

Lesson Learned:

Saya dapat memahami bagaimana database pada mysql dapat terkoneksi dengan spring boot. Saya juga dapat memahami bagaimana cara kerja dari form pada spring boot. Saya juga bagaimana cara submit form melalui method POST tanpa mendefinisikan masing-masing atribut dari form satu-satu.

Pertanyaan:

1. Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required? Apakah validasi diperlukan? Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.

Dapat dilakukan dengan menambahkan @PostMapping pada Controller. Tetapi, validasi tidak langsung dilakukan pada form.

(Source: <https://spring.io/guides/gs/validating-form-input/>)

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

Karena dengan menggunakan POST method data tidak ditampilkan secara eksplisit di URL. Dengan menggunakan POST method data juga terlindungi. Diperlukan penanganan yang berbeda di header method.

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

Tidak bisa.

(Source: <https://stackoverflow.com/questions/6918991/submit-post-and-get-variables-in-one-form>)

- **Method yang Anda buat pada Latihan Menambahkan Delete, jelaskan**

```

StudentMapper.java
3+ import java.util.List;
13
14 @Mapper
15 public interface StudentMapper {
16     @Select("select npm, name, gpa from student where npm = #{npm}")
17     StudentModel selectStudent (@Param("npm") String npm);
18
19     @Select("select npm, name, gpa from student")
20     List<StudentModel> selectAllStudents ();
21
22     @Insert("INSERT INTO student (npm, name, gpa) VALUES (#{npm}, #{name}, #{gpa})")
23     void addStudent (StudentModel student);
24
25     @Delete("DELETE from student where npm = #{npm}")
26     void deleteStudent (@Param("npm") String npm);
27

```

- Pertama saya menambahkan method delete student pada StudentMapper. Lalu membuat annotation delete untuk sql databasenya

```

StudentMapper.java StudentServiceDatabase.java
}

@Override
public void addStudent (StudentModel student)
{
    studentMapper.addStudent (student);
}

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

```

- kemudian saya membuat log info dan memanggil method deleteStudent pada kelas StudentServiceDatabase

```

StudentMapper.java StudentServiceDatabase.java StudentController.java
82     List<StudentModel> students = studentDAO.selectAllStudents();
83     model.addAttribute("students", students);
84
85     return "viewall";
86 }
87
88
89 @RequestMapping("/student/delete/{npm}")
90 public String delete (Model model, @PathVariable(value = "npm") String npm)
91 {
92     StudentModel student = studentDAO.selectStudent (npm);
93     if (student != null) {
94         studentDAO.deleteStudent (npm);
95         return "delete";
96     } else {
97         return "not-found";
98     }
99 }
100

```

- Kemudian saya membuat method delete pada StudentController. Pertama saya mencari student yang memiliki npm yang sesuai dengan npm yang dicari (menerima parameter npm sebagai @PathVariable).. Kemudian jika ditemukan, maka student tersebut akan di delete, dan akan ditampilkan view delete. Jika tidak ditemukan akan ditampilkan view not found

- **Method yang Anda buat pada Latihan Menambahkan Update, jelaskan**

```

StudentMapper.java StudentServiceDatabase.java StudentController.java
3 import java.util.List;
13
14 @Mapper
15 public interface StudentMapper {
16     @Select("select npm, name, gpa from student where npm = #{npm}")
17     StudentModel selectStudent (@Param("npm") String npm);
18
19     @Select("select npm, name, gpa from student")
20     List<StudentModel> selectAllStudents ();
21
22     @Insert("INSERT INTO student (npm, name, gpa) VALUES (#{npm}, #{name}, #{gpa})")
23     void addStudent (StudentModel student);
24
25     @Delete("DELETE from student where npm = #{npm}")
26     void deleteStudent (@Param("npm") String npm);
27
28     @Update("UPDATE student SET npm = #{npm}, name = #{name}, gpa = #{gpa} WHERE npm = #{npm}")
29     void updateStudent (StudentModel student);
30 }
31

```

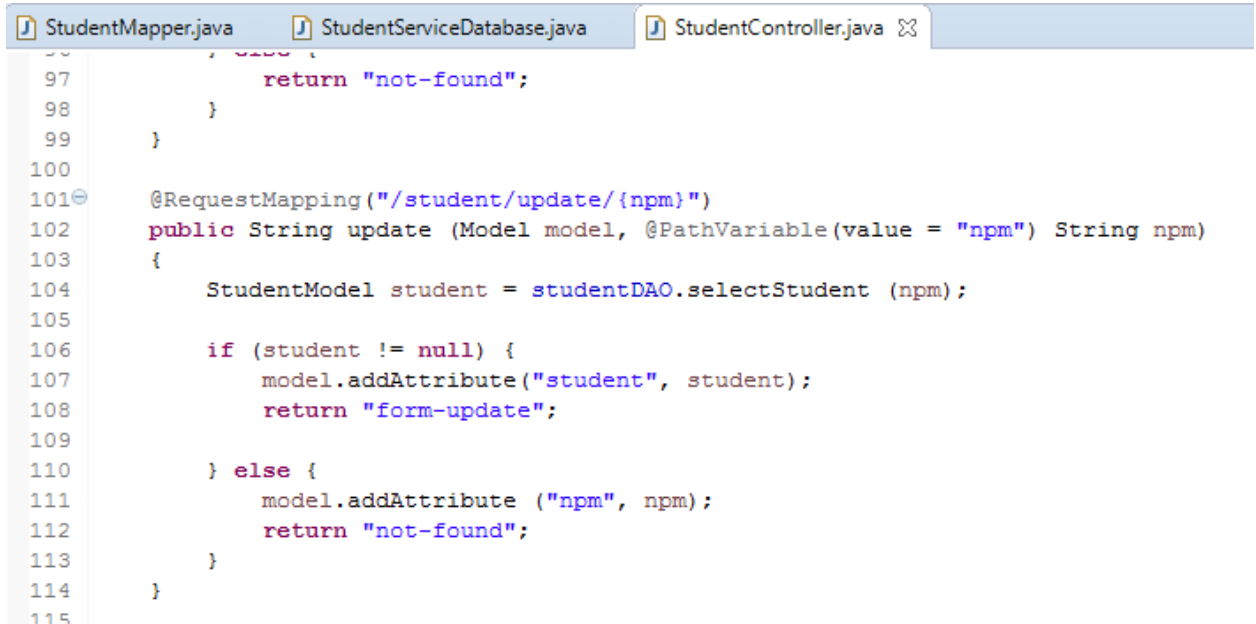
- Pertama saya membuat method update student pada StudentMapper. Kemudian saya membuat annotation update sql diatasnya.

```
StudentMapper.java StudentServiceDatabase.java StudentController.java StudentService.java ✖
1 package com.example.tutorial4.service;
2
3 import java.util.List;
4
5
6
7 public interface StudentService {
8     StudentModel selectStudent (String npm);
9
10
11     List<StudentModel> selectAllStudents ();
12
13
14     void addStudent (StudentModel student);
15
16
17     void deleteStudent (String npm);
18
19     void updateStudent (StudentModel student);
20 }
21
```

- Kemudian saya membuat method updateStudent pada interface StudentService

```
StudentMapper.java StudentServiceDatabase.java ✖ StudentController.java
37 public void addStudent (StudentModel student)
38 {
39     studentMapper.addStudent (student);
40 }
41
42
43 @Override
44 public void deleteStudent (String npm)
45 {
46     studentMapper.deleteStudent (npm);
47     log.info("student " + npm + " deleted");
48 }
49
50 @Override
51 public void updateStudent(StudentModel student) {
52     studentMapper.updateStudent(student);
53     log.info("student " + student.getNpm() + " updated");
54 }
55 }
56
```

- Setelah itu saya membuat method updateStudent pada StudentServiceDatabase untuk membuat log info dan memanggil method updateStudent pada studentMapper



```

97         return "not-found";
98     }
99 }
100
101 @RequestMapping("/student/update/{npm}")
102 public String update (Model model, @PathVariable(value = "npm") String npm)
103 {
104     StudentModel student = studentDAO.selectStudent (npm);
105
106     if (student != null) {
107         model.addAttribute("student", student);
108         return "form-update";
109     } else {
110         model.addAttribute ("npm", npm);
111         return "not-found";
112     }
113 }
114
115

```

- Setelah itu saya membuat method update pada StudentController. Pada method ini saya akan mencari student dengan npm yang sama dengan npm yang dicari (menerima parameter npm sebagai @PathVariable). Setelah itu, jika ditemukan maka akan ditampilkan halaman not found, dan jika ditemukan akan ditampilkan halaman form-update dan akan menambahkan object student sebagai atribut pada model.

- **Method yang Anda buat pada Latihan Menggunakan Object Sebagai Parameter, jelaskan**

```

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

```

Method ini menerima parameter object student dari class StudentModel lalu akan memanggil method updateStudent() pada class studentDAO. Setelah itu akan ditampilkan halaman "success-update"