

Tutorial 4 Write-Up

Muhammad Alvin Aryan – 1506721844 – APAP [A]

Pertanyaan:

1. Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required? Apakah validasi diperlukan? Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.

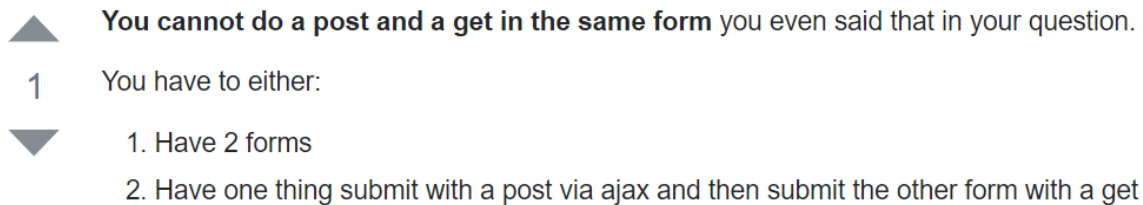
Untuk melakukan validasi pada form POST dapat dilakukan dengan menambahkan @PostMapping pada Controller. Validasi tetap dilakukan tetapi tidak langsung pada formnya. Sumber: <https://spring.io/guides/gs/validating-form-input/>

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?

Form submit biasanya menggunakan POST method dibanding GET method karena agar data tidak tampil secara eksplisit di URL, tujuannya untuk melindungi data. Perlu penanganan yang berbeda di header method.

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

Tidak bisa. Sumber: <https://stackoverflow.com/questions/6918991/submit-post-and-get-variables-in-one-form>

- 
- ▲ You cannot do a post and a get in the same form you even said that in your question.
- 1 You have to either:
- ▼
1. Have 2 forms
 2. Have one thing submit with a post via ajax and then submit the other form with a get

- Method yang Anda buat pada Latihan Menambahkan Delete, jelaskan

```
@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";
    }
}
```

Method ini mempunyai parameter npm sebagai @PathVariable, lalu npm yang telah diinputkan tersebut dicek terlebih dahulu melalui method selectStudent() pada object studentDAO. Jika object student tersebut null atau tidak ada, maka akan ditampilkan halaman “not-found”, kebalikannya

jika student tersebut tidak null yang berarti npm tersebut tersedia pada database, maka method ini akan menjalankan method deleteStudent() berdasarkan npm pada object studentDAO

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

Method ini akan menjalankan SQL DELETE yang ditrigger dengan eksekusi method delete dari Controller.

- Method yang Anda buat pada Latihan Menambahkan Update, jelaskan

```
@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";
    }
}
```

Method ini mempunyai parameter npm sebagai @PathVariable, lalu npm yang telah diinputkan tersebut dicek terlebih dahulu melalui method selectStudent() pada object studentDAO. Jika object student tersebut null atau tidak ada, maka akan ditampilkan halaman “not-found”, kebalikannya jika student tersebut tidak null yang berarti npm tersebut tersedia pada database, maka method ini akan menambahkan object student sebagai atribut pada model, lalu memanggil view “form-update”

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

Method ini akan menjalankan SQL UPDATE yang ditrigger dengan eksekusi method update dari Controller.

```
void updateStudent (StudentModel student);
```

Pendefinisian method ini terletak pada interface

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

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

Method ini menerima parameter object student dari class StudentModel lalu akan memanggil method updateStudent() pada class studentDAO dan menampilkan halaman “success-update”

Lesson learned:

Yang telah saya pelajari dari Tutorial 4 kali ini adalah saya dapat memahami bagaimana Spring Boot terkoneksi dengan database pada mysql. Lalu, saya juga dapat memahami bagaimana cara kerja dari form pada Spring Boot baik itu melalui method GET maupun method POST. Yang terakhir, saya dapat memahami mengenai salah satu cara untuk submit form melalui method POST tanpa harus mendefinisikan masing-masing atribut dari form satu-satu, yaitu dengan menggunakan Object sebagai Parameter.