Nama : Olvi Lora S

NPM : 1606954943

Latihan Menambahkan Delete

1. Menambahkan row baru untuk menampilkan link Delete Data di viewall.html

```
<a th:href="'/student/delete/' + ${student.npm}"> Delete Data </a><br/>
```

2. Menambahkan method deleteStudent di class StudentMapper dan menambahkan anotasi @DELETE dan menuliskan script untuk mendelete student berdasarkan npm yang diberikan.

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

3. Memastikan method deleteStudent sudah didefinisikan di class StudentService

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

4. Menambahkan method deleteStudent di class StudentServiceDatabase (class yang meng-implement interface StudentService) yang memanggil method deleteStudent di class StudentMapper. Kemudian menambahkan log info untuk lebih mudah dalam melakukan debugging

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

5. Menambahkan method delete di class StudentController dan menambahkan validasi. Jika npm ada, maka tampilkan view delete.html. Jika npm tidak ditemukan maka tampilkan view not-found.html

```
@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);
    }else {
        return "not-found";
    }

    return "delete";
}
```

6. Melakukan Insert Student
    - Membuka url (http://localhost:8080/student/add) untuk menambahkan data Student dan akan ditampilkan view form-add.html seperti gambar berikut ini.

Problem Editor

NPM
Name
GPA
Save

- Isi field npm, nama dan gpa kemudin klik button Save data kita tersimpan di database. Untuk melihat data Student yang sudah diinsert buka url http://localhost:8080/student/viewall
  Akan ditampilkan viewall.html seperti gambar berikut ini

All Students

No. 1

NPM = 11112110

Name = rona

GPA = 3.25

Delete Data
Update Data

No. 2

NPM = 1606954943

Name = olvi

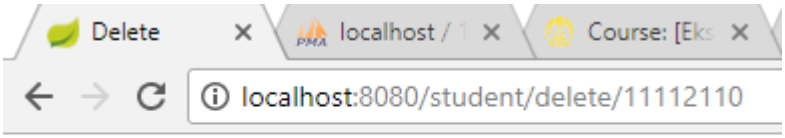
GPA = 3.5

Delete Data
Update Data

- Pilih data Student yang akan dihapus, klik link Delete Data. Page akan redirect ke url http://localhost:8080/student/delete/11112110 sesuai yang sudah didefinisikan di view viewall.html

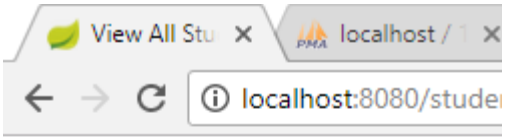
```
<a th:href="'/student/delete/' + ${student.npm}"> Delete Data </a><br/>
```

- Apabila data berhasil dihapus akan ditampilkan view delete.html seperti gambar berikut ini.



Data berhasil dihapus

- Untuk membuktikan data Student telah terhapus, buka url http://localhost:8080/student/delete/11112110 untuk menampilkan semua data mahasiswa. Data mahasiswa dengan NPM 11112110 sudah tidak ada lagi seperti ditunjukkan gambar berikut.



All Students

No. 1

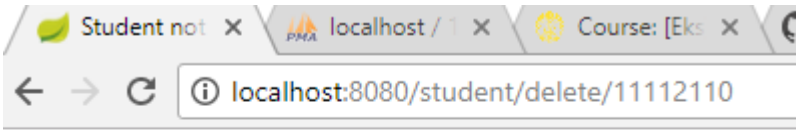
NPM = 1606954943

Name = olvi

GPA = 3.5

Delete Data
Update Data

- Apabila kita menghapus data mahasiswa yan tidak ada di database dengan mengakses url http://localhost:8080/student/delete/11112110, maka akan ditampilkan view not-found.html seperti ditunjukkan gambar berikut ini.



Student not found

NPM = 11112110

Latihan Menambahkan Update

1. Menambahkan method updateStudent pada class StudentMapper dan menambahkan query SQL untuk meng-update tabel

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

2. Menambahkan method updateStudent di interface StudentService

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

3. Menambahkan implementasi method updateStudent di class StudentServiceDatabase (class yang mengimplementasi interface StudentService). Kemudia menambahkan log untuk memudahkan debugging

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

4. Menambahkan row di viewall.html sebagai link untuk Update Data

```
<a th:href="'/student/update/' + ${student.npm}"> Update Data </a><br/>
```

5. Menambahkan form-update.html untuk tampilan ketika meng-update data Student

```
<h1 class="page-header">Update Student</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="text" name="gpa" th:value="${student.gpa}"/>

    </div>

    <div>
        <button type="submit" name="action" value="update">Update</button>
    </div>
</form>
```

6. Menambahkan view success-update.html untuk tampilan ketika update sudah selesai

```
<html>
    <head>
        <title>Update</title>
    </head>
    <body>
        <h2>Data berhasil diupdate</h2>
    </body>
</html>
```

7. Menambahkan method update pada class StudentControoler yang akan me-mapping request url. Kemudian melakukan validasi jika npm yang dijadikan parameter ada maka menampilkan view form-update.html. Sementara jika tidak ada, akan menampilkan view not-found.html

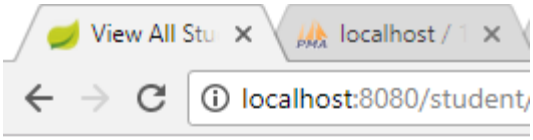
```
@RequestMapping("/student/update/{npm}")
public String update (Model model, @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent(npm);
    if(student != null) {
        studentDAO.updateStudent(npm);
    }else {
        return "not-found";
    }

    return "form-update";
}
```

8. Menambahkan method updateSubmit pada class 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)
{
    StudentModel student = studentDAO.selectStudent (npm);
    student.setNpm(npm);
    student.setName(name);
    student.setGpa(gpa);
    studentDAO.updateStudent(student);
    return "success-update";
}
```

9. Menjalankan aplikasi untuk menguji method update yang ditambahkan
   - Menampilkan data Student yang diambil dari database seperti ditunjukkan gambar berikut ini.



All Students

No. 1

NPM = 1606954943

Name = olvi

GPA = 3.5

Delete Data
Update Data

No. 2

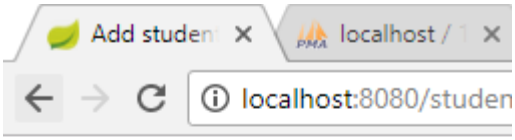
NPM = 12345

Name = vivi

GPA = 3.75

Delete Data
Update Data

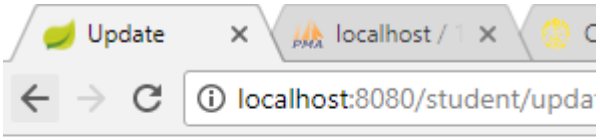
- Terdapat dua data Student dengan npm 1606954943 dan 12345
- Meng-update nama Student dengan npm 1606954943 menjadi olvi lora dengan mengklik link Update Data dan akan ditampilkan gambar sebagai berikut.



Update Student

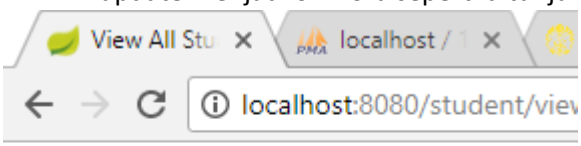
NPM 1606954943
Name olvi lora
GPA 3.5
Update

- Setalah field yang ingin diubah selesai diubah, pilih tombol Update maka akan ditampilkan view success-update.html yang ditunjukkan seperti gambar berikut ini.



Data berhasil diupdate

- Untuk melihat datanya berhasil di-update buka url http://localhost:8080/student/viewall akan ditampilkan view viewall yang menampilkan data mahasiswa. Nama mahasiswa yang sebelumnya olvi telah ter-update menjadi olvi lora seperti ditunjukkan gambar berikut.



All Students

No. 1

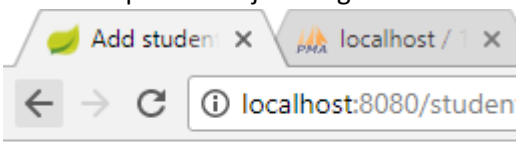
NPM = 1606954943

Name = olvi lora

GPA = 3.5

Delete Data
Update Data

- Meng-update gpa mahasiswa dengan npm 12345 menjadi 3.8 di form-update seperti ditunjukkan gambar berikut. Kemudian klik tombol Update.



Update Student

NPM 12345
Name vivi
GPA 3.8
Update

- Buka kembali url http://localhost:8080/student/viewall untuk memastikan data Student telah ter-update. Gambar berikut ini menunjukkan data gpa Student dengan npm 12345 telah berubah dari 3.75 menjadi 3.8

No. 2

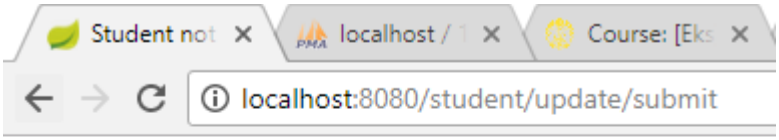
NPM = 12345

Name = vivi

GPA = 3.8

Delete Data
Update Data

- Apabila kita meng-udate data Student dengan npm yang tidak ada di database maka akan ditampilkan view not-found.html seperti ditunjukkan gambar berikut ini.



Student not found

NPM = submit

Latihan Menggunakan Object Sebagai Parameter

1. Menambahkan th object pada view form-update seperti ditunjukkan gambar berikut.

```
<form action="/student/update/submit" method="post" th:object="${student}">
```

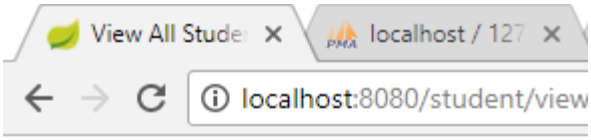
2. Menambahkan th field di setiap input

```
<div>
    <label for="npm">NPM</label> <input type="text" name="npm" readonly="true" th:value="${student.npm}" th:field="${student.npm}" />
</div>
<div>
    <label for="name">Name</label> <input type="text" name="name" th:value="${student.name}" th:field="${student.name}"  />
</div>
<div>
    <label for="gpa">GPA</label> <input type="text" name="gpa" th:value="${student.gpa}" th:field="${student.gpa}" />
```

3. Mengubah method updateSubmit pada StudentController

```
@RequestMapping(value="/student/update/submit", method=RequestMethod.POST)
public String updateSubmit(@ModelAttribute StudentModel student)
//       @RequestParam(value="npm", required=false) String npm,
//       @RequestParam(value="name", required=false) String name,
//       @RequestParam(value="gpa", required=false) double gpa)
{
//  StudentModel student = studentDAO.selectStudent (npm);
//  student.setNpm(npm);
//  student.setName(name);
//  student.setGpa(gpa);
    studentDAO.updateStudent(student);
    return "success-update";
}
```

4. Menguji kembali method update yang telah diubah
   - Menampilkan data Student sebelum di-update dengan membuka url http://localhost:8080/student/viewall



All Students

No. 1

NPM = 1606954943

Name = olvi lora

GPA = 3.5

Delete Data
Update Data

No. 2

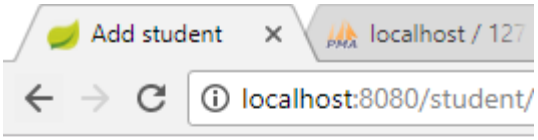
NPM = 12345

Name = vivi

GPA = 3.8

Delete Data
Update Data

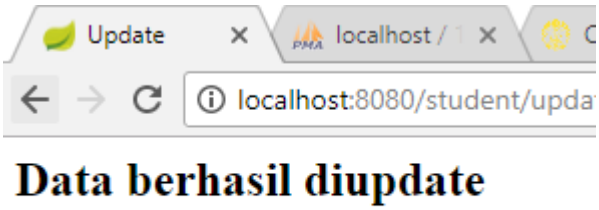
- Meng-update nama Student dengan npm 1606954943 yang sebelumnya olvi lora menjadi mega cs di view form-update.html seperti gambar berikut.



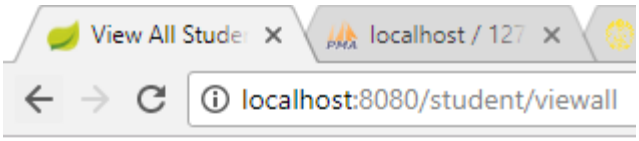
Update Student

NPM 1606954943
Name mega cs
GPA 3.5
Update

- Setelah mengklik tombol Update akan ditampilkan view success-update.html seperti ditunjukkan gambar berikut.



- Untuk mengecek data berhasil di-update buka kembali url http://localhost:8080/student/viewall. Data nama Student telah update menjadi mega cs seperti ditunjukkan gambar berikut ini.



All Students

No. 1

NPM = 1606954943

Name = mega cs

GPA = 3.5

Delete Data
Update Data

No. 2

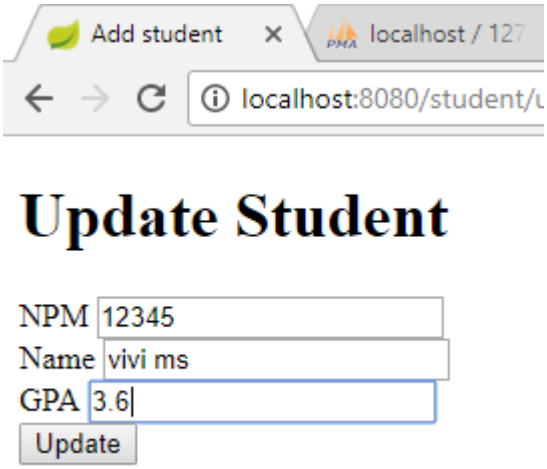
NPM = 12345

Name = vivi

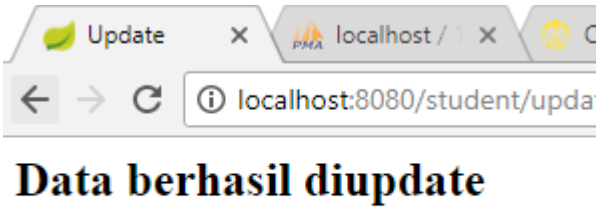
GPA = 3.8

Delete Data
Update Data

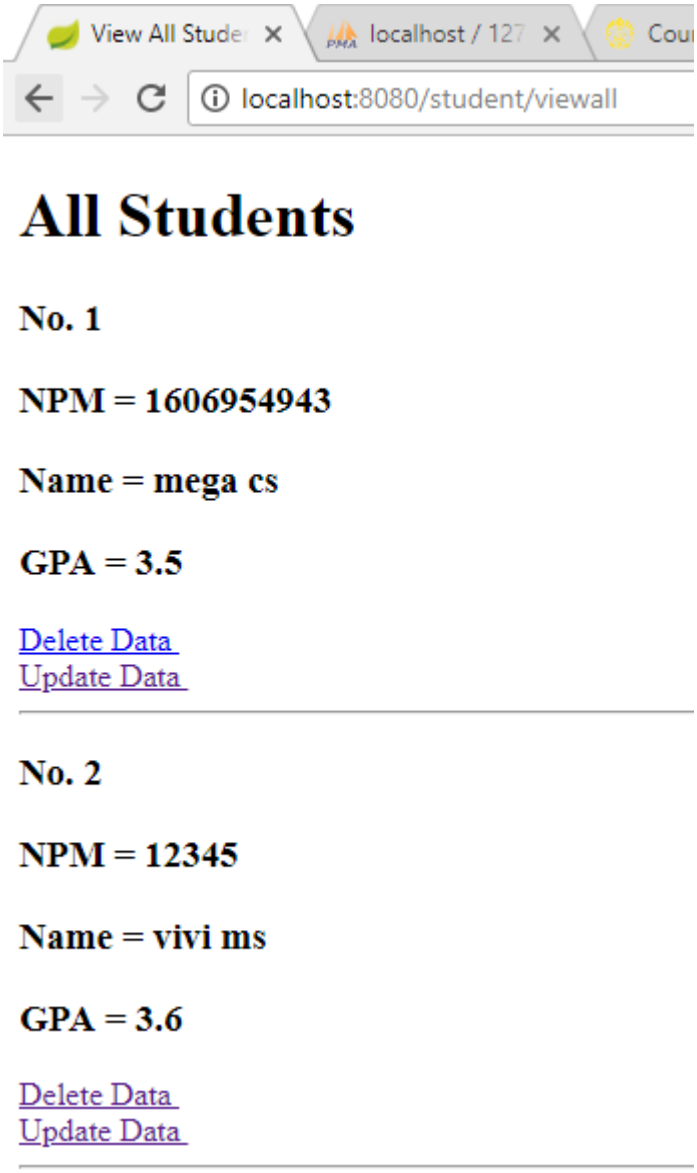
- Meng-update gpa Student dengan npm 12345 yang sebelumnya 3.8 menjadi 3.6 dan nama yang sebelumnya vivi menjadi vivi ms di form-update.html



- Klik tombol Update maka akan ditampilkan view success-update.html seperti gambar berikut.



- Untuk memastikan data telah berhasil di-update buka kembali url http://localhost:8080/student/viewall. Nama dan gpa Student telah berubah seperti ditunjukkan gambar berikut.



Pertanyaan

1. Ya validasi diperlukan. Untuk melakukan validasi di backend bisa menggunakan Hibernate Validator. Hibernate Validation menyediakan API yang tepat yang mengijinkan untuk batasan dalam konfigurasi programmatic. Kita dapat menggunakan anotasi @Valid untuk melakukan validasi seperti contoh berikut.

```
@Valid
Car rentCar(
        @NotNull Customer customer,
        @NotNull @Future Date startDate,
        @Min(1) int durationInDays);
```

2. Form submit biasanya menggunakan post karena beberpa alasan:
    - POST digunakan untuk aksi yang mengubah seperti create, update, delete
    - POST lebih aman dibandingkan dengan GET karena jika menggunakan GET parameter akan ditampilkan di URL. Parameter tersebut dapat tersimpan di cache browser sehingga tingkat keamanannya rendah.

- Post dapat mengirim data yang lebih besar dibandingkan GET

Untuk penanganan di body method tidak perlu dilakukan

3. Ya, mungkin. Kita dapat mendefinisikan suatu method menerima GET dan POST. Misalnya : @RequestMapping(value="/testonly", method={RequestMethod.GET, RequestMethod.POST})
   Jika pemanggilan method di url dengan GET (localhost:8080/perkalian?a=10&b=10) maka akan menggunakan tipe GET. Jika pemanggilan di url dengan POST (http://localhost:8080/student/delete/11112110) maka akan menggunakan tipe POST.