Method Delete

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

Pertama membuat sebuah method *abstract* pada *class interface Student Mapper*. Gunakan anotasi @Delete untuk mengeksekusi query menghapus data dalam database. Parameter method tersebut menggunakan objek dari *student* dan menggunakan @Param untuk menangkap data dengan inisal "student".

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

Tambahkan method *abstract deleteStudent* pada *class interface StudentService*. Pada parameter method tersebut menerima sebuah *String* npm.

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

Pada *class StudentServiceDatabase* implementasikan method *deleteStudent* dari *class interface StudentService*. Dalam method tersebut gunakan method *deleteStudent* dari *studentMapper* untuk melakukan penghapusan data dari database.

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

Tambahkan method *delete* pada *class StudentController*. Dalam method tersebut terdapat kode untuk memvalidasi data *student* dengan NPM tersebut ada, jika ada maka akan me*return* halaman *delete*, jika tidak ada maka akan me*-return* halaman *not-found*.

Tambahkan sebuah link untuk menghapus data student pada halaman viewall.

Berikut adalah halaman *not-found* jika data yang akan dihapus berdasarkan NPM yang diberikan tidak ditemukan.

Berikut adalah halaman *delete* untuk menginformasikan bahwa data sudah berhasil dihapus, halaman ini muncul ketika NPM yang dimasukkan ada datanya.

All Students

No. 1

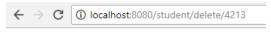
NPM = 4213

Name = zetra wibawa putra

GPA = 9.0

Delete Data Update Data

Berikut adalah data dari semua student.



Data berhasil dihapus

Ketika diklik "Delete Data" maka data student tersebut terhapus.

Method Update

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

Pertama membuat sebuah method *abstract updateStudent* pada *class interface*StudentMapper. Gunakan anotasi @Update untuk mengeksekusi query meng-update data dalam database berdasarkan NPM yang diberikan. Parameter method tersebut menggunakan objek dari student dan menggunakan @Param untuk menangkap data dengan inisal "student".

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

Tambahkan method *abstract updateStudent* pada *class interface StudentService*. Pada parameter method tersebut menerima sebuah objek *student*.

```
@Override
    public void updateStudent(StudentModel student) {
        // TODO Auto-generated method stub
        studentMapper.updateStudent(student);
        log.info("student " + student.getNpm() + " updated");
    }
}
```

Pada class StudentServiceDatabase implementasikan method updateStudent dari class interface StudentService. Dalam method tersebut gunakan method updateStudent dari studentMapper untuk melakukan penghapusan data dari database.

```
@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";
}
```

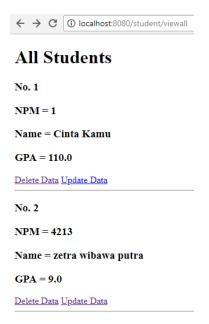
Tambahkan method *update* pada *class StudentController*. Dalam method tersebut terdapat kode untuk memvalidasi data *student* dengan NPM tersebut ada, jika ada maka akan me*return* halaman *form-update*, jika tidak ada maka akan me*-return* halaman *not-found*.

Tambahkan method *updateSubmit* pada *class StudentController*. Method tersebut untuk menyimpan data *student* yang sudah dimasukkan pada *form*.

Berikut adalah halaman *not-found* jika data yang akan dihapus berdasarkan NPM yang diberikan tidak ditemukan.

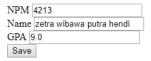
Berikut adalah halaman *form-update* untuk mengubah data *student* berdasarkan NPM, halaman ini muncul ketika NPM yang diberikan *valid*.

Berikut adalah halaman *success-update*, halaman ini muncul ketika data berhasil diubah dan disimpan kedalam database.



Berikut adalah data dari seluruh data *student* yang ada di database, dimana halaman ini diakses lewat localhost:8080/student/viewall.

| Pro | oblem Editor |
|----------|--|
| NPM [| .2.10 |
| GPA Save | zetra wibawa putra 0.0 |
| | |
| ← → | C O localhost:8080/student/update/4213 |



Berikut adalah halaman pengubahan data *student*, kasus ini mengganti nama dari ""zetra wibawa putra" menjadi "zetra waibawa putra hendi".



Data berhasil di Update

Ketika sesudah di submit maka data berhasil di Update.

Method Object

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

Pada class StudentController tambahkan sebuah method updateSubmitObject (tanpa mengubah method updateSubmit). Method ini sama dengan method updateSubmit perbedaanya adalah method updateSubmitObject mengubah sebuah data dari databsase dengan sebuah objek dari form halaman yang diinputkan atau tanpa mendefinisikan @RequestParam berdasarkan masing-masing variabel yang dimasukkan.

Berikut adalah halaman form-update yang sudah diubah kode pada form-nya.

Soal

- 1. Dengan melakukan validasi masing-masing variabel yang didapatkan @RequestParam dari hasil *input form* tersebut. Validasi ini penting atau dibutuhkan ketika ada suatu masukkan membutuhkan *value* yang khusus atau bersifat opsional.
- 2. Menggunakan method POST ketika data yang akan dikirim tersebut adalah data yang penting atau bersifat *private*, untuk menjamin keamanan data tersebut maka pengiriman methodnya menggunakan method POST dibandingkan dengan method GET. GET memungkin pengguna dapat memasukkan sebuah data di URL secara langsung. Iya perlu penanganan yang berbeda.
- 3. Bisa, karena pada @RequestMapping dapat menerima lebih satu jenis method POST dan GET.