Nama : Budi Indrawan
NPM : 1606954754

Tutorial Database & Melakukan Debugging dalam Project Spring Boot

A. Ringkasan Materi

- Pada tutorial4 ini mempelajari yaitu menghubungkan project dengan database MySQL dengan menggunakan library MyBatis dan Lombok. Fungi library Lombok adalah sebagai helper annotation pada project. MyBatis berfungsi melakukan koneksi dan gemerate query dengan helper annotation. Dalam tutorial ini juga mempelajari memberikan argument log pada project yang dibuat. Log tersebut berguna untuk melakukan debuggingpada Spring Boot.

B. Tutorial

❖ LATIHAN MENAMBAHKAN DELETE (Implementasi pada Database)

- Menambahkan link text delete student pada viewall.html

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3     <head>
4         <title>View All Students</title>
5     </head>
6     <body>
7         <h1>All Students</h1>
8 
9     <div th:each="student,iterationStatus: ${students}">
10            <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
11            <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
12            <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
13            <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
14            <a th:href="'/student/delete/' + ${student.npm}"> Delete Data </a><br/>
15            <a th:href="'/student/update/' + ${student.npm}"> Update Data </a><br/>
16            <hr/>
17        </div>
18    </body>
19 </html>
```

- Menambahkan anotasi dan SQL Query untuk menghapus mahasiswa di **studentMapper.java** dengan NPM tertentu.

```
@Delete("DELETE FROM student WHERE npm = #{npm}")
void delStudent (@Param("npm") String npm);
```

- Menambahkan interface delete student pada **studentService.java**

```
void deleteStudent (String npm);
```

- Menambahkan implementasi interface delete student dengan parameter npm pada **studentServiceDatabase.java** dan log.info ditambahkan.

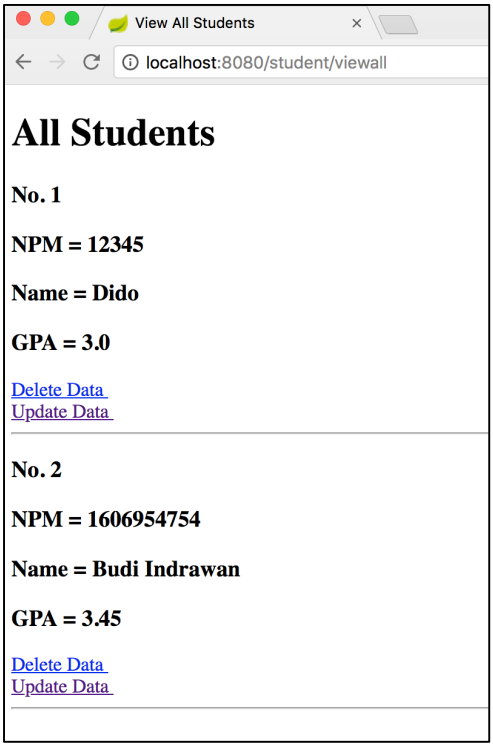
```
@Override
public void deleteStudent (String npm)
{
    log.info("student" + npm + "deleted");
    studentMapper.delStudent(npm);
}
```

Nama : Budi Indrawan
NPM : 1606954754

- Selanjutnya pada **studentController.java** menambahkan method delete untuk melakukan validasi terhadap npm yang akan di delete, jika npm tersebut ada pada database maka akan melakukan method deleteStudent pada **studentServiceDatabase.java** dan kemudian informasil telah berhasil melakukan delete pada **delete.html**.

```
@RequestMapping("/student/delete/{npm}")
public String delete (Model model, @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent (npm);

    if (student != null) {
        studentDAO.deleteStudent (npm);
        return "delete";
    } else {
        model.addAttribute ("npm", npm);
        return "not-found";
    }

}
```

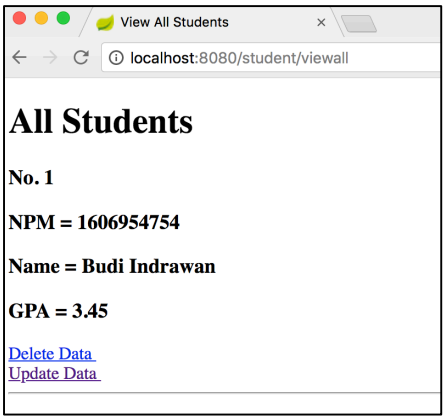
- Tampilan yang ada pada **http://localhost:8080/student/viewall** **sebelum** dilakukan delete data mahasiswa



Nama : Budi Indrawan
NPM : 1606954754

- kemudian memilih link delete data dan hasilnya akan seperti gambar dibawah ini

- Tampilan yang ada pada http://localhost:8080/student/viewall **setelah** dilakukan delete data mahasiswa



❖ **LATIHAN MENAMBAHKAN UPDATE**

- Menambahkan anotasi dan SQL Query untuk memperbaharui mahasiswa di **studentMapper.java** dengan NPM tertentu.

```
@Update("UPDATE student SET name = #{name}, gpa = #{gpa} WHERE npm = #{npm}")
void upStudent (StudentModel student);
```

- Menambahkan interface update student pada **studentService.java**

```
void updateStudent (StudentModel student);
```

- Menambahkan implementasi interface update student dengan parameter studentModel student pada **studentServiceDatabase.java** dan log.info ditambahkan

```
@Override
public void updateStudent(StudentModel student) {
    log.info("Student {} name updated to {}", student.getNpm(), student.getName());
    studentMapper.upStudent(student);
}
```

Nama : Budi Indrawan
NPM : 1606954754

- Menambahkan link text delete student pada viewall.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View All Students</title>
    </head>
    <body>
        <h1>All Students</h1>

    <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>
            <a th:href="'/student/delete/' + ${student.npm}"> Delete Data </a><br/>
            <a th:href="'/student/update/' + ${student.npm}"> Update Data </a><br/>
            <hr/>
        </div>
    </body>
</html>
```

- Selanjutnya pada **studentController.java** menambahkan method update untuk melakukan validasi terhadap npm yang akan di update, jika npm tersebut ada pada database maka akan melakukan method updateStudent pada **studentServiceDatabase.java** dan kemudian data mahasiswa akan di tampilkan pada **form-update.html**

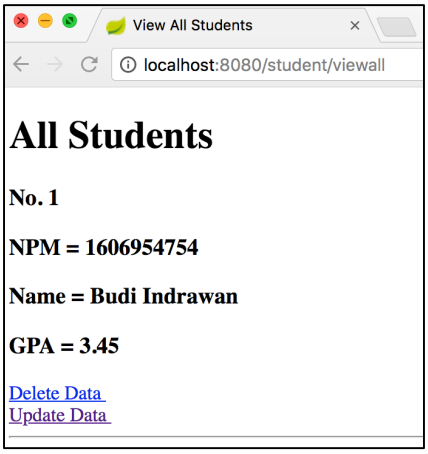
```
@RequestMapping("/student/update/{npm}")
public String update (Model model, @PathVariable(value = "npm") String npm)
{
     StudentModel student = studentDAO.selectStudent(npm);

    if (student != null) {
         model.addAttribute("student", student);
         return "form-update";
    } else {
        model.addAttribute ("npm", npm);
        return "not-found";
    }

}
```

- Ditambahkan juga updateSubmit pada **studentController.java** yang akan menampilkan data student dan akan menyimpan data student yag terbaru dengan menjalankan method updateStudent.

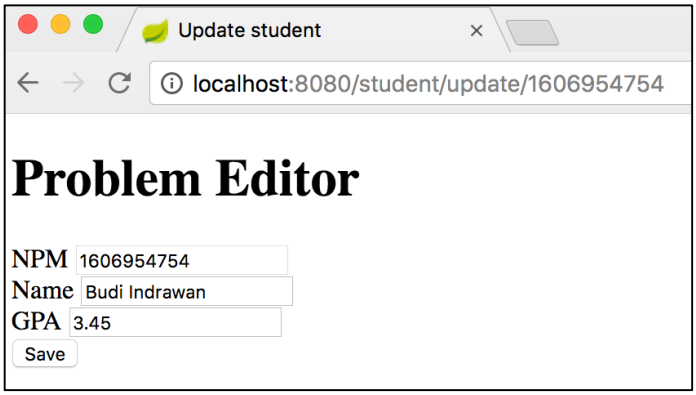
```
@RequestMapping(value="/student/update/submit", method = RequestMethod.POST)
public String updateSubmit(
@RequestParam(value = "npm", required = false) String npm,
@RequestParam(value = "name", required = false) String name,
@RequestParam(value = "gpa", required = false) double gpa) {
     StudentModel student = new StudentModel (npm, name, gpa);
    studentDAO.updateStudent(student);
    return "success-update";
}
```

Nama : Budi Indrawan
NPM : 1606954754

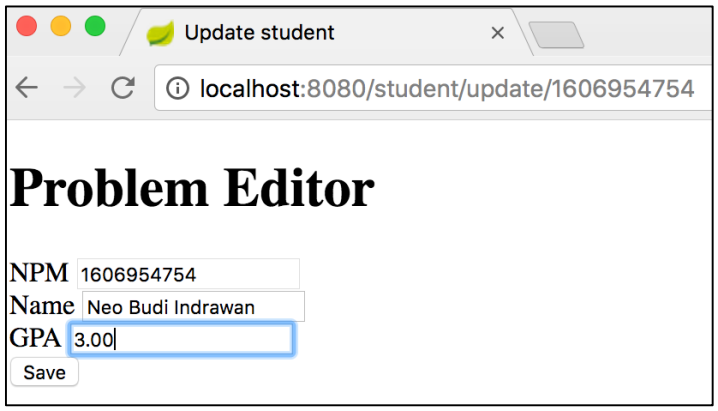
- Tampilan yang ada pada **http://localhost:8080/student/viewall** **sebelum** dilakukan perubahan data mahasiswa



- Setelah di klik link update data pada halaman **http://localhost:8080/student/viewall** akan muncul data mahasiswa yang sudah tersimpan sebelumnya pada database

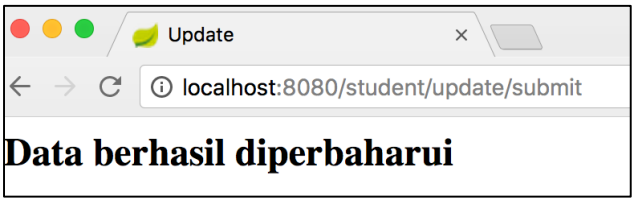


- Gambar dibawah menampilkan saat melakukan perubahan data mahasiswa pada **form-update.html**

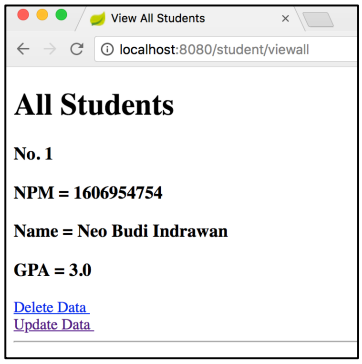


Nama : Budi Indrawan
NPM : 1606954754

- Ketika klik save maka akan muncul gambar dibawah pada **success-update.html**



- Hasil setelah dilakukan perubahan di halaman http://localhost:8080/student/viewall



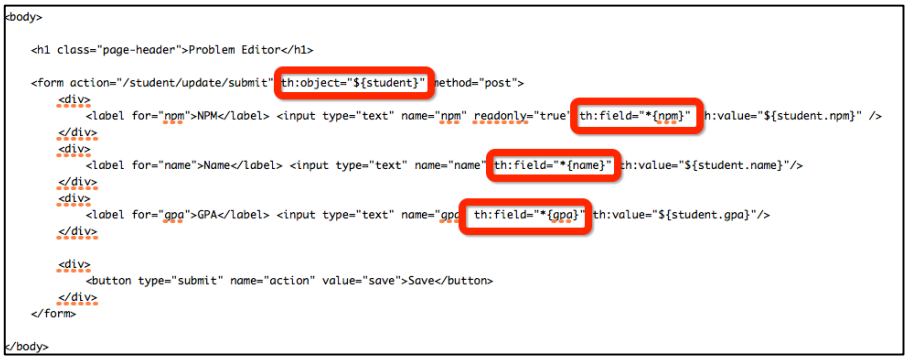
❖ **LATIHAN MENGGUNAKAN OBJECT SEBAGAI PARAMETER**

- Menggunakan model StudentModel untuk menghandle form submit pada **studentController.java**

```
//menggunakan object sebagai parameter
@RequestMapping(value="/student/update/submit", method = RequestMethod.POST)
public String updateSubmit(@ModelAttribute StudentModel student) {
    studentDAO.updateStudent(student);
    return "success-update";
}
```

Nama : Budi Indrawan
NPM : 1606954754

- Sedangkan pada form-update.html menambahkan th:object untuk mendeklarasikan sebuah object yang akan mengumpulkan data mahasiswa. Ada pula th:field digunakan untuk memberikan ekspresi seperti npm, nama & gpa sesuai dengan object student.



Pertanyaan

1. Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required seperti jika menggunakan RequestParam? Apakah validasi diperlukan?

   Jawab :

   Cara yang dapat dilakukan dengan menambahkan standar validation seperti @valid pada parameterobject dan penambahan parameter object bindingResult. Setelah itu harus ditambahkan kondisi pada body [ada form-update menggunakan code “bindingResult.hasErrors()”, yang apabila validasi berhasil akan melnajutkan pada halamat berikutnya namun jika tidak akan kembali ke halaman form-update

   Validasi diperlukan untukmenyimpan data dalam database agar tidak terjadi kesalahan input.

2. Menurut anda, mengapa form submit biasanya menggunakan POST method disbanding GET method? Apakah perlu penanganan berbeda di header atau body method di controller jika form post dikirim menggunakan method berbeda?

   Jawab:

   Dikarenakan POST method lebih aman dalam mengirimkan data, dimana data biasanya terkandung data yang memiliki tingkat sensifitas tinggi misalkan password.

   Perlu dilakukan penanganan yang berbeda dengan menggunakan method RequestMethod.POST pada RequestMapping.

Nama : Budi Indrawan
NPM : 1606954754

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   Jawab:

   Hal itu bisa dilakukan, namun untuk pemanggilan harus satu persatu agar dapat dijalankan, jika tidak akan error.