A. Ringkasan dari materi yang Saya pelajari pada tutorial ini

Pada tutorial 4 ini saya belajar bagaimana menggunakan database dalam pengerjaan project dengan menggunakan Spring Boot app. Untuk operasi-operasi database ada pada studentMapper.java, yaitu dengan menggunakan beberapa anotasi seperti @Select, @Insert, @Delete, @Update.

Proses alur project ini sendiri yang saya pelajari adalah, dari StudentController akan memanggil StudentService yang merupakan interface lalu ke StudentServiceDatabase dimana sudah ada implementasinya. Di StudentServiceDatabase ini lah yang akan memanggil StudentMapper lalu dilakukan operasi pada Database.

Selain itu pada tutorial kali ini juga belajar mengenai debugging. Salah satu library yang digunakan adalah Slf4j yang ada pada library eksternal Lombok.

B. Hasil Jawaban dari poin-poin Tutorial

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 :

   Jika menggunakan Object sebagai parameter pada form POST, cara melakukan validasi input yang optional dan input yang required adalah dengan menambahkan secara eksplisit pada StudentController dengan membuat kondisi dari yang ingin dilakukan 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?

   Jawab :

   POST dan submit pada dasarnya berfungsi untuk mengirimkan nilai variabel ke halaman lain atau mengirimkan nilai ke database serta untuk mengambil nilai variabel dari halaman lain atau mengambil data pada database. Perbedaan dari GET dan POST sendiri adalah method POST akan mengirimkan data atau nilai langsung ke action untuk ditampung, tanpa menampilkan pada URL. Sedangkan method GET akan menampilkan data/nilai pada URL. Sehinga biasanya untuk submit form menggunakan POST karena isi dari form terkadang mengandung data-data pribadi dan lebih baik untuk tidak ditampilkan di link. Jadilah menggunakan POST. Lalu untuk header method ketika POST menggunakan “method = RequestMethod.POST”, dan jika GET menggunakan “method = RequestMethod.GET”.

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

   Jawab :

   Ketika implementasi method ketika POST dan GET relatif sama mungkin saja di buat menjadi satu method yang menerima lebih dari satu jenis request method yaitu GET dan POST sekaligus. Yaitu dengan -> method = { RequestMethod.GET, RequestMethod.POST }.

   Source : https://stackoverflow.com/questions/17987380/combine-get-and-post-request-methods-in-spring

C. Penjelasan Method-Method

a. Method yang di buat pada Latihan Menambahkan Delete

Pada StudentMapper.java ditambahkan method delete student dengan anotasi @Delete.

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

Lalu mengedit di StudentController.java ditambahkan method delete.

```
@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";
    }
}
```

Pada method delete tersebut diberikan parameter yaitu npm. Lalu dilakukan pengecekan apakah ada student dengan npm sesuai yang diberikan. Jika tidak ada maka ia akan langsung menampilkan halaman not-found.html. Kalau student dengan npm tersebut ditemukan maka akan dilakukan penghapusan student.

b. Method yang di buat pada Latihan Menambahkan Update,

Pada StudentMapper.java ditambahkan method update student dengan anotasi @Update.

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

Selanjutnya menambahkan method updateStudent pada interface StudentService serta menambahkan implementasi method updateStudent pada class StudentServiceDatabase.

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

Methodnya sendiri pada StudentController.java adalah sebagai berikut.

```java
@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";
    }
}

@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";
}
```

Penjelasan :

Pertama, yang dilakukan pada method update adalah menerima parameter berupa npm dari link. Lalu Melakukan pengecekan apakah terdapat student dengan npm tersebut. Kalau tidak ada maka akan menampilkan halaman not-found. Dan kalau student dengan npm tersebut ada, maka akan mengirim parameter berupa student dan menampilkan halaman form-update.

Untuk method updateSubmit yaitu method yang dijalankan ketika setelah masuk form update, dan mengklik tombol update. Pada method ini akan mengambil nilai-nilai yang di input di form-update. Lalu dari input-input tersebut di buat suatu objek student yang akan menjadi parameter untuk method updateStudent. Dari StudentController.java, akan menuju StudentServiceDatabase yang memanggil method updateStudent di studentMapper. Disinilah dilakukan update student yang ada di database, dirubah nilainya menjadi nilai-nilai yang ada di student yang dikirim di parameter.

c. Method yang di buat pada Latihan Menggunakan Object Sebagai Parameter

Untuk method update dengan menggunakan Object sebagai parameter ini sebenarnya tidak banyak berubah dari update.html. Yang berubah diantaranya :

1. Form-update.html

```html
<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="update">Update</button>
    </div>
</form>
```

Penjelasan : Menambahkan th:object pada form dan di masin-masing input ditambahkan th:field.

2. StudentController.java

Method yang berubah adalah method updateSubmit. Sedangkan method update tetap seperti sebelumnya.

```
@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";
    }
}

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

    return "success-update";
}
```

Penjelasan :

Pertama, yang dilakukan pada method update adalah menerima parameter berupa npm dari link. Lalu Melakukan pengecekan apakah terdapat student dengan npm tersebut. Kalau tidak ada maka akan menampilkan halaman not-found. Dan kalau student dengan npm tersebut ada, maka akan mengirim parameter berupa student dan menampilkan halaman form-update.

Untuk method updateSubmit yaitu method yang dijalankan ketika setelah masuk form update, dan mengklik tombol update. Di StudentController.java, method updateSubmit akan langsung menerima ModelAttribute berupa object StudentModel karena di form sudah ditambahkan th:object. Lalu selanjutnya akan menuju StudentServiceDatabase yang memanggil method updateStudent di studentMapper. Disinilah dilakukan update student yang ada di database, dirubah nilainya menjadi nilai-nilai yang ada di student yang dikirim di parameter.