Pance Satria Naibaho
1706106904

- ## Latihan Menambahkan Delete

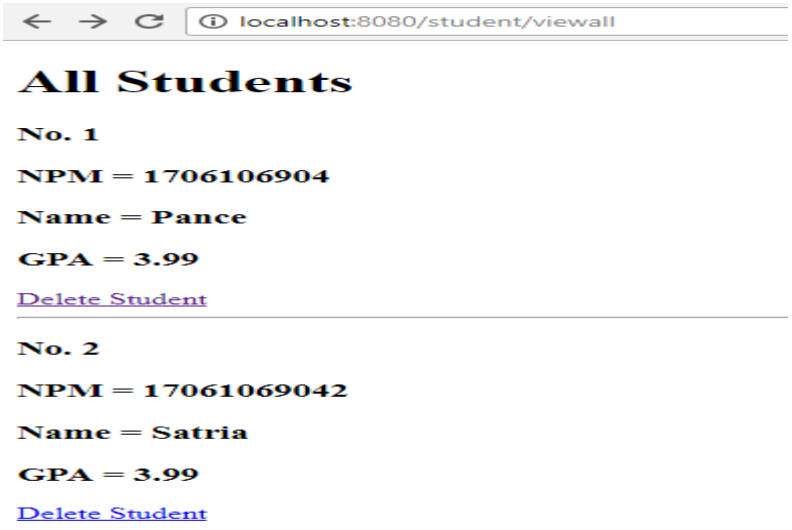
Method ini akan menghapus data student yang ada di database berdasarkan npm yang diberikan. Code yang saya buat untuk method delete:

```
StudentModel student = studentDAO.selectStudent(npm);

  if (student != null) {
      studentDAO.deleteStudent (npm);
      return "delete";
  } else {
      model.addAttribute ("npm", npm);
      return "not-found";
  }
```

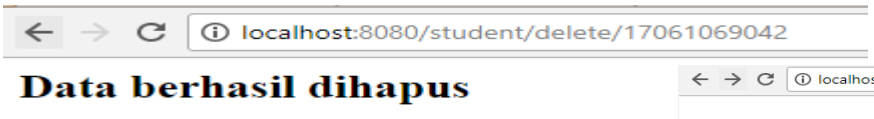
Pertama saya melakukan pengecekan apakah npm yang diberikan ada di database (melalui method *selectStudent*), jika ada maka *student* tidak null dan data student tersebut dihapus melalui *method deleteStudent(npm).* Jika data student tidak ada dalam database maka akan ditampilkan terangan npm tidak ditemukan

Demo:

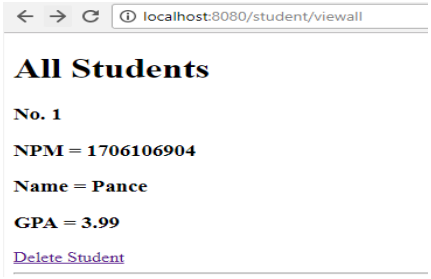
Tampilan viewall



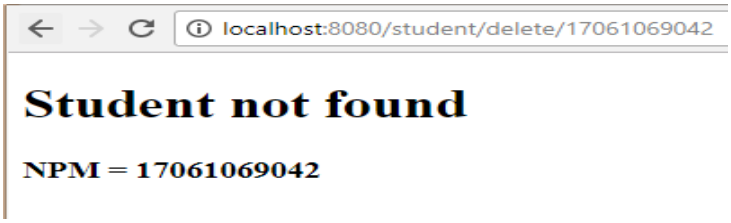
Kita akan menghapus student dengan npm 17061069042



Data Student berhasil dihapus

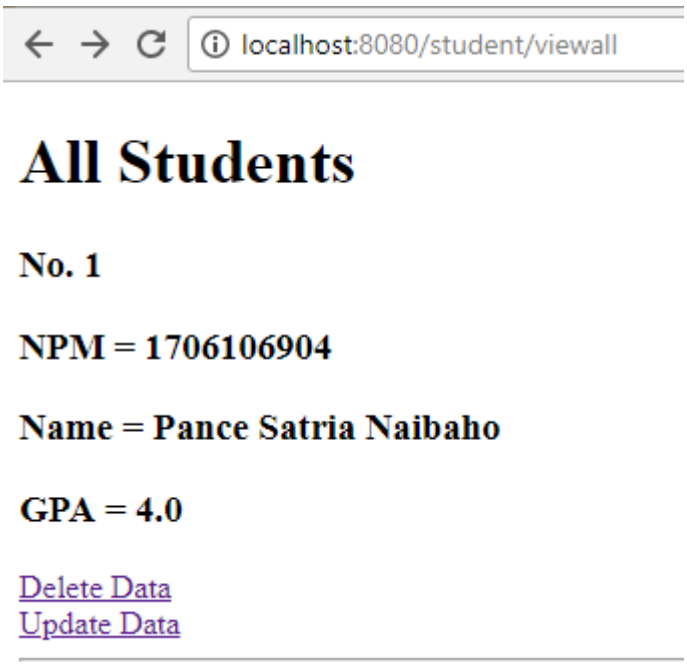


Gambar di bawah ini menunjukkan ketika npm yang akan dihapus tidak ditemukan di database



- **Latihan Menambahkan Update**

Method update digunakan untuk mengubah data student berdasarkan npm.
Method update dapat diakses melalui viewall, klik Update Data



Aksi klik Update Data tersebut akan ditangani oleh code berikut:

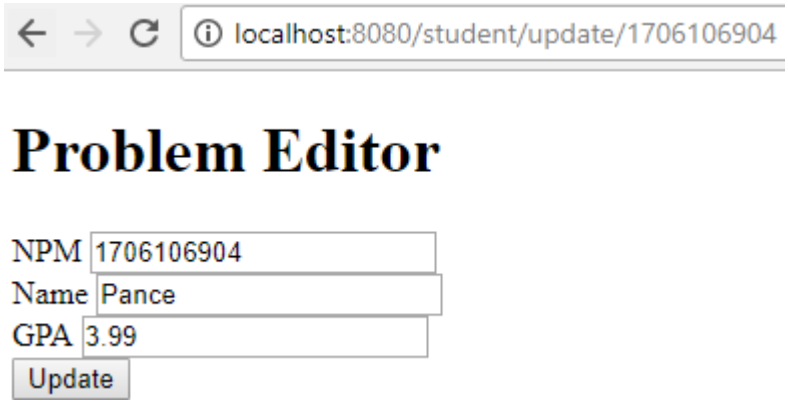
```
@RequestMapping("/student/update/{npm}")
public String update(Model model, @PathVariable(value = "npm") String
npm) {

	StudentModel student = studentDAO.selectStudent(npm);
	if(student != null) {
		model.addAttribute("student", student);
		return "form-update";
	}else
		return "not-found";
	}
```

Jika student dengan npm yang diberikan ada di database akan menampilkan form update dengan field-field (npm, name, gpa) sudah otomatis terisi otomatis.

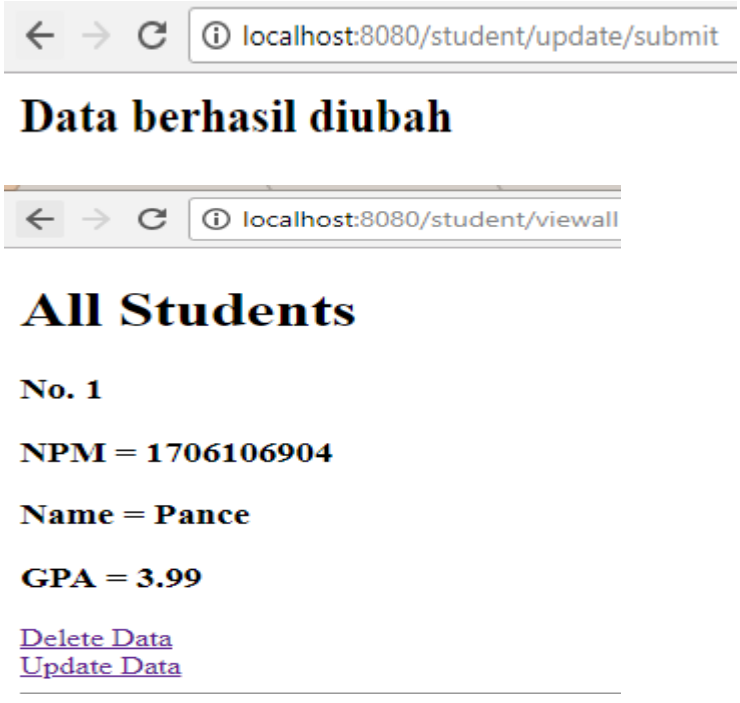
Contoh tampilan update data:

Pance Satria Naibaho
1706106904

Ubah name dan gpa sesuai yang diinginkan seperti pada gambar di bawah, dan klik tombol update



Tampilan data berhasil diubah:



Pance Satria Naibaho
1706106904

- ## Menggunakan Object Sebagai Parameter

  Menggunakan objek sebagai parameter untuk menangani form submit sangat efisien. Kita tidak perlu menuliskan semua field sebagai parameter. Berikut code yang saya gunakan pada method updateSubmit (method ini belum ada validasi) dengan menggunakan objek StudentModel sebagai parameter.

```
@RequestMapping("/student/update/submit")
public String updateSubmit(@ModelAttribute StudentModel student) {

	studentDAO.updateStudent(student);

	return "success-update";
}
```

Jawaban Pertanyaan

1. Untuk validasi input di backend saya menggunakan Bean Validation. Cara penggunaannya cukup sederhana, jika field yang saya inginkan tidak boleh null (required) maka tinggal diberikan anotasi @NotNull pada variable yang ada di model (StudentModel), jika field optional maka tidak perlu diberikan anotasi apapun. Contoh code nya:

```
@NotNull
private String npm;

@NotNull
@Size(min=2, max=30)
private String name;

@NotNull
@Min(1)@Max(4)
private Double gpa;
```

2. Mengapa form submit biasanya menggunakan POST method dibanding GET method? Karena form submit itu gunanya untuk menerima inputan dari user serta mengirim data tersebut ke server web dan method yang bisa menangani hal tersebut adalah method POST sedangkan method GET digunakan untuk mengambil data dari web server berdasarkan parameter yang diberikan.

3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST? Mungkin, contoh pengimplementasiannya di header seperti ini:

   @RequestMapping(“/student”, method = {RequestMethod.GET, RequestMethod.POST})

Pance Satria Naibaho
1706106904

Ringkasan

Tutorial ini bertujuan membuat aplikasi web yang sudah terhubung dengan database local. Ada beberapa library baru yang ditambahkan di project ini dibanding dengan tutorial-tutorial sebelumnya, yaitu Lombok, MyBastis, dan MySQL. Library Lombok digunakan untuk helper annotation, library MyBastis memiliki fungsi untuk menghubungkan project dengan MySQL. MyBastis digunakan untuk melakukan koneksi dan generate query dengan helper annotation.

Pada tutorial ini saya mempelajari bagaimana menangani form submit dengan 2 cara yaitu dengan menggunakan RequestParam dan menggunakan objek sebagai parameter. Menggunakan objek sebagai parameter lebih efisien dibanding RequestParam.

Selain itu saya juga mempelajari bagaimana menangani validasi form jika menggunakan objek sebagai parameter. Cara yang saya lakukan adalah dengan menggunakan Bean Validation seperti yang sudah dijelaskan pada jawaban pertanyaan nomor satu.