Dayana Sheilla Agathy
1406572630
Tutorial 4

Beberapa pertanyaan yang perlu Anda jawab:

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.
   Dengan cara menambahkan validasi di controller apakah object tersebut bernilai null atau tidak. Jika bernilai null maka tidak akan dijalankan oleh method tersebut.

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 GET itu dapat diakses langsung melalui browser sehingga keamanannya sangat kurang, maka dari itu untuk form submit yang bersifat rahasia karena biasanya mengandung data penting sebaiknya menggunakan method POST karena lebih aman. Sebaiknya method GET digunakan hanya untuk ketika ingin melihat sesuatu sedangkan method POST digunakan untuk mengubah sesuatu. Perlu, karena request parameter di controller harus sesuai dengan method yang kita request.

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   Ya, bisa. Dengan cara menggabungkan RequestMethod secara array.

```
122
123     @RequestMapping(value = "/student/update/submit", method = {RequestMethod.POST, RequestMethod.GET})
124     public String updateSubmit(@Valid @ModelAttribute("student") StudentModel student) {
125         studentDAO.updateStudent(student);
126         return "success-update";
127     }
128
```

- **Method Delete**

  Query delete student dengan parameter npm di class StudentMapper

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

  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){
        studentDAO.deleteStudent (npm);
        return "delete";
    }
    else{
        return "notfound";
    }
}
```

Sebelum melakukan delete, akan dilakukan pengecekkan terhadap npm, apakah npm null atau tidak. Jika npm null maka akan didirect ke halaman not-found.html. Jika ada npm maka method yang berisi query untuk mendelete akan dijalankan.

- **Method Update**

  Query update student dengan parameter npm di class StudentMapper

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

  Method untuk menampilkan form update dan submit 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) {
        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(@Valid @ModelAttribute("student") StudentModel student) {
    studentDAO.updateStudent(student);
    return "success-update";
}
```

  Form update

```
<body>

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

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

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

</body>
```

Untuk melakukan update data student, pertama yang dilakukan adalah mengecek apakah npm terdaftar. Jika tidak terdaftar maka akan didirect ke halaman not-found.html. Jika npm terdaftar maka akan didirect ke halaman form-update.html. Setelah itu, pada form update terdapat 3 buah parameter, yaitu nama, npm, dan gpa. Setelah memasukkan parameter tersebut, akan dicek apakah ketiga parameter tersebut valid atau tidak, jika valid maka data tersebut datanya akan diupdate dalam database.

Dayana Sheilla Agathy
1406572630
Tutorial 4

- **Object sebagai parameter**

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

```
<form action="/student/update/submit" th:object="${student}" method="POST">
    <div>
```

Pada method updateSubmit menerima parameter berupa StudentObject, karena parameter object tersebut akan digunakan dalam form-update.html yang menggunakan object untuk mengirim parameternya.