Nama : Adil Krisnadi Pradana
NPM : 1406623631
Kelas : APAP-C

TUTORIAL 4

Lesson Learned

Hal yang dapat dipelajari dari tutorial kali ini diantaranya adalah penggunaan database dan juga melakukan debugging dalam project Spring Boot. Selain itu, juga diajarkan mengenai penggunaan Slf4j dan juga mengimplementasikan suatu method dengan menggunakan database. Kemudian, hal penting lainnya yang dapat dipelajari dari tutorial kali ini adalah penggunaan method POST untuk melakukan input, serta penggunaan object sebagai parameter.

PERTANYAAN

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? Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.**
   Validasi dapat dilakukan dengan cara memodifikasi kodingan pada back-end seperti memberikan beberapa anotasi yang diperlukan dalam melakukan validasi input. Validasi sendiri diperlukan untuk mengontrol data-data yang akan diinput, seperti misalnya batasan jumlah atau jenis karakter yang akan dimasukkan, dan lain sebagainya.

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 method POST, data yang kita submit akan langsung dikirim ke server tanpa harus disimpan pada suatu URL terlebih dahulu. Penanganan dapat dilakukan dengan menambahkan anotasi @RequestMethod.

3. **Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?**
   Tidak, karena GET dan POST memberikan request yang berbeda. Selain itu, method juga hanya dapat menerima satu jenis method request saja.

Method Menambahkan Delete

Pada awalnya, kita terlebih dahulu membuat method delete pada StudentMapper seperti berikut:

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

Selanjutnya, lengkapi method deleteStudent pada StudentServiceDatabase seperti berikut:

```
@Override
public void deleteStudent (String npm)
{
    studentMapper.deleteStudent (npm);
}
```

Selanjutnya, kita membuat method delete pada controller seperti berikut:

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

    return "delete";
}
```

Terakhir, lakukan modifikasi pada halaman HTML “View All” dengan menambahkan th:href seperti berikut:

```
<a th:href="'/student/delete/' + ${student.npm}" > Delete Data</a><br/>
<a th:href="'/student/update/' + ${student.npm}" > Update Data</a><br/>
```

Nantinya, fitur delete akan berjalan dengan menghapus data berdasarkan NPM yang telah dimasukkan sebelumnya.

Method Menambahkan Delete

Pada awalnya, kita terlebih dahulu membuat method update pada StudentMapper seperti berikut:

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

Selanjutnya, lengkapi method updateStudent pada StudentServiceDatabase seperti berikut:

```
@Override
public void updateStudent(StudentModel student) {
    // TODO Auto-generated method stub
    studentMapper.updateStudent(student);
}
```

Selanjutnya, kita membuat method update dan juga updateSubmit pada controller seperti berikut:

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

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


@PostMapping("/student/update/submit")
public String updateSubmit (@ModelAttribute StudentModel student) {

    studentDAO.updateStudent(student);

    return "success-update";
}
```

Method tersebut akan menjalankan fitur update berdasarkan dengan NPM yang telah dimasukkan sebelumnya, di mana akan ditampilkan form untuk mengedit data berupa NPM, Nama, dan GPA yang sebelumnya telah dimasukkan.

Menggunakan Object Sebagai Parameter

Penggunaan Object sebagai Parameter dapat dilakukan dengan memodifikasi method updateSubmit pada controller. Di mana yang digunakan bukanlah anotasi @RequestParam, melainkan anotasi @ModelAttribute. Selain itu, perubahan juga dilakukan pada halaman HTML “Form Update” seperti berikut:

```html
<form action="/student/update/submit" th: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>
```

Di mana untuk penggunaan Object menggunakan th:field, dan juga terdapat th:object pada tag form yang ada.