Kevin Noel Anugrah
1506689370

Menggunakan Database dan Melakukan Debugging dalam Project Spring Boot

I. Menambahkan Delete

Pada interface StudentMapper ditambahkan method delete dengan query delete.

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

Lalu, tambah method delete pada StudentService dan implementasikan dalam StudentServiceDatabase yang memanggil method DAO tersebut.

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

Tambahkan mapping dengan path variable pada controller, di mana dilakukan pengecekan apakah query tadi mereturn null atau tidak. Tampilkan view yang sesuai.

```
@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);model.addAttribute ("student", student);
         return "delete";
     }
     model.addAttribute ("npm", npm);
     return "not-found";
 }
```

II. Menambahkan Update

Pada interface StudentMapper ditambahkan method update dengan query update.

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

Lalu, tambah method update pada StudentService dan implementasikan dalam StudentServiceDatabase yang memanggil method DAO tersebut.

```
@Override
public void updateStudent(String npm, String name, Double gpa) {
        log.info("student " + npm + " updated");
        studentMapper.updateStudent(npm, name, gpa);
}
```

Buat form update yang menerima atribut model dari student dan mengisi default value dari input.

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:th="http://www.thymeleaf.org">
<head>
<title>Update student</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
        <h1 class="page-header">Student Editor</h1>
        <form action="/student/update/submit" method="post">
        <div>
                <label for="npm">NPM</label> <input type="text" name="npm" readonly="true"
                th:value="${student.npm}"/>
        </div>
        <div>
                <label for="name">Name</label> <input type="text" name="name"
                th:value="${student.name}"/>
                </div>
        <div>
                <label for="gpa">GPA</label> <input type="number" step="any" name="gpa"
                 th:value="${student.gpa}"/>
        </div>
        <div>
                <button type="submit" name="action" value="save">Save</button>
        </div>
        </form>
</body>
</html>
```

Tambahkan mapping untuk form update dengan path variable pada controller, di mana dilakukan pengecekan apakah query tadi mereturn null atau tidak. Jika student ada, lanjutkan ke form update.

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

Tambahkan mapping untuk hasil submit form pada controller, di mana method update akan dilakukan dan halaman success ditampilkan.

```
@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)
    {
        studentDAO.updateStudent(npm,name,gpa);
        return "success-update";
    }
```

III. Menambahkan Object Sebagai Parameter

Ubah parameter updateSubmit untuk menerima StudentModel.

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

Tambahkan th:object sebagai penanda objek StudentModel pada tag form dan th:field pada tag input sebagai penanda field StudentModel.

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:th="http://www.thymeleaf.org">
<head>
<title>Update student</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
        <h1 class="page-header">Student 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="${student.name}"/>
        </div>
        <div>
                <label for="gpa">GPA</label> <input type="number" step="any" name="gpa" th:field="*{gpa}"
                th:value="${student.gpa}"/>
        </div>
        <div>
                <button type="submit" name="action" value="save">Save</button>
        </div>
        </form>
</body>
</html>
```

IV. Pertanyaan: Bagaimana dengan validasi required input jika menggunakan parameter object?

Asumsi : validasi optional yang dimaksud adalah kemampuan untuk menginsert variable null pada input form update. Karena, dari kode yang disediakan untuk setiap update semua field akan terupdate walaupun variablenya sama, sehingga jika tidak ingin mengganti suatu value, default value dimunculkan pada setiap isian form pada saat halaman muncul adalah value student pada saat itu.

Input yang kosong akan mengisi string kosong pada database, sehingga diperlukan cara untuk mengubah string kosong menjadi null value. Salah satu cara adalah mengkustomasi data binding pada request parameter. Berikut cara yang diusulkan salah satu pengguna dari StackOverflow.com:

```
@InitBinder
    public void initBinder(WebDataBinder binder) {
        binder.registerCustomEditor(String.class, new StringTrimmerEditor(true));
    }
```

Untuk membatasi input angka gpa, ganti type input menjadi number pada html.

```
<input type="number" step="any" name="gpa" th:field="*{gpa}" th:value="${student.gpa}"/>
```

Tetapi, primitive variable pada StudentModel tidak dapat menghandle null value. Ubah tipe data menjadi wrapper classnya.

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class StudentModel
{
    private String npm;
    private String name;
    private Double gpa;

}
```

V. Pertanyaan: Mengapa form submit menggunakan POST, bukan GET method?

Jika form di submit menggunakan get, aplikasi akan mencoba mengembalikan halaman yang dimapping dengan "/submit," beserta semua parameter yang ada di form karena dihandle oleh mapping form update.

Sama halnya dengan controller, jika mappingnya adalah getmapping maka request post dari submit dihandle oleh mapping form update yang tidak dispesifikasikan sebagai postmapping/getmapping.

VI. Pertanyaan: Apakah mungkin method menerima GET sekaligus POST Request?

Ya.