Nama : Radhitya Rahman
Kelas : 1406623335

Banyak hal yang saya pelajari dari tutorial ini, yaitu penggunaan banyak anotasi untuk menjalankan command sql seperti select, insert, delete, dan update. Kemudian anotasi @GetMapping dan @PostMapping kemudian @ModelAttribute yang berguna untuk mengembalikan objek tertentu.

Pertanyaan:

1. Cara saya untuk melakukan validasi input dapat dilakukan di backend dengan cara menambahkan decision flow di dalam method updateSubmit seperti: if(student.getNama atau getNpm atau getBlabla dst tidak null atau panjang dari student.getNama tidak melebihi batas tertentu akan melanjutkan updateStudent dan mengembalikan halaman success-update. Intinya validasi yang saya ajukan tidak berada di parameter tetapi di isi method-nya.

   Sebagai tambahan, saya juga mengajukan iteration flow untuk mengatasi requiered input yg dibutuhkan. Misalkan jika student.getBlabla == NULL maka akan memanggil method update untuk menjalankan form-update secara berulang.

   Untuk pertanyaan apakah validasi diperlukan perlu atau tidak? Menurut saya validasi penting untuk mengatasi program agar tidak menuju ke error flow, akan tetapi semua kepentingan ini balik lagi ke si programmer apakah ingin ada catch error atau tidak. Selain itu tujuan dari validasi juga dapat mencegah buffer overflow di mana hal ini akan menimbulkan celah keamanan informasi yaitu kondisi di mana setiap bit dari data yang ditransfer untuk melakukan pengecekan password(misalkan jika password salah, maka variable yang mengatasi checking validasi akan mengembalikan nilai=0 dan sebaliknya) akan dapat menimpa binary dari variable selanjutnya di mana akan berdampak pada skip variable, input nilai=0 tsb dapat menjadi nilai!=0 sehingga password akan selalu dalam kondisi true. Buffer overflow terjadi jika password yang di-input melebih dari batas maksimal karakter pemasukannya.

   Tambahan lainnya yang lebih sederhana, agar dapat dimasukan ke dalam database.
2. Alasannya sederhana, karena dengan menggunakan POST submit tidak menampilkan data yg diinput pada URL sehingga dapat terhindar dari serangan luar yang melihat pengiriman URL tersebut.
3. Tidak mungkin karena akan terjadi bentrok nilai input

Menambahkan Delete:

- StudentController: hal pertama yg dilakukan adalah mengecek apakah student yang ingin dihapus ada atau tidak dengan mengecek npm yg ditulis di URL dan menggunakan method selectStudent yang menerima parameter npm. Jika ada maka akan menjalankan method deleteStudent dengan parameter dari npm sebelumnya. Kemudian

menampilkan page yang menampilkan halaman keberhasilan penghapusan file. Jika tidak ada maka akan menampilkan page yang menunjukan bahwa file tidak ditemukan.
- StudentMapper: delete dari table student dengan npm = npm yang diterima dari parameter.
- StudentService: membuat interface untuk deleteStudent dengan parameter npm yang diterima sebelumnya.
- StudentServiceDatabase: membuat log yang menunjukan bahwa student dengan npm tertentu telah dihapus. Kemudian menjalankan deleteStudent untuk class studentMapper.

Menambahkan Update:
- StudentControlerl: hal pertama yg dilakukan adalah mengecek apakah student yang ingin diupdate ada atau tidak dengan mengecek npm yg ditulis di URL dan menggunakan method selectStudent yang menerima parameter npm. Jika ada maka akan menjalankan method updateStudent dengan parameter dari npm sebelumnya. Kemudian menampilkan menuju ke page form-update. Jika tidak ada maka akan menampilkan page yang menunjukan bahwa file tidak ditemukan. Setelah masuk ke form-update maka akan menampilkan halaman form yang menerima input data yang kemudian akan disubmit untuk melakukan update. Setelah melakukan submit akan menjalankan method updateSubmit di mana memiliki mapping dari halaman form ke halaman untuk meng-update submit. Pada method updateSubmit akan menjalankan updateStudent yang menerima parameter objek dari mapping halaman form sebelumnya. Terakhir akan menampilkan halaman success-update.
- StudentMapper: melakukan update di table student untuk parameter name, npm, dan gpa di mana diterima dari parameter objek student.
- StudentService: membuat interface untuk updateStudent dengan parameter objek student yang diterima sebelumnya
- StudentServiceDatabase: membuat log yang menunjukan bahwa student dengan npm tertentu telah di-update. Di sini karena parameter berupa objek student maka untuk menampilkan npm dari student terkait menggunakan getNpm(). Kemudian menjalankan deleteStudent untuk class studentMapper.

Menggunakan Objek sebagai parameter:

Menggunakan anotasi @PostMapping untuk menerima objek yang di-set di form-update pada attribute form (th:object=”${student}”) di mana objek student sebelumnya dikirim dari method update dan menambahkan attribute dengan nama student. Kemudian menggunakan anotasi @ModelAttribute untuk meng-assign value dengan objek yang dikirim dari form sebelumnya.