Muthia Nabila
1406577064
APAP-B

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 melakukannya dengan memberikan banyak perubahan pada body method. Ya, diperlukan validasi.
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?
   Menurut saya, karena POST method lebih cocok untuk kegiatan pengeditan. Menurut w3school, POST dapat mengirimkan lebih banyak informasi daripada GET. Jika ingin menggunakan method berbeda, tentu saja diperlukan penanganan yang berbeda.
3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   Tidak. Karena fungsi dari GET dan POST berkebalikan.

Muthia Nabila
1406577064
APAP-B

- Method yang Anda buat pada Latihan Menambahkan Delete, jelaskan
  Pada StudentMapper, dibuat method deleteStudent yang berguna untuk melakukan delete menggunakan script SQL, yaitu:

```
@Delete("DELETE FROM student where npm=#{npm}")void deleteStudent (String npm);
```

  Pada StudentServiceDatabase, dibuat method deleteStudent yang berguna untuk mengimplementasi fungsi tersebut pada StudentService. Di dalamnya juga terdapat fungsi logging untuk debugging. Isi methodnya, yaitu:

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

  Pada StudentController, dibuat method delete yang jika npm tidak ditemukan akan menampilkan tampilan dari not-found.html dan jika ada akan menampilkan tampilan pada delete.html seperti berikut:

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

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

Muthia Nabila
1406577064
APAP-B

- Method yang Anda buat pada Latihan Menambahkan Update, jelaskan

  Pada StudentMapper, dibuat method updateStudent yang berguna untuk melakukan update menggunakan script SQL, yaitu:

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

  Pada StudentService, dibuat method updateStudent seperti berikut:

```
package com.example.service;

import java.util.List;

public interface StudentService
{
    StudentModel selectStudent (String npm);


    List<StudentModel> selectAllStudents ();


    void addStudent (StudentModel student);


    void deleteStudent (String npm);

    void updateStudent (StudentModel student);
}
```

  Pada StudentServiceDatabase, dibuat method updateStudent yang berguna untuk mengimplementasi fungsi tersebut pada StudentService. Di dalamnya juga terdapat fungsi logging untuk debugging. Isi methodnya, yaitu:

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

  Pada viewall.html, dibuat link untuk melakukan update seperti berikut:

```
<!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>
```

Muthia Nabila
1406577064
APAP-B

Pada form-update.html, ditambahkan method post karena memerlukan post data. Isi file tersebut, yaitu:

```
<body>

    <h1 class="page-header">Problem Editor</h1>

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

        <div>
            <button type="submit" name="action" value="save">Update</button>
        </div>
    </form>

</body>
```

Pada success-update.html, dibuat kalimat yang lebih merepresentasikan keberhasilan proses update. Isinya adalah sebagai berikut:

```
<html>
    <head>
        <title>Update</title>
    </head>
    <body>
        <h2>Data berhasil diubah</h2>
    </body>
</html>
```

Pada StudentController, dibuat method update yang jika npm tidak ditemukan akan menampilkan tampilan dari not-found.html dan jika ada akan menampilkan tampilan pada form-update.html seperti berikut:

Muthia Nabila
1406577064
APAP-B

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

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

Kemudian, masih pada StudentController, dibuat lanjutan setelah form-update selesai diisi. Jika berhasil, maka fungsi method updateSubmit ini akan memberikan tampilan success-update.html seperti berikut:

```
@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";
}
```

Muthia Nabila
1406577064
APAP-B

- Method yang Anda Buat pada Latihan Menggunakan Object sebagai Parameter, jelaskan

  Pada form-update.html dilakukan penambahan th:object dan th:field seperti berikut:

```
<body>

    <h1 class="page-header">Problem Editor</h1>

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

        <div>
            <button type="submit" name="action" value="save">Update</button>
        </div>
    </form>

</body>
```

  Pada StudentController, method updateSubmit diubah menjadi hanya memiliki masukan dari StudentModel student supaya barisnya lebih singkat. Method tersebut seperti berikut:

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