DINA FEBRIANTI KRISNA
1506757421

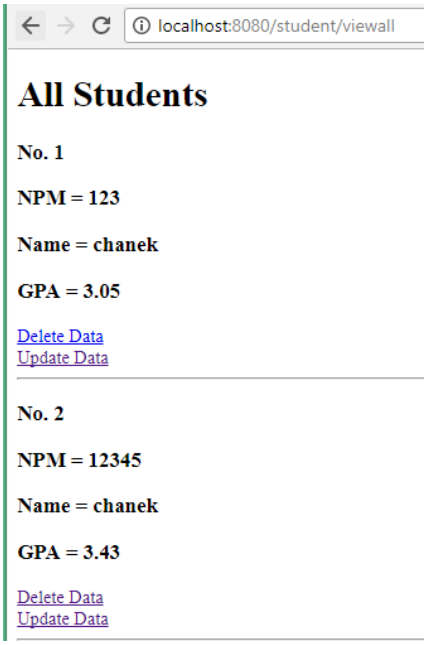
TUTORIAL 4
KELAS B

Pada tutorial kali ini, program yang dibuat dari tutorial sebelumnya dikoneksikan ke database local. Selain itu, tutorial juga menjelaskan cara melakukan debugging melalui spring boot.

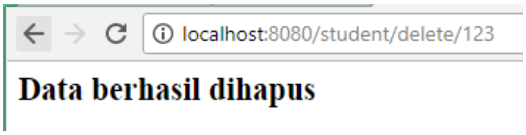
Latihan Menambahkan Delete

Proses delete:

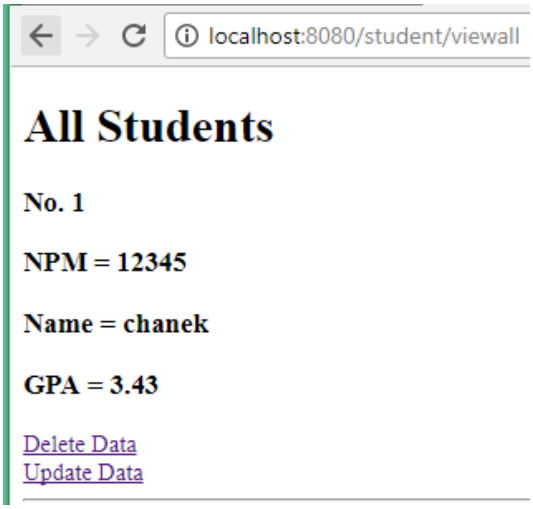
1. Klik tombol delete pada tampilan viewall students



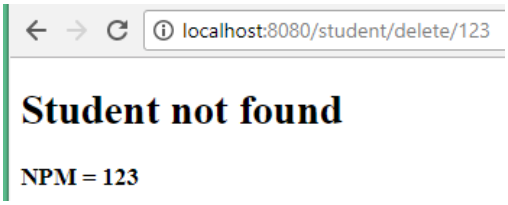
2. Pesan bahwa data berhasil dihapus



3. Kondisi halaman viewall setelah salah satu data dihapus



4. Mengakses halaman delete/123 setelah mahasiswa dengan npm tersebut dihapus



Untuk menangani permintaan delete, dibutuhkan method delete pada StudentController untuk melakukan mapping URL ke method yang benar. Apabila student dengan npm yang dimaksud, maka pemrosesan delete dilanjutkan.

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

    }
```

Apabila student dengan npm yang diminta berhasil ditemukan, maka method deleteStudent yang berada pada class StudentServiceDatabase akan dipanggil untuk mengakses method deleteStudent pada class StudentMapper.

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

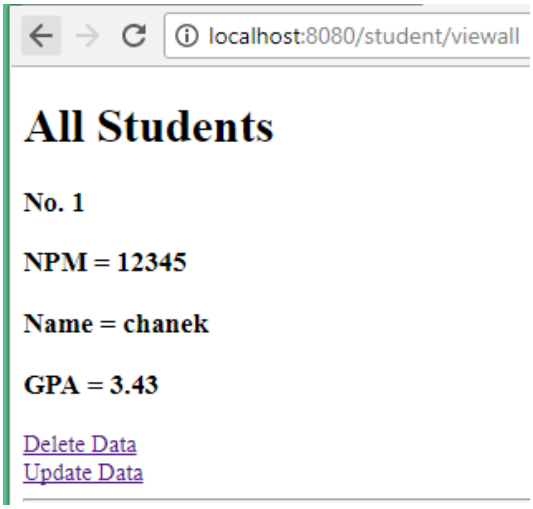
Method deleteStudent pada StudentMapper akan menghapus data dari database.

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

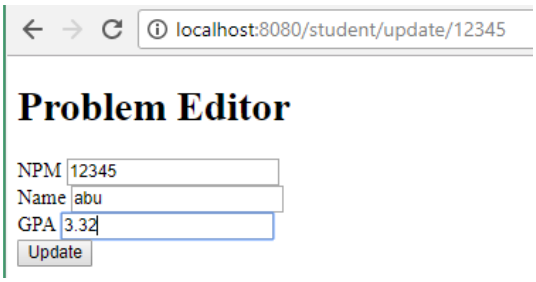
Latihan Menambahkan Update

Proses update:

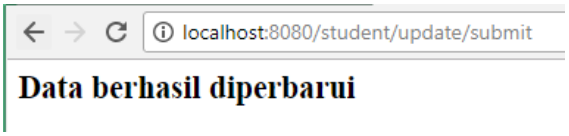
1. Klik Update Data pada halaman viewall students



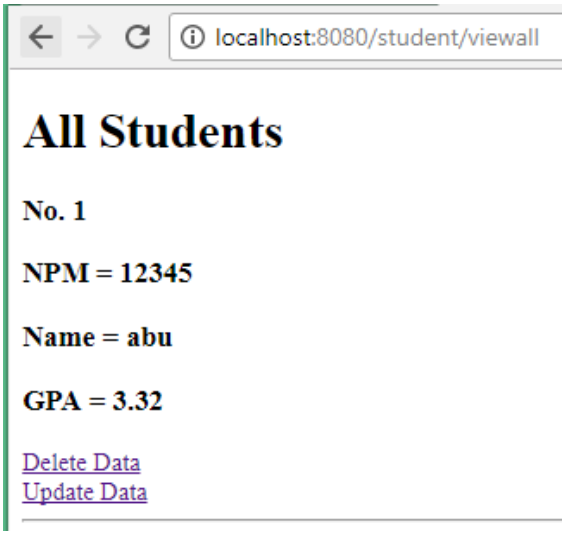
2. Mengupdate field nama dan/atau gpa



3. Pesan bahwa update berhasil



4. Perubahan detail mahasiswa setelah dilakukan update



Untuk menangani permintaan update, dibutuhkan method update pada StudentController untuk melakukan mapping URL ke method yang benar.

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

    }
```

Apabila student dengan npm yang diminta berhasil ditemukan, maka pengguna akan dialihkan ke halaman submit. Pada method updateSubmit akan dipanggil method updateStudent pada class StudentServiceDatabase.

```
    @RequestMapping(value = "/student/update/submit", method =
RequestMethod.POST)
    public String updateSubmit(StudentModel student)
    {
      StudentModel student1 = studentDAO.selectStudent (student.getNpm());
      student1.setGpa(student.getGpa());
      student1.setName(student.getName());
      studentDAO.updateStudent(student);
      return "success-update";
    }
```

Selanjutnya, method updateStudent akan memanggil method updateStudent pada class StudentMapper

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

Method updateStudent pada class StudentMapper akan melakukan penghapusan data yang dimaksud

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

Latihan Method Menggunakan Object sebagai Parameter

Form yang dikirimkan ke halaman submit menggunakan method POST. Method updateSubmit() menerima StudentModel object yang di-retrieve dari database berdasarkan input npm. Object yang diterima pada parameter dijadikan input untuk melakukan update object dengan npm yang sama dari database yang ada.

```
    @RequestMapping(value = "/student/update/submit", method =
    RequestMethod.POST)
```

```
public String updateSubmit(StudentModel student)
{
  StudentModel student1 = studentDAO.selectStudent (student.getNpm());
  student1.setGpa(student.getGpa());
  student1.setName(student.getName());
  studentDAO.updateStudent(student);
  return "success-update";
}
```

Jawaban

1. Validasi *required field* pada *backend* dilakukan dengan menambahkan anotasi *not null* pada setiap *instance variable* suatu model. Java telah menyediakan *package* untuk menangani hal ini, yaitu *package* javax.validation.constraints.NotNull

2. *Form submit* biasanya melakukan perubahan pada *database*, seperti operasi *delete*, *insert* dan *update*. *Method* POST memiliki keamanan yang lebih karena tidak bisa diakses langsung melalui URL, berbeda dengan *method* GET yang dapat diakses melalui URL sehingga *user* bisa memodifikasi data melalui URL. Dengan demikian, menggunakan *method* POST dirasa lebih aman. Tidak perlu penanganan berbeda pada *method controller* untuk kedua *method* tersebut

3. Mungkin, dengan menjadikan *method* pada anotasi *request mapping* sebagai suatu *array* (e.g : method = {RequestMethod.POST, RequestMethod.GET})