Alex Fransiscus Manihuruk
1706106614

Delete

Code yang ditambahkan:

- Pada *StudentMapper.java*

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

- Pada *StudentService.java*

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

- Pada *StudentServiceDatabase.java*

```
@Override
      public void deleteStudent (String npm){
         studentMapper.deleteStudent(npm);
}
```

StudentMapper digunakan untuk berinteraksi dengan database. *Query* database yang dipakai delete. *StudentService* merupakan interface yang mendefinisikan *method-method* apa saja yang dapat dilakukan untuk memanipulasi kelas *Student. StudentServiceDatabase* adalah *service* yang meng-*implements StudentService. Parameter* yang dipakai adalah *npm* yang bertipe data *String.*

Update

Code yang ditambahkan:

- Pada *StudentMapper.java*

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

- Pada *StudentService.java*

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

- Pada *StudentServiceDatabase.java*

```
@Override
public void updateStudent(StudentModel student) {
      studentMapper.updateStudent(student);
}
```

StudentMapper digunakan untuk berinteraksi dengan database. *Query* database yang dipakai update. *StudentService* merupakan interface yang mendefinisikan *method-method* apa saja yang dapat dilakukan untuk memanipulasi kelas *Student. StudentServiceDatabase* adalah *service* yang meng-*implements StudentService. Parameter* yang dipakai adalah `StudentModel` yang bertipe data student.

Alex Fransiscus Manihuruk
1706106614

Object As Paramater

Code yang ditambahkan pada *StudentController*

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

Ketika menggunakan object sebagai parameter maka parameter berupa objek saja pada service dan mapper sehingga tidak perlu membuat setiap field menjadi parameter.

Q & A

1) Q : Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required seperti jika menggunakan RequestParam? Apakah validasi diperlukan? Asumsikan input pada form Anda tidak menggunakan attribute required sehingga butuh validasi di backend.

A : Validasi dapat dilakukan dibackend dengan mengecek apakah object yang dikirim bernilai null atau tidak. Untuk pengecekannya tergantung pada konsep yang dipakai, bisa di condtroler ataupun model.

2) Q : Menurut Anda, mengapa form submit biasanya menggunakan POST method dibanding GET method? Apakah perlu penanganan berbeda di header atau body method di controller jika form di post dikirim menggunakan method berbeda?

A : Karena POST method lebih aman dimana dengan cara mengirimkan data atau nilai langsung ke action untuk ditampung, tanpa menampilkan pada URL seperti halnya GET method.

3) Q : Apakah mungkin satu method menerima lebih dari satu jenis request method, misalkan menerima GET sekaligus POST?

A : Tidak, karena di awal ditentukan method yang dipakai