Gitta Gracia Evelyn Diva 1406544135 PSP - B

PERTANYAAN

1. Validasi dapat dilakukan menggunakan annotasi @Valid dan kelas BindingResult, BindingResult akan melakukan pengecekkan apakah terjadi error atau tidak, jika objek yang dikirimkan memiliki error maka kembalikan halaman view error, namun jika tidak, akan meneruskan eksekusi lainnya

2. Alasan mengapa form submit biasanya menggunakan POST method, dikarenakan data dikirimkan melalui JSON sehingga lebih aman, apabila menggunakan GET method, maka data akan dikirimkan melalui URL sehingga tidak aman. Oleh karena itu, GET method biasanya digunakan untuk mengirimkan ID atau data yang dibutuhkan untuk menampilkan suatu webpage, bukan data yang akan disimpan pada data store.

3. Sebuah method menerima GET dan POST sekaligus.
   Tidak bisa, karena pada formulir html hanya dapat mengeksekusi 1 method.

LATIHAN IMPLEMENTASI METHOD

- Method Delete

```
@RequestMapping("/student/delete/{npm}")
public String delete (Model model, @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent(npm);

    if(student == null){
        return "not-found";
    }
    studentDAO.deleteStudent (npm);

    return "delete";
}
```

Method menerima parameter berupa NPM, kemudian akan dilakukan pengecekkan apakah student dengan npm yang dicari ada, jika tidak ada maka method akan mengembalikan view not-found.html. Namun jika ada, maka akan dipanggil method deleteStudent dengan parameter NPM di kelas StudentServiceDatabase.java

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

Method tersebut akan memanggil method deleteStudent dengan parameter NPM pada kelas studentMapper untuk menghapus student sesuai npm dari database.

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

- Method Update

```
/** update dengan parameter menggunakan @RequestParam
    @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 student = new StudentModel(npm,name,gpa);

        studentDAO.updateStudent (student);

        return "success-update";
    }
**/
```

Method menerima parameter berupa NPM, nama dan gpa dari student, kemudian akan dilakukan pembuatan objek student secara lokal yang berisikan data yang telah diperbarui user, pembuatan objek ini bukan berarti menambah student baru pada database, namun hanya membuat objek secara lokal. Kemudian akan dipanggil method updateStudent dengan parameter objek student pada kelas StudentServiceDatabase.java

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

Method tersebut akan memanggil method updateStudent dengan parameter student pada kelas mapper untuk memperbarui data student pada database.

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

Pada kelas StudentMapper.java akan dilakukan perbaruan data student pada database dengan mencari terlebih dahulu student dengan npm yang sesuai.

PS: Method dikomen karena saat ini menggunakan method update dengan parameter object

- Method Update dengan parameter Object

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

    return "success-update";
}
```

Alur kerja method ini kurang lebih sama dengan method pada update sebelumnya, bedanya parameter pada method ini menerima objek student yang dikirimkan oleh form-update.html.