

Ringkasan

Pada tutorial 3 kali ini adalah mempelajari cara membuat sebuah *service* dengan Spring boot dengan konsep Model-View-Controller (MVC). *Service* adalah sebuah layer yang menjembatani antara layer *database* dengan *controller*. *Model* digunakan untuk merepresentasikan sebuah objek. *View* digunakan untuk menampilkan hasil presentasi dari sebuah *model*. *Controller* digunakan untuk mengatur *view* dan *model*.

Implementasi *method* `selectStudent`

```
import java.util.Arrays;
import java.util.List;

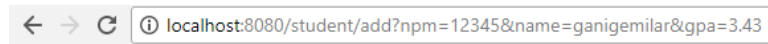
import com.example.tutorial3.model.StudentModel;

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

    @Override
    public StudentModel selectStudent(String npm) {
        // TODO Auto-generated method stub
        for (StudentModel sm : studentList) {
            if (sm.getNpm().equals(npm)) return sm;
        }
        return null;
    }
}
```

Berikut adalah method “`selectStudent(String NPM)`” yang saya implementasikan. Di dalam method tersebut saya menggunakan *foreach* berdasarkan isi dari *studentList*. Masing-masing isinya akan dibandingkan dengan NPM yang dimasukkan pada parameter method tersebut, jika NPM dari salah satu *studentList* sama dengan NPM yang dimasukkan maka akan me-*return* data student, jika tidak maka akan me-*return* null.

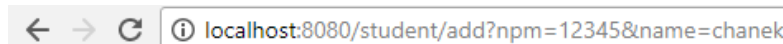
1. Pertanyaan 1



Data berhasil ditambahkan

Tidak terjadi *error* karena URL yang berikan dengan *format* yang benar.

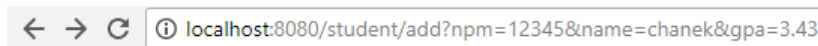
2. Pertanyaan 2



Required double parameter 'gpa' is not present

Terjadi *error* karena URL yang diberikan tidak sesuai *format* atau tidak menyertakan nilai *gpa* pada URL. *Gpa* harus dimasukkan karena pada variabel *gpa* sudah diatur “*required* = true”.

3. Pertanyaan 3



Data berhasil ditambahkan

← → ↻ ⓘ localhost:8080/student/view?npm=12345

NPM = 12345

Name = chanek

GPA = 3.43

Tidak terjadi *error* karena URL yang diberikan dengan *format* yang benar dan NPM yang berikan sesuai atau data dengan NPM 12345 ada datanya.

4. Pertanyaan 4

← → ↻ ⓘ localhost:8080/student/view?npm=12345

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

Terjadi *error* yang disebabkan NPM yang dimasukkan tidak sesuai atau tidak ada datanya (belum dimasukkan data baru), dimana objek *student* yang akan ditampilkan kehalaman bernilai *null*.

← → ↻ ⓘ localhost:8080/student/add?npm=12346&name=ganigemilar&gpa=4.00

Data berhasil ditambahkan

← → ↻ ⓘ localhost:8080/student/view/12346

NPM = 12346

Name = ganigemilar

GPA = 4.0

Data lain berhasil ditambahkan.

5. Pertanyaan 5

← → ↻ ⓘ localhost:8080/student/add?npm=12345&name=chanek&gpa=3.43

Data berhasil ditambahkan

← → ↻ ⓘ localhost:8080/student/viewall

No. 1

NPM = 12345

Name = chanek

GPA = 3.43

Semua data *student* muncul.

6. Pertanyaan 6

← → ↻ ⓘ localhost:8080/student/add?npm=12346&name=ganigemilar&gpa=4.00

Data berhasil ditambahkan

← → ↻ ⓘ localhost:8080/student/viewall

No. 1

NPM = 12345

Name = chanek

GPA = 3.43

No. 2

NPM = 12346

Name = ganigemilar

GPA = 4.0

Semua data *student* muncul termasuk data *student* yang baru ditambahkan.

Latihan

1. Method View

```
@RequestMapping(value = {"/student/view/", "/student/view/{npm}"})
public String view1(
    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("message", String.format("Data student dengan NPM %s tidak ditemukan!", npm.get()));
        return "error";
    }
    model.addAttribute("message", "NPM kosong, data NPM harus dimasukkan!");
    return "error";
}

@RequestMapping(value = {"/student/delete/", "/student/delete/{npm}"})
```

Berikut adalah sebuah *method view* baru yang menggunakan “@PathVariable” pada parameter di *method*-nya. *Method* ini bernama “view1”.

```
@RequestMapping(value = {"/student/view/", "/student/view/{npm}"})
```

Pada “@RequestMapping” disini menggunakan 2 value yaitu “/student/view/” dan “/student/view/{NPM}”, hal ini digunakan ketika URL yang dimasukkan tidak menyertakan NPMnya ataupun menyertakan NPMnya.

```
public String view1(
    Model model,
    @PathVariable Optional<String> npm)
```

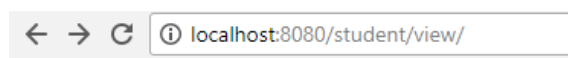
Pada parameter “view1” terdapat “@PathVariable” untuk mengambil NPM yang dimasukkan pada URL. Optional<String> untuk menyimpan value NPMnya.

```

if (npm.isPresent()) {
    StudentModel student = studentService.selectStudent(npm.get());
    if (student != null) {
        model.addAttribute("student", student);
        return "view";
    }
    model.addAttribute("message", String.format("Data student dengan NPM %s tidak ditemukan!", npm.get()));
    return "error";
}
model.addAttribute("message", "NPM kosong, data NPM harus dimasukkan!");
return "error";

```

Pada isi dari method “view1” ini mengolah hasil dari masukkan URL. Saat ketika NPM ada atau dimasukkan maka “NPM.isPresent()” bernilai *TRUE* dan akan mengeksekusi kode dalam *if* tersebut. Student merupakan sebuah objek yang menyimpan hasil *return* dari “studentService.selectStudent(NPM.get())”, lalu ketika student tersebut bernilai tidak null maka method “view1” akan me-*return* halaman “view”, jika null maka akan me-*return* halaman “error”. Jika NPM yang dimasukkan kosong maka akan me-*return* halaman “error”.



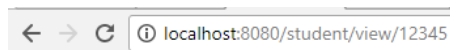
NPM kosong, data NPM harus dimasukkan!

Berikut adalah tes ketika “localhost:8080/student/view/” dijalankan, hal tersebut ketika NPM tidak disertakan. Hal ini menyebabkan method “view1” me-*return* halaman “error” yang berisi pesan diatas.



Data student dengan NPM 12346 tidak ditemukan!

Berikut adalah tes ketika “localhost:8080/student/view/12346” dijalankan. Hal ini menyebabkan *error* karena NPM yang dimasukkan tidak sesuai atau tidak ada datanya.



NPM = 12345

Name = ganigemilar

GPA = 3.43

Berikut adalah tes ketika “localhost:8080/student/view/12345” dijalankan lalu halaman merespon dengan menampilkan sebuah data student yang diminta (asumsikan bahwa NPM 12345 sudah ada di database).

2. **Method Delete**

```

@RequestMapping(value = {"/student/delete/", "/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("student", student);
            return "delete";
        }
        model.addAttribute("message", String.format("Data student dengan NPM %s tidak ditemukan", npm.get()));
        return "error";
    }
    model.addAttribute("message", "NPM kosong, data NPM harus dimasukkan!");
    return "error";
}
}

```

Berikut adalah *method delete* yang digunakan untuk menghapus data *student* berdasarkan NPM.

```

@RequestMapping(value = {"/student/delete/", "/student/delete/{npm}"})

```

Pada “@RequestMapping” disini menggunakan 2 value, dimana salah satunya adalah untuk menangkap NPM yang dimasukkan maupun tidak ada NPM yang dimasukkan pada URL.

```

public String delete(
    Model model,
    @PathVariable Optional<String> npm)

```

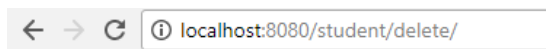
Pada parameter *method delete* ini menggunakan “@PathVariable” untuk mengambil NPM pada *path* yang masukan di URL. Optional<String> untuk menyimpan nilai NPM.

```

if (npm.isPresent()) {
    StudentModel student = studentService.deleteStudent(npm.get());
    if (student != null) {
        model.addAttribute("student", student);
        return "delete";
    }
    model.addAttribute("message", String.format("Data student dengan NPM %s tidak ditemukan", npm.get()));
    return "error";
}
model.addAttribute("message", "NPM kosong, data NPM harus dimasukkan!");
return "error";
}

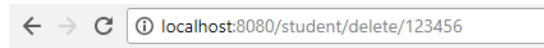
```

Berikut adalah isi kode dari *method delete*. Jika NPM yang dimasukkan ke *path url* tidak kosong, maka “npm.isPresent()” akan bernilai *TRUE*, jika tidak maka akan bernilai *FALSE* dan mengembalikan halaman “error”. Saat “npm.isPresent()” bernilai *TRUE* maka “studentService.deleteStudent(npm.get())” dijalankan untuk menghapus data *student*. *Service* tersebut akan mengembalikan objek *student* yang akan disimpan pada variabel *student* jika NPM yang dimasukkan sesuai atau ada datanya. *Student* bernilai null jika NPM yang dimasukkan tidak sesuai atau tidak ada dan akan mengembalikan halaman “error”. Jika NPM yang dimasukkan ada datanya maka akan mengembalikan halaman “delete” untuk menampilkan data *student* yang dihapus.



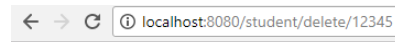
NPM kosong, data NPM harus dimasukkan!

Berikut adalah ketika “localhost:8080/student/delete/” dijalankan, hal ini menyebabkan *error* karena pada *path url* yang diberikan tidak disertakan NPM.



Data student dengan NPM 123456 tidak ditemukan

Berikut adalah ketika "localhost:8080/student/delete/123456" dijalankan, hal ini menyebabkan *error* karena NPM yang dimasukkan tidak sesuai atau tidak ada datanya.



NPM = 12345

Name = ganigemilar

GPA = 3.43

Data student berhasil dihapus

Berikut adalah ketika "localhost:8080/student/delete/12345" dijalankan, hal ini akan mengembalikan halaman yang berisi data *student* yang berhasil dihapus berdasarkan NPM yang diberikan.