Nama: Arri Kurniawan
NPM: 1606954735

Ringkasan

Pada tutorial 4, dipelajari bagaimana membuat sebuah *project* Spring Boot yang dapat terhubung dengan database, yang mana dbms yang digunakan pada kesempatan kali ini adalah MySQL. Kemudian, pada kesempatan ini dipelajari juga cara menggunkan POST method dan GET method yang berfungsi untuk melakukan request-response dari program yang dibuat seperti membuat method delete dan update. Pada kesempatan ini, juga dipelajari bagaimana perilaku yang harus dilakukan jika parameter yang dikirimkan berupa object atau bukan.

Jawaban dari 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.)
   Jawaban:

   Validasi dapat dilakukan dengan cara menambahkan validation annotations pada class model yang telah dibuat. Adapun annotations yang dapat digunakan misalnya @NotNull untuk membuat input menjadi required atau @Null untuk pilihan optional dari package javax.validation.constraints [1] atau bisa juga menggunakan @NonNull jika ingin menggunakan annotations dari package lombok[2].

   Validasi tersebut diperlukan karena program yang dibuat sekarang langsung terhubung ke dalam database. Pada setiap table yang dalam suatu database pasti memiliki constraint tertentu dan untuk mengisi data ke dalam table tersebut constraint yang ada harus selau terpenuhi. Dengan dilakukannya validasi adalah untuk menghindari error yang mungkun terjadi ketika ada transaksi data dengan database.

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?
   Jawaban:
   GET method memiliki beberapa kelemahan seperti terbatasnya parameter yang dapat dikirimkan maupun parameter yang dapat terlihat pada browser sehingga sangat berpengaruh jika ada data yang sensitif. Oleh karena itu, POST method biasanya digunakan untuk submit data yang akan diproses pada suatu resourse tertentu sedangkan GET method biasanya digunakan untuk melakukan request data dari suatu resource tertentu [3].
   Penanganan yang dilakukan pada header atau body method di controller pasti berbeda tergantung pada method yang digunakan. Misalkan jika menggunakan POST method dalam annotation @RequestMapping maka digunakan method RequestMethod.POST atau menggunakan annotation @PostMapping pada header di controller.

---

[1] https://docs.oracle.com/javaee/7/api/javax/validation/constraints/package-summary.html
[2] https://projectlombok.org/api/lombok/package-summary.html
[3] https://www.w3schools.com/tags/ref_httpmethods.asp

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   Jawaban:
   Tidak bisa, sebab satu method hanya dapat menggunakan satu request method saja apabila lebih dari satu akan muncul error duplication.

Latihan – Method Delete

Untuk menambahkan method delete pada tutorial kali ini, hal pertama yang dilakukan adalah dengan membuat method delete pada controller seperti pada Gambar 1. Di mana proses yang dilakukan pada method tersebut pertama kali adalah melakukan pencarian data student dengan menggunakan method selectStudent yang telah ada sebelumnya.

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

Gambar 1. Method Delete di Controller

Kemudian, apabila data student tersebut ada dalam database maka selanjutnya akan memanggil method deleteStudent yang ada pada bagian service (untuk melihat isi dari method tersebut dapat melihat Gambar 2). Setelah, selesai melakukan method delete tersebut controller akan mengembalikan tampilan ke halaman delete.html yang telah dibuat. Jika, sebelumnya data student tidak ditemukan maka akan dilakukan pengembalian kepada halaman not-found.html dengan parameter npm yang dikirimkan.

```
@Override
public void deleteStudent (StudentModel student)
{
    log.info("student " + student.getNpm() + " deleted");
    studentMapper.deleteStudent(student);
}

@Override
public void updateStudent (StudentModel student) {
    log.info("student " + student.getNpm() + " updated");
    studentMapper.updateStudent(student);
}
```

Gambar 2. Method deleteStudent & updateStudent di StudentServiceDatabase

Untuk melihat dari studentMapper pada Gambar 2, dapat dilihat pada Gambar 3 berikut.

```java
@Mapper
public interface StudentMapper
{
    @Select("select npm, name, gpa from student where npm = #{npm}")
    StudentModel selectStudent (@Param("npm") String npm);

    @Select("select npm, name, gpa from student")
    List<StudentModel> selectAllStudents ();

    @Insert("INSERT INTO student (npm, name, gpa) VALUES (#{npm}, #{name}, #{gpa})")
    void addStudent (StudentModel student);

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

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

}
```

Gambar 3. Interface StudentMapper

Latihan – Method Update

Selanjutnya, untuk menambahkan method update disiapkan 2 method yaitu method update dan method updateSubmit (lihat Gambar 4). Method update digunakan untuk mengambil dan menampilkan data student yang akan diperbarui/update. Proses yang dilakukan pada method tersebut pertama kali ada mencari data student dulu di database seperti pada method delete. Kemudian, jika data tersebut ditemukan makan akan dikembalikan dan ditampilkan pada halaman form-update.html yang telah dibuat dan jika tidak akan ditampilkan halaman not-found.html. Adapun halaman form-update.html dapat dilihat pada Gambar 5. Dari halaman tersebut, dapat dilakukan pengubahan data untuk student yang bersangkutan, selanjutnya apabila data tersebut akan disimpan maka akan memanggil method updateSubmit.

```
@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 = true) String name,
        @RequestParam(value = "gpa", required = true) Double gpa) {
    StudentModel student = new StudentModel (npm, name, gpa);
    studentDAO.updateStudent(student);

    return "success-update";
}
```

Gambar 4. Method Update di Controller

Cara kerja dari method updateSubmit sendiri adalah pertama kali method tersebut akan menangkap parameter yang telah dikirimkan oleh halaman html dengan menggunakan @RequestParam dan selanjut dari parameter tesebut dibuatkan object student. Kemudian, object tersebut menjadi parameter dari method updateStudent di StudentServiceDatabase (lihat Gambar 2). Untuk melihat query yang dilakukan pada method updateStudent tersebut dapat dilihat pada Gambar 3.

Latihan – Object sebagai Parameter

Pada bagian ini, method updateForm yang dibuat hampir cara kerjanya dengan method update yang telah dibuat sebelumnya. Yang menjadi pembeda pada method ini adalah pada bagian annotation yang digunakan adalah @GetMapping & @PostMapping sedangkan pada method sebelumya yang digunakan adalah @RequestMapping. Kemudian, pembeda selanjutnya adalah updateFormSubmit hanya menerima parameter object yang dikirimkan dari halaman html sedangkan pada method update sebelum ada 3 parameter yang dikirimkan yaitu String npm, String name, dan double gpa. Untuk melihat method tersebut dapat dilihat pada Gambar 5.

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

    if (student != null) {
        model.addAttribute ("student", student);
        return "form-update2";
    } else {
        model.addAttribute ("npm", npm);
        return "not-found";
    }
}

@PostMapping("/student/updateForm/submit")
public String updateFormSubmit(@ModelAttribute StudentModel student) {
    studentDAO.updateStudent(student);
    return "success-update";
}
```

Gambar 5. Method updateForm di Controller