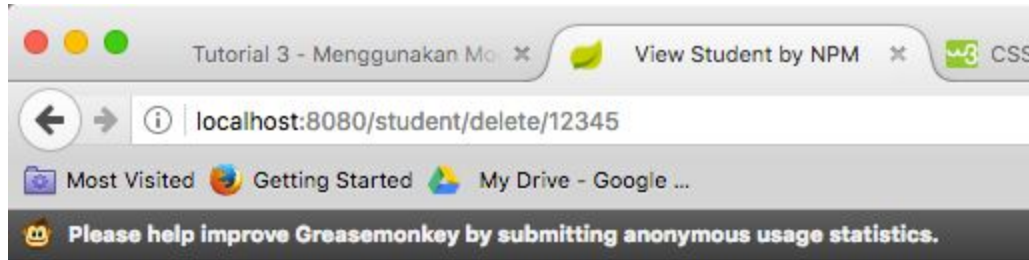
```

Sekarang kita akan test hasilnya. Kita coba insert Chanek dengan npm 12345.



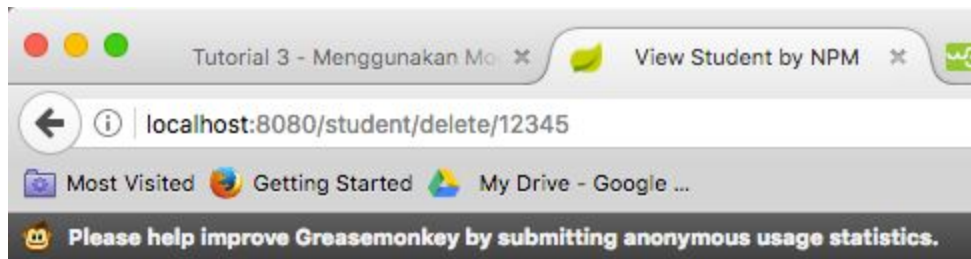
Data berhasil ditambahkan

Sekarang kita coba delete si npm 12345, dan ternyata berhasil!



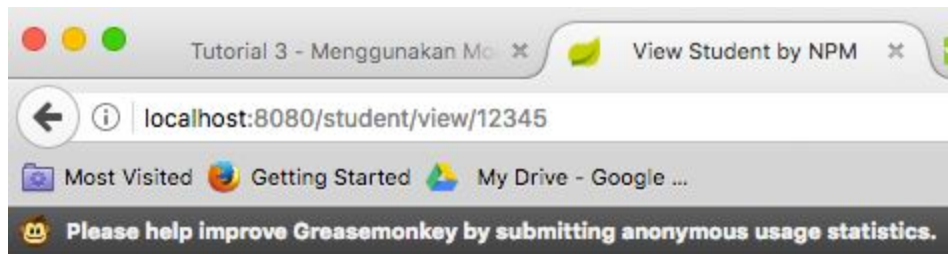
Proses Delete terhadap data NPM 12345 Berhasil Dilakukan

Sekarang coba lagi delete ulang 12345, ternyata hasilnya jadi gagal. Ini karena npm 12345 sekarang sudah tidak ditemukan (karena tadi sudah di delete)



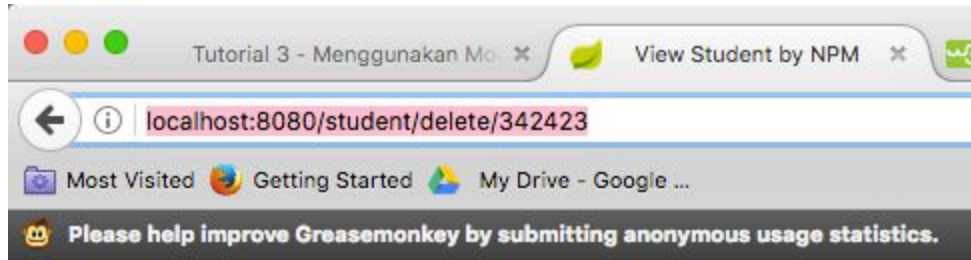
NPM Kosong atau Tidak Ditemukan

Buktinya saat kita coba view, datanya benar-benar tidak ditemukan

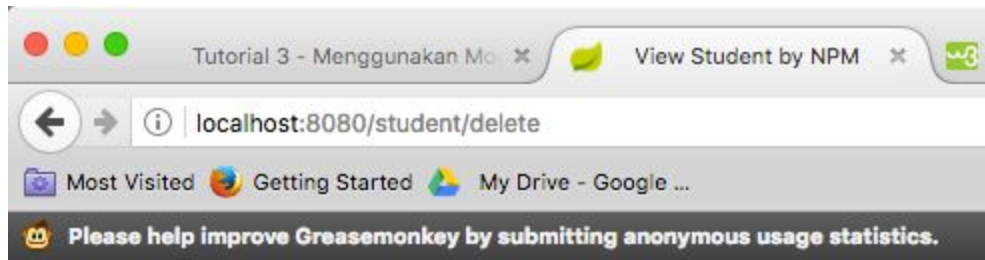


NPM Kosong atau Tidak Ditemukan

Sekarang kita test dengan kasus-kasus npm tidak dikirim maupun tidak terdaftar.



NPM Kosong atau Tidak Ditemukan



NPM Kosong atau Tidak Ditemukan

Semuanya sesuai ekspektasi, maka latihan nomor 2 selesai!!

Lesson Learned

Yang sudah saya pelajari dari tutorial 3 kali ini adalah:

- Saya menjadi mengerti bahwa didunia ini ada framework berbasis java yang juga sama-sama MVC seperti Laravel pada PHP yang pernah kita gunakan saat kuliah PPW. Saya juga mengerti dunia kerja juga bisa saja kita akan berhadapan dengan code semacam ini (ada model, ada service, ada controller, dll).
- Saya menjadi mengerti konsep MVC yang selama ini hanya diomong-omongi oleh orang-orang saja, kurang lebihnya adalah (note: definisi dibawah hanya berdasarkan pemahaman saya saja selama menjalankan tutorial, mohon maklum)
 - Model: sejenis struktur data untuk menampung data yang akan kita simpan.
 - View: template html yang untuk tampilan websitenya, biasanya view akan di panggil melalui controller.
 - Controller: tempat semua logic berasal, yang menjadi otak dari sistem kita saat suatu url (request mapping) dipanggil.
 - Service: logic-logik utama manipulasi data model kita.
- Saya juga jadi lebih paham kurang lebih bagaimana membuat program/sistem/web berbasis java menggunakan framework spring. Lebih mengerti daripada tutorial 2 sebelumnya karena disini sudah diajarkan konsep MVC nya.
- Saya jadi merasa konsep Object Oriented Programming saat kuliah DDP cukup penting dan dibutuhkan.