Pertanyaan

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? Apakah validasi diperlukan?

   Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.
   - Validasi dapat dilakukan dengan menambahkan anotasi @Valid dan class BindingResult, yang bisa mendapatkan kesalahan yang diminta oleh Validator dan diimplementasikan di controller dalam request handler method (https://www.journaldev.com/2668/spring-validation-example-mvc-validator)

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?
   - Menggunakan method POST pada form submit, karena method POST digunakan untuk mengirim data yang dapat meliputi menyimpan data atau mengubah data, sedangkan method GET biasanya digunakan untuk me-*retrive* data

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   - Pada dasarnya, method GET dan POST tidak dapat digunakan dalam satu method yang sama, akan tetapi dapat dilakukan dengan mengubah single annotation menjadi @RestMethod (https://stackoverflow.com/questions/17338828/can-i-use-both-post-and-get-on-the-same-method)

Latihan

Method yang Anda buat pada Latihan Menambahkan Delete, jelaskan

- Menambahkan 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 {
        model.addAttribute("npm", npm);
        return "not-found";
    }
}
```

- Menambahkan method delete pada student mapper

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

- Menambah method delete pada student service

```
void deleteStudent(String npm);
```

- Menambahkan method delete pada student service database

```
@Override
public void updateStudent(StudentModel student) {
    studentMapper.updateStudent(student);
}
```

Method yang Anda buat pada Latihan Menambahkan Update, jelaskan

- Menambahkan method 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(@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";
}
```

- Menambahkan method update pada student mapper

```
@Select("update student set name = #{name}, gpa = #{gpa} where npm = #{npm}")
void updateStudent(StudentModel student);
```

- Menambah method update pada student service

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

- Menambahkan method update pada student service database

```
@Override
public void updateStudent(StudentModel student) {
    studentMapper.updateStudent(student);
}
```

Method yang Anda buat pada Latihan Menggunakan Object Sebagai Parameter

- Mengganti method delete pada controller menjadi sebagai berikut :

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

    return "success-update";
}
```