**Ilyas Fahreza** 1506733371 Kelas B

**Tutorial 4**

**Menambahkan Delete**
Untuk mendelete student, maka pertama kita mengambil student yang akan didelete melalui StudentMapper selectStudent, sistem akan mereturn sesuai dengan value student yang deselect. Pada StudentController, StudentMapper, StudentService, serta StudentServiceDatabase ditambahkan method berikut:

Student Controller
```
@RequestMapping("/student/delete/{npm}")
    public String delete (Model model, @PathVariable(value = "npm") String npm)
    {
        if(studentDAO.selectStudent(npm) == null){

            model.addAttribute ("npm", npm);
            return "not-found";
        }

        studentDAO.deleteStudent (npm);
        return "delete";
    }
```

StudentMapper
```
@Delete("Delete from student where npm = #{npm}")
    void deleteStudent(@Param("npm") String npm);
```

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

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

**Menambahkan Update**
Untuk mengupdate data student, maka pertama kita mengambil student yang akan diupdate melalui StudentMapper selectStudent, sistem akan mereturn sesuai dengan value student yang deselect. Pada StudentController, StudentMapper, StudentService, serta StudentServiceDatabase ditambahkan method berikut:
Student Controller (menambahkan 2 method yaitu update () serta updateSubmit)
```
  @RequestMapping("/student/update/{npm}")
    public String update (Model model, @PathVariable(value = "npm") String npm)
    {
       StudentModel student = studentDAO.selectStudent(npm);

       if(student == null) {
              return "not-found";
       }

       model.addAttribute("student", student);
        return "form-update ";
```

```
    }

    @RequestMapping(value = "/student/update/submit", method = RequestMethod.POST)
    public String updateSubmit (StudentModel student)
    {
       StudentModel updatedStudent = student;

       studentDAO.updateStudent(updatedStudent);

         return "success-update";
    }
```

StudentMapper

```
@Update("UPDATE student set name = #{name}, gpa = #{gpa} where npm = #{npm}")
    void updateStudent(StudentModel student);
```

StudentServiceDatabase

```
@Override
    public void updateStudent (StudentModel student){
        log.info("student " + student.getNpm() + " updated");
        studentMapper.updateStudent(student);
    }
```

StudentService

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

Untuk update juga menambahkan file html baru, yaitu form-update yang berisikan sebagai berikut:

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>

    <title>Add student</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>

<h1 class="page-header">Form Update Student</h1>

<form action="/student/update/submit" th:object="${student}" method="post">
    <div>
        <label for="npm">NPM</label> <input type="text" readonly="true"
th:value="${npm}" th:field="*{npm}" name="npm" />
    </div>
    <div>
        <label for="name">Name</label> <input type="text" th:value="${name}"
th:field="*{name}" name="name" />
    </div>
    <div>
        <label for="gpa">GPA</label> <input type="text" th:value="${gpa}"
th:field="*{gpa}" name="gpa" />
    </div>

    <div>
        <button type="submit" name="action" value="save">Save</button>
```

```
    </div>
</form>

</body>

</html>
```

**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.
   **Jawab: Validasi terhadap input yang diisi dilakukan dengan menambahkan atribut required="true" atau "false" (optional) pada setiap tag <input>. Validasi diperlukan untuk *handle* input *field* yang kosong.**

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 di post dikirim menggunakan method berbeda?
   **Jawab: Dengan menggunakan POST method data akan menjadi lebih aman, karena data yang dikirim tidak tercantum pada path, sehingga mengurangi pencurian data. Perlu.**

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST? Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka localhost:8080/student/viewall, Apakah semua data Student muncul?
   **Jawab: Ya. Kta dapat menambahkan @RequestMapping dengan keduanya menjadi method = {RequestMethod.GET, RequestMethod.POST}**