

Code :

- Viewall.html

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3   <head>
4     <title>View All Students</title>
5   </head>
6   <body>
7     <h1>All Students</h1>
8
9     <div th:each="student, iterationStatus: ${students}">
10      <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
11      <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
12      <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
13      <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
14      <a th:href="/student/delete/" + ${student.npm}" > Delete Data </a><br/>
15      <a th:href="/student/update/" + ${student.npm}" > Update Data </a><br/>
16    </div>
17  </body>
18 </html>
```

- method deleteStudent pada StudentMapper

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

- method deleteStudent pada StudentServiceDatabase

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

- method delete pada StudentController

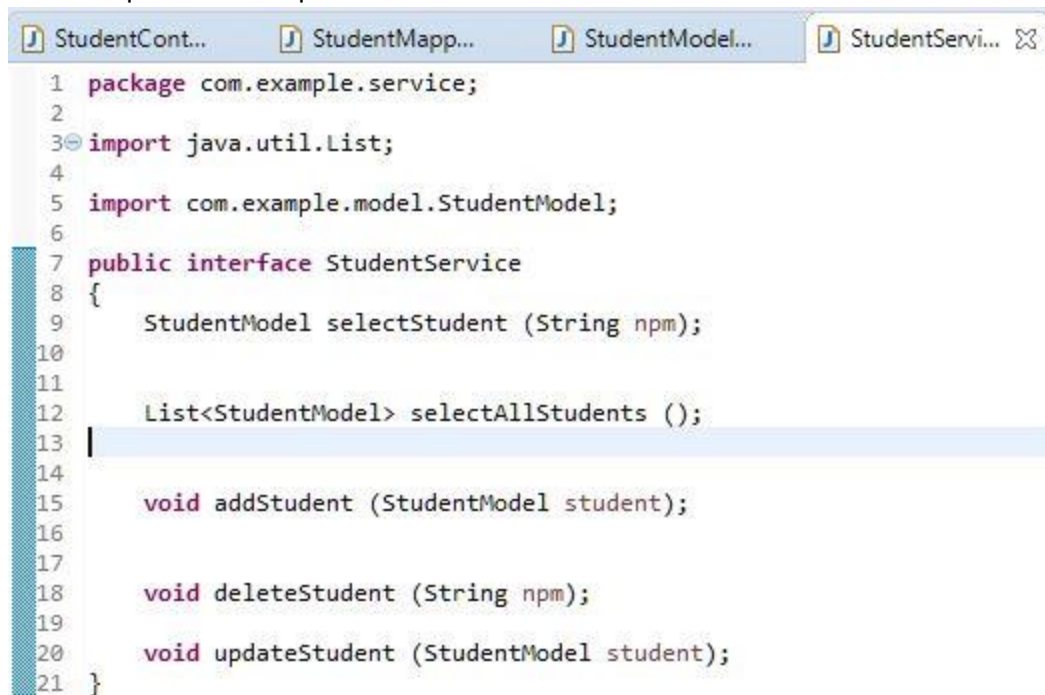
```
@RequestMapping("/student/delete/{npm}")
public String delete (Model model, @PathVariable(value = "npm") String npm)
{
    for(int i = 0; i < studentDAO.selectAllStudents().size(); i++){
        if(studentDAO.selectAllStudents().get(i).getNpm().equalsIgnoreCase(npm)){
            studentDAO.deleteStudent (npm);
            return "delete";
        }
    }

    return "not-found";
}
```

- method updateStudent pada StudentMapper

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

- method updateStudent pada interface StudentService



```

1 package com.example.service;
2
3 import java.util.List;
4
5 import com.example.model.StudentModel;
6
7 public interface StudentService
8 {
9     StudentModel selectStudent (String npm);
10
11
12     List<StudentModel> selectAllStudents ();
13
14
15     void addStudent (StudentModel student);
16
17
18     void deleteStudent (String npm);
19
20     void updateStudent (StudentModel student);
21 }

```

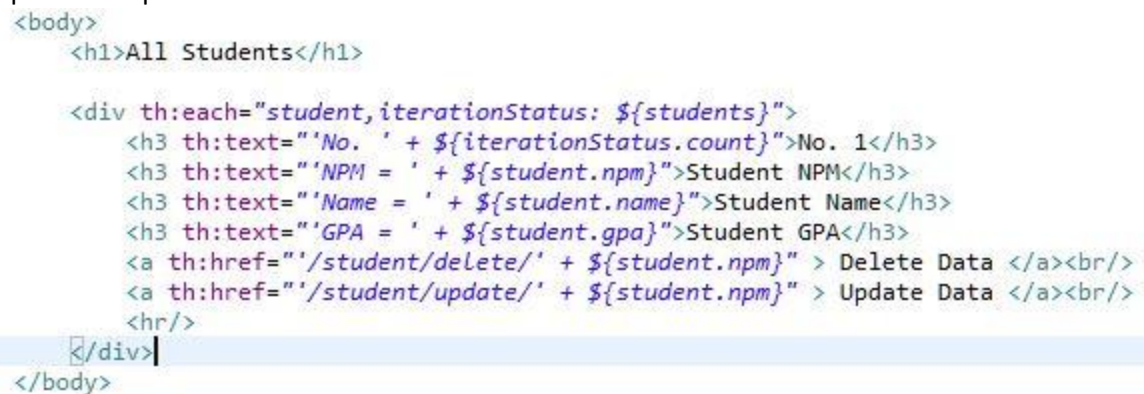
- method updateStudent pada StudentServiceDatabase

```

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

```

- updateData pada viewall



```

<body>
    <h1>All Students</h1>

    <div th:each="student, iterationStatus: ${students}">
        <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
        <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
        <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
        <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
        <a th:href="'/student/delete/' + ${student.npm}" > Delete Data </a><br/>
        <a th:href="'/student/update/' + ${student.npm}" > Update Data </a><br/>
        <hr/>
    </div>
</body>

```

- copy view form-add.html menjadi form-update.html

```

6 <title>Update student</title>
7 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
8 </head>
9
10 <body>
11
12 <h1 class="page-header">Update Student</h1>
13
14 <form th:object="${student}" action="/student/update/submit" method="post">
15     <div>
16         <label for="npm">NPM</label> <input th:field="*{npm}" type="text" name="npm" readonly="true" th:value="${student.npm}" />
17     </div>
18     <div>
19         <label for="name">Name</label> <input th:field="*{name}" type="text" name="name" th:value="${student.name}" />
20     </div>
21     <div>
22         <label for="gpa">GPA</label> <input th:field="*{gpa}" type="text" name="gpa" th:value="${student.gpa}" />
23     </div>
24     <div>
25         <button type="submit" name="action" value="save">Update</button>
26     </div>
27 </form>
28
29 </body>

```

- copy view success-add.html menjadi success-update.html

```

1 <html>
2     <head>
3         <title>Add</title>
4     </head>
5     <body>
6         <h2>Data berhasil diupdate</h2>
7     </body>
8 </html>

```

- method update pada StudentController

```

@RequestMapping("/student/update/{npm}")
public String update (Model model, StudentModel student)
{
    List<StudentModel> list = studentDAO.selectAllStudents();
    for(int i = 0; i< list.size(); i++){
        if(list.get(i).getNpm().equalsIgnoreCase(student.getNpm())){
            model.addAttribute("student", student);
            return "form-update";
        }
    }

    return "not-found";
}

```

- method updateSubmit pada StudentController

```

@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(new StudentModel(npm, name, gpa));
    return "success-update";
}

```

- menambahkan **th:object="\${student}"** pada tag di view & menambahkan **th:field="\*{[nama\_field]}"** pada setiap input

```
<body>

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

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

</body>
```

- Method updateSubmit pada StudentController yang hanya menerima parameter berupa StudentModel

```
@RequestMapping("/student/update/{npm}")
public String update (Model model, StudentModel student)
{
    List<StudentModel> list = studentDAO.selectAllStudents();
    for(int i = 0; i < list.size(); i++){
        if(list.get(i).getNpm().equalsIgnoreCase(student.getNpm())){
            model.addAttribute("student", student);
            return "form-update";
        }
    }
    return "not-found";
}
```

## Penjelasan

- Method yang anda buat pada latihan menambahkan delete  
Akan memproses masukan pada param yang diawali dengan query pada class StudentMapper dengan @Delete, dilengkapi dengan method pada class StudentServiceDatabase dengan log serta StudentController, pada StudentController melakukan pencarian student sesuai masukan. Jika tidak ketemu akan mengeluarkan "not-found" >view not-found.html, jika ketemu akan mengeluarkan "delete" >view delete.html.
- Method yang anda buat pada latihan menambahkan update  
Membuat query dan @Update dengan method yang menerima parameter StudentModel pada class StudentMapper >>> membuat method updateStudent pada interface StudentService dengan implementasi pada StudentServiceDatabase dengan log. Menambahkan link update data dengan merefer ke student/update/{npm} pada view wall.html >>> membuat form-update.html untuk mengisi apa yang mau diupdate. Pada Student Controller membuat method dengan request mapping /student/update/{npm} untuk menemukan student yang dituju >>> jika tidak ketemu akan merefer ke view not-found.html, jika ada merefer form-update dan mengisi apa yang mau diupdate pada form-update.html >>> jika tombol submit ditekan maka akan merefer ke updateSubmit kemudian mengeluarkan success-update pada view success-update.html.

Ammar Shidqi

1506729185

APAP – A

- Method yang anda buat pada latihan menggunakan object sebagai parameter  
Mengubah penggunaan RequestParam agar menjadi object yaitu StudentModel, pada form-update.html memberikan th:object="\${student}" pada tag <form>, th:field="{[nama\_field]}" pada setiap input, mengubah param updateSubmit dengan Student Controller maka menerima parameter berupa StudentModel.