Nama: Muhammad Hasby Rosyadi
NPM: 1306386642

1. Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukanvalidasi 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:

1. dapat dengan cara melakukan code tambahan pada backend untuk mengatasi validasi dengan memberikan anotasi-anotasi yang diperlukan, dapat juga dengan cara memberikan restric pada database
2. validasi diperlukan untuk dapat mencontrol data-data yang disimpan. Pada tutor ini validasi dilakukan oleh thymeleaf.

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?

Jawab:

1. karena ketika mensubmit ada request yang kita berikan ke server. Dengan POST data akan langsung dikirim ke server tanpa disimpan pada sebuah URL terlebih dahulu. Hal ini akan menghindarkan dari penyalahgunaan dalam proses pengiriman data.
2. Perlu, hal tersebut dapat dilakukan dengan menambahkan @RequestMethod

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

Jawab:

Tidak, karena GET dan POST memberikan request yang berbeda. Selain itu method hanya menerima satu jenis request method saja

Menambahkan Delete

Membuat method delete pada studentMapper

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

Lengkapi method deleteStudent pada StudentServiceDatabase

Nama: Muhammad Hasby Rosyadi
NPM: 1306386642

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

tambakan th:href pada viewall

```
<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>
```

buat method delete pada controller

```
@RequestMapping("/student/delete/{npm}")
public String delete (Model model, @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent (npm);
    if (student == null) {
        return "not-found";
    } else {
        studentDAO.deleteStudent (npm);
        return "delete";
    }
}
```

Menambahkan Update

Menambahkan method update pada studentMapper

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

Lengkapi method updateStudent pada StudentServiceDatabase

```
@Override
public void updateStudent(StudentModel student) {
    // TODO Auto-generated method stub
    log.info("Student berhasil diperbaharui");

    studentMapper.updateStudent(student);

}
```

Nama: Muhammad Hasby Rosyadi
NPM: 1306386642

buat method update pada controller

```
@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";
    } else {
        model.addAttribute ("student", student);
        return "form-update";
    }
}
```

buat method updateSubmit pada controller

```
   @PostMapping("/student/update/submit")
  @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";
  }
```

Menggunakan Object sebagai Parameter

Pada penggunaan Object akan berbeda pada updateSubmit pada controller

```
  @PostMapping("/student/update/submit")
    @RequestMapping(value = "", method=RequestMethod.POST)
  public String updateSubmit (@ModelAttribute StudentModel student)
  {
        StudentModel student = new StudentModel (npm, name, gpa);
      studentDAO.updateStudent(student);

      return "success-update";
  }
```

dan perubahan pada form-update.html

```
<h1 class="page-header">Update Editor</h1>

<form action="#" th:action="@{/student/update/submit}" th:object="${student}" method="post">
    <div>
        <p>NPM <input type="text" th:field="*{npm}" readonly = "true"/></p>
    </div>
    <div>
        <p>Nama <input type="text" th:field="*{name}" /></p>
    </div>
    <div>
         <p>GPA <input type="text" th:field="*{gpa}" /></p>
    </div>
    <div>
        <button type="submit" name="action" value="save">Update</button>
    </div>
</form>
```

Nama: Muhammad Hasby Rosyadi
NPM: 1306386642

pada form-update sebelumnya menggunakan th:value sebagai inputnya, namun untuk object menggunakan th:field selain hal tersebut, terdapat th:object pada tag form