Bagaskoro Meyca Dwiyananda Putra
1506736530

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 mengecek parameter di object yang diperlukan, parameter akan dicek di dalam method, ketika parameternya null atau tidak sesuai bisa return dengan pesan error atau redirect ke form dengan tambahan pesan error/warning untuk mengisi field yang required.

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 method POST tidak menggunakan parameter di url sehingga data lebih aman dan tidak mudah terlihat. Perlu adanya penanganan yang berbeda untuk mendapatkan data dari GET maupun POST method.

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

   Tidak bisa, berbeda metode request harus memiliki method yang berbeda untuk menanganinya

- Method yang Anda buat pada Latihan Menambahkan Delete, jelaskan
  - pengubahan pada viewall.html

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

  - Method pada deleteStudent di StudentMapper

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

  - Method pada delete di STudentController

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

Pada viewall menghubungkan ke method pada controller. Sedangkan method deleteStudent akan mengeksekusi SQL query untuk menghapus data yang diinginkan dari npm sebagai parameter 'where'nya. deleteStudent dipanggil pada method controller yang sudah divalidasi jika npm tersebut ada di database.

- Method yang Anda buat pada Latihan Menambahkan Update, jelaskan
  - pengubahan pada viewall.html

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

  - Method pada updateStudent di StudentMapper

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

  - Method pada update di STudentController

```
@RequestMapping("/student/update/{npm}")
public String update(Model model, @PathVariable(value = "npm") String npm) {
    StudentModel student = studentDAO.selectStudent(npm);
    if (student != null) {
        model.addAttribute("std", new StudentModel());
        model.addAttribute("student", student);
    } else {
        model.addAttribute("npm", npm);
        return "not-found";
    }
    return "form-update";
}
```

Pada viewall menghubungkan ke method pada controller. Sedangkan method deleteStudent akan mengeksekusi SQL query untuk mengupdate data nama dan gpa dari npm sebagai parameter 'where'nya. update merupakan controller yang mengambil setail siswa berdasarkan npm kemudian ditampilkan pada form. Form kemudian akan mengarahkan ke controller pada saat di submit dan controller updateSubmit menjalankan perintah updateStudent yang sebelumnya sudah ada di StudentMapper.

- Method yang Anda buat pada Latihan Menggunakan Object Sebagai Parameter, jelaskan

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

    return "success-update";
}
```

Controler akan menerima object dari method post dari fom-update.html dan langsung menjalankan method pada studentMapper untuk menjalankan query update di database yang telah saya jelaskan di poin sebelumnya

```
<h1 class="page-header">Edit Student</h1>

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

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

Pada html nya th:object berperan sebagai pembentuk objek student dan th:field akan menjadi atributnya.