

a. Hasil ringkasan materi dalam tutorial kali ini

Pada tutorial kali ini membahas mengenai beberapa hal, diantaranya ada *service*. *Service* merupakan interface yang mendefinisikan *method* apa saja yang dapat dilakukan untuk melakukan manipulasi data *student*. Selain itu, pada tutorial ini juga dipelajari mengenai pembuatan sebuah *Model*. *Model* merupakan sebuah objek yang merepresentasikan dan menyimpan informasi terhadap suatu hal. Model yang dibuat dalam tutorial ini adalah objek mahasiswa yang mana atribut yang disimpan didalam objek mahasiswa tersebut adalah npm, nama, dan gpa.

b. Jawaban dari setiap poin pertanyaan beserta screenshot

❖ Membuat *class* model

Membuat *class* StudentModel beserta *setter* dan *getter*

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

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

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

    public String getNpm() {
        return npm;
    }

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

    public String getName() {
        return name;
    }

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

    public double getGpa() {
        return gpa;
    }

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

❖ Membuat service

Membuat interface StudentService.java

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

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

import com.example.tutorial3.model.StudentModel;

public interface StudentService {
    StudentModel selectStudent(String npm);

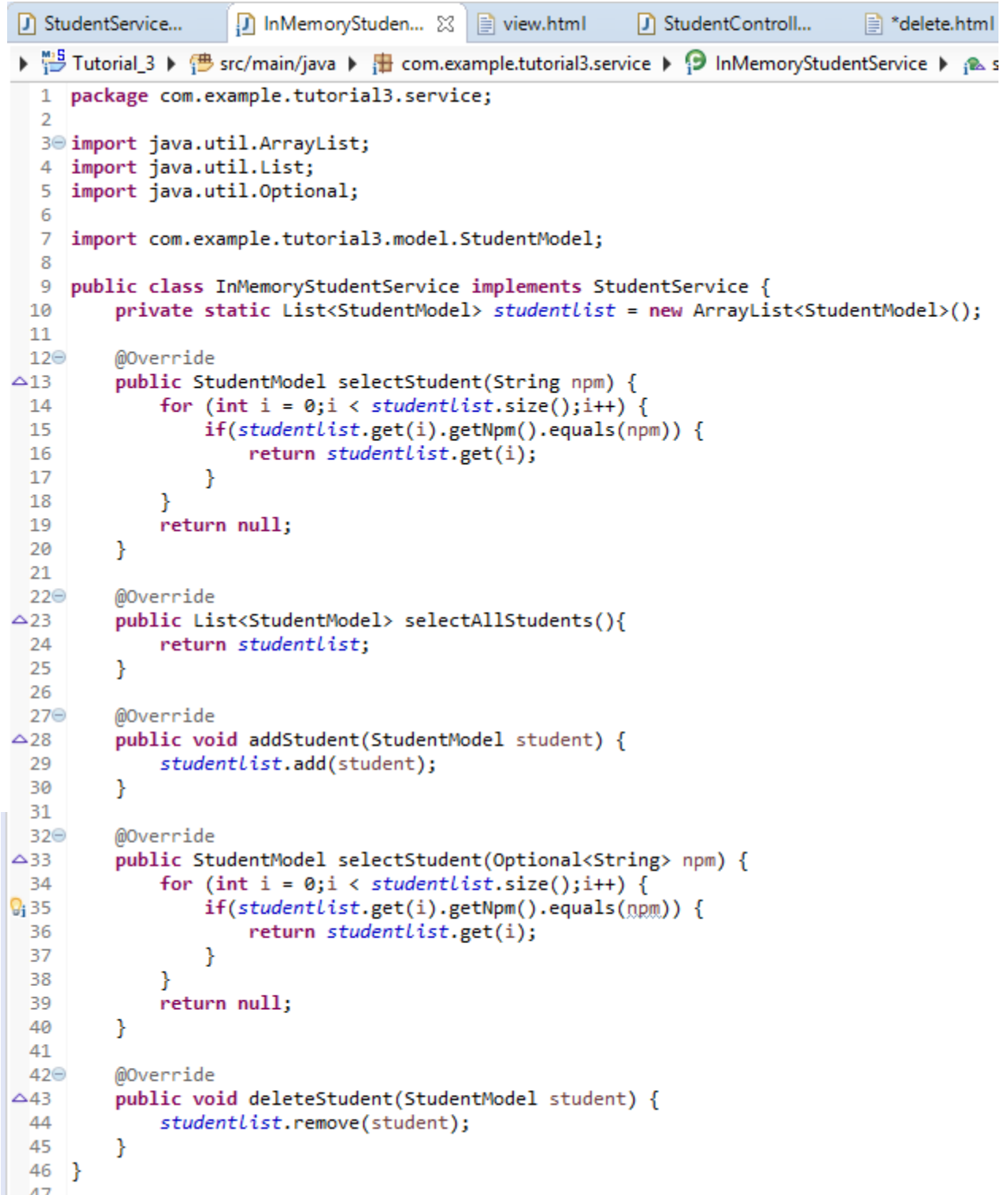
    StudentModel selectStudent(Optional<String> npm);

    List<StudentModel> selectAllStudents();

    void addStudent(StudentModel student);

    void deleteStudent(StudentModel student);
}
```

Membuat InMemoryStudentService untuk mengimplement StudentService



The screenshot shows an IDE window with the following tabs: StudentService..., InMemoryStuden..., view.html, StudentControll..., and *delete.html. The breadcrumb navigation shows the path: Tutorial_3 > src/main/java > com.example.tutorial3.service > InMemoryStudentService. The code is as follows:

```
1 package com.example.tutorial3.service;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Optional;
6
7 import com.example.tutorial3.model.StudentModel;
8
9 public class InMemoryStudentService implements StudentService {
10     private static List<StudentModel> studentlist = new ArrayList<StudentModel>();
11
12     @Override
13     public StudentModel selectStudent(String npm) {
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         return studentlist;
25     }
26
27     @Override
28     public void addStudent(StudentModel student) {
29         studentlist.add(student);
30     }
31
32     @Override
33     public StudentModel selectStudent(Optional<String> npm) {
34         for (int i = 0; i < studentlist.size(); i++) {
35             if (studentlist.get(i).getNpm().equals(npm)) {
36                 return studentlist.get(i);
37             }
38         }
39         return null;
40     }
41
42     @Override
43     public void deleteStudent(StudentModel student) {
44         studentlist.remove(student);
45     }
46 }
47
```

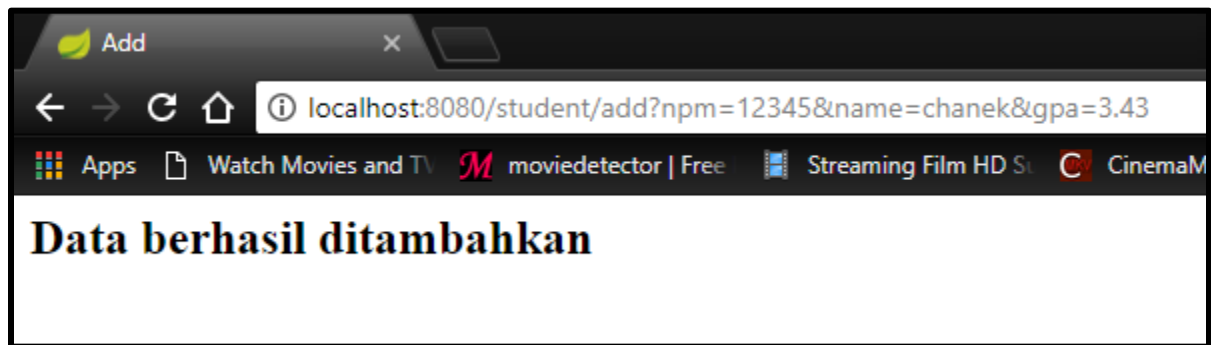
- ❖ Membuat *controller* dengan fungsi *add*

```
@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";
}
```

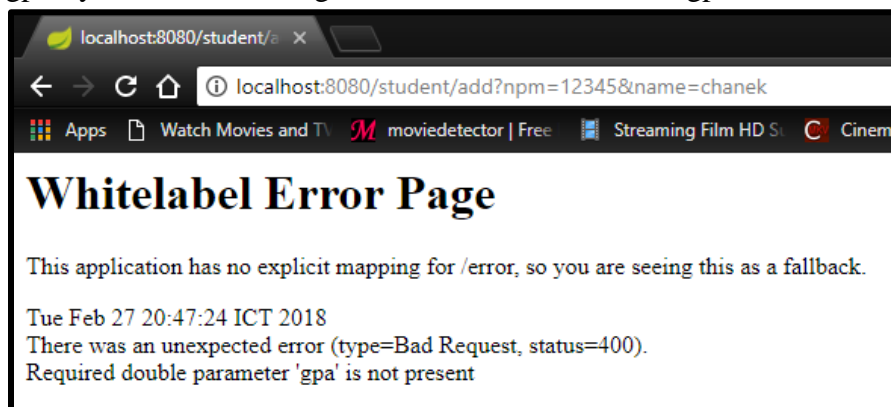
Membuat halaman add.html

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

Pertanyaan 1 : apakah hasilnya? Jika *error* , tuliskan penjelasan Anda
Hasilnya seperti yang ditampilkan pada gambar dibawah ini.



Pertanyaan 2: apakah hasilnya? Jika *error* , tuliskan penjelasan Anda.
Hasilnya *error* seperti yang ditampilkan gambar dibawah ini, hal ini terjadi karena inputan untuk gpa-nya tidak ada sedangkan di-*constructor* variabel gpa didefinisikan.



❖ Method view by NPM

Membuat method view pada controller

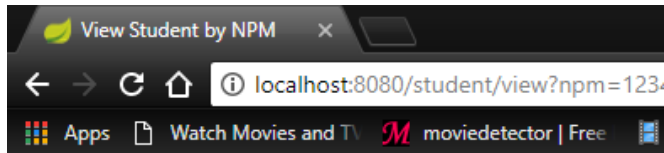
```
@RequestMapping("/student/view")
public String view(Model model, @RequestParam(value = "npm", required = true) String npm) {
    StudentModel student = studentService.selectStudent(npm);
    model.addAttribute("student", student);
    return "view";
}
```

Membuat halaman 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>
```

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

Iya, data yang baru ditambahkan muncul seperti yang ditampilkan pada gambar dibawah ini



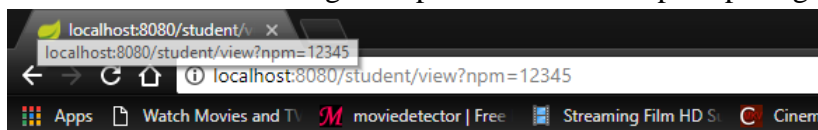
NPM = 12345

Name = chanek

GPA = 3.43

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

Tidak, data tersebut tidak muncul karena di URL belum dimasukan data siswa sehingga isi dari *studentList* masih kosong. Tampilan di browser seperti pada gambar dibawah ini.



Whitelabel Error Page

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

Wed Feb 28 13:08:29 ICT 2018

There was an unexpected error (type=Internal Server Error, status=500).

Exception evaluating SpringEL expression: "student.npm" (view:7)

❖ Method view all

Method viewall pada controller

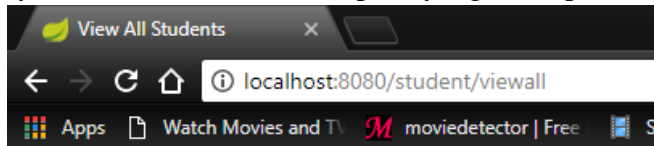
```
@RequestMapping("/student/viewall")
public String viewAll(Model model) {
    List<StudentModel> students = studentService.selectAllStudents();
    model.addAttribute("students", students);
    return "viewall";
}
```

Membuat halaman viewall.html

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

Pertanyaan 5 : apakah data Student tersebut muncul?

Iya, data student muncul seperti yang ditampilkan pada gambar dibawah ini



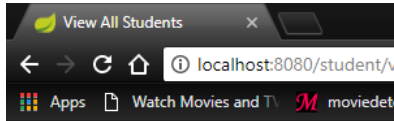
NPM = 12345

Name = chanek

GPA = 3.43

Pertanyaan 6 : Apakah semua data Student muncul?

Iya, semua data student yang ditambahkan muncul seperti pada gambar dibawah ini



NPM = 12345

Name = chaneke

GPA = 3.43

NPM = 12345

Name = chaneke

GPA = 3.43

NPM = 9090

Name = Frans

GPA = 3.98

c. Method selectStudent

Untuk *method selectStudent* terdapat dua buah, pertama untuk *method selectStudent* yang akan digunakan oleh *method view* pada *controller*. Tampilan dari *method selectStudent* tersebut seperti yang ada pada gambar dibawah ini

```
@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;
}
```

for digunakan untuk melakukan perulangan pada *studentList* untuk mengecek isi dari *studentList*. Lalu *statement if, if* berguna untuk melakukan seleksi kondisi jika npm yang diinputkan pada URL ada didalam *studentList* maka npm tersebut akan dikembalikan, jika tidak ada maka akan dikembalikan *null*.

Sedangkan untuk *method studentList* yang kedua digunakan oleh *method view* dengan menggunakan *path variable* pada *controller*. Tampilan *method* tersebut seperti gambar dibawah ini

```

@Override
public StudentModel selectStudent(Optional<String> npm) {
    for (int i = 0; i < studentlist.size(); i++) {
        if (studentlist.get(i).getNpm().equals(npm)) {
            return studentlist.get(i);
        }
    }
    return null;
}

```

Method ini sama seperti method *selectStudent* yang pertama, tetapi bedanya disini menggunakan optional untuk menangkap npm dari *method view* dengan menggunakan *path variable*.

d. Penjelasan fitur *delete*

Method *delete* seperti yang ditampilkan pada gambar dibawah ini

```

@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) {
            model.addAttribute("student", student);
            studentService.deleteStudent(student);
        }
        else {
            model.addAttribute("message", "NPM " + npm.get() + " tidak ditemukan");
            return "fileTidakDitemukan";
        }
    }
    else {
        model.addAttribute("message", "NPM kosong atau tidak diberikan");
        return "fileTidakDitemukan";
    }
    return "delete";
}

```

Method *delete* dibuat menggunakan *path variable* untuk menangkap apakah npm diinputkan atau tidak. Setelah itu, langkah selanjutnya membuat halaman *delete.html* untuk menampilkan halaman apabila proses delete data mahasiswa sukses dilakukan. Halaman *delete.html* seperti pada gambar dibawah ini.

```

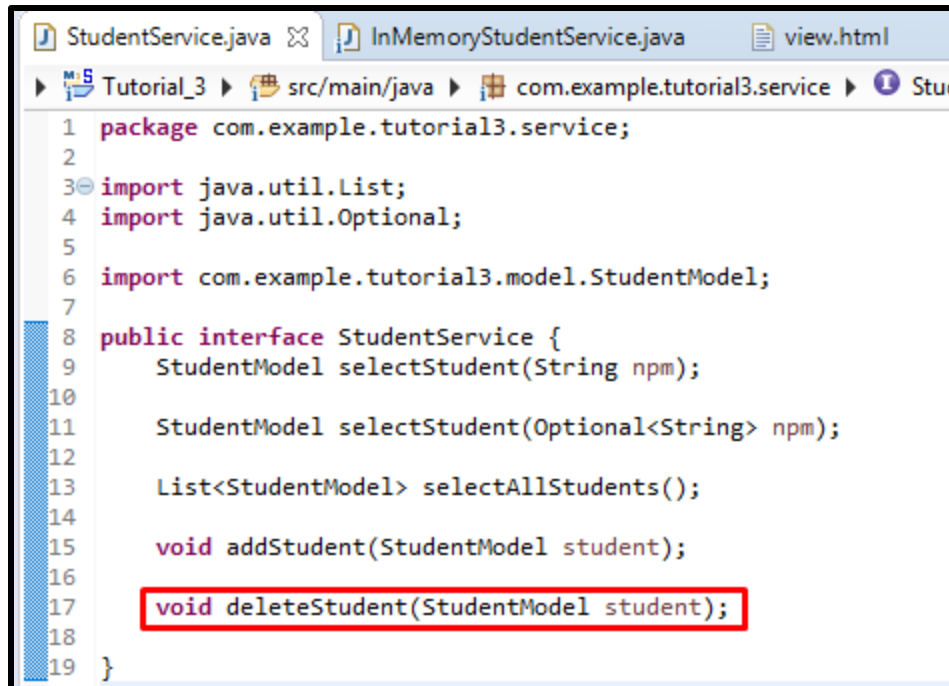
1 <html>
2 <head>
3   <title>Add</title>
4 </head>
5 <body>
6   <h2>Data berhasil dihapus</h2>
7 </body>
8 </html>

```

Selanjutnya membuat *deleteStudent* yang menerima inputan objek student dari controller. Method ini seperti pada gambar dibawah ini.


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

Terakhir mendefinisikan method deleteStudent() pada StudentService.java seperti pada gambar dibawah ini.



```
StudentService.java InMemoryStudentService.java view.html
Tutorial_3 ▸ src/main/java ▸ com.example.tutorial3.service ▸ Stu
1 package com.example.tutorial3.service;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import com.example.tutorial3.model.StudentModel;
7
8 public interface StudentService {
9     StudentModel selectStudent(String npm);
10
11     StudentModel selectStudent(Optional<String> npm);
12
13     List<StudentModel> selectAllStudents();
14
15     void addStudent(StudentModel student);
16
17     void deleteStudent(StudentModel student);
18
19 }
```