PERTANYAAN

1. Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required? Apakah validasi diperlukan?Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.
   Jawab : Validasi dapat dilakukan di Controller dengan memeriksa masing-masing nilai pada object yang dikirimkan. Validasi penting untuk memastikan data yang dimasukan sesuai dengan database dan logika pemrograman yang dibuat
2. Menurut Anda, mengapa form submit biasanya menggunakan POST method disbanding GET method? Apakah perlu penanganan berbeda di header atau body method di controller jika form di post dikirim menggunakan method berbeda?
   Jawab : POST lebih aman daripada GET karena data yang dikirimkan tidak mudah dibaca oleh user secara langsung. Untuk penangan di header atau body method controller perlu penanganan berbeda untuk jenis method form yang berbeda
3. Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?
   Jawab : Mungkin.

- Method yang Anda buat pada Latihan Menambahkan Delete, jelaskan

StudentMapper.java

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

StudentServiceDatabase.java

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

StudentController.java

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

    return "delete";
  }
```

Penjelasan :

Request user dihandle pada **StudentController.java** . Dengan menerima parameter npm, method pada controller kemudian memastikan apakah terdapat data siswa dengan npm tersebut. Jika tidak tersedia akan mengembalikan halaman not found. Sementara jika ditemukan, maka akan mengembalikan halaman delete. Sebelum mengembalikan halaman delete, program akan mengeksekusi method delete student dengan query sesuai yang tertera pada studentMapper.java

- Method yang Anda buat pada Latihan Menambahkan Update, jelaskan

StudentMapper.java

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

StudentServiceDatabase.java

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

StudentController.java

```
@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 students = new StudentModel(npm, name, gpa);
        studentDAO.updateStudent(students);
        return "success-update";
  }
```

Penjelasan :

Request user dihandle pada **StudentController.java** . Dengan menerima parameter npm, gpa dan name. method pada controller kemudian menghidupkan object StudentModel sesuai parameter. Kemudian akan mengeksekusi method updateStudent dengan parameter object yang sebelumnya dihidupkan. Method tersebut akan mengeksekusi query sesuai pada **StudentMapper.java**. Setelah selesai akan mengembalikan halaman success-update.

- Method yang Anda buat pada Latihan Menggunakan Object Sebagai Parameter, jelaskan

StudentMapper.java

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

StudentServiceDatabase.java

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

StudentController.java

```
@RequestMapping(value = "/student/update/submit", method= RequestMethod.POST)
  public String updateSubmit(StudentModel students) {
        studentDAO.updateStudent(students);
        return "success-update";
  }
```

Penjelasan :

Perbedaan hanya terletak pada controller dimana objek yang menjadi parameter berupa object StudentModel. Parameter tersebut didapatkan dari halaman view yang terlebih dahulu mengirimkan object studentModel. Parameter tersebut kemudian digunakan untuk memanggil method updateStudent dan mengeksekusi query yang sama. Kemudian mengembalikan halaman success-update