Write Up Tutorial 4
Ristya Nintyana
1406574623
APAP - B

Pada tutorial 4 ini, saya mempelajari untuk menghubungkan program dengan database menggunakan Spring, dan mempelajari operasi CRUD yang terhubung pada database.

Pertanyaan:

1. 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.

   Validasi dilakukan dengan mengecek objek yang ada pada parameter apakah nilai-nilai yang diperlukan sudah ada dalam database. Ya, validasi diperlukan apabila data tersebut harus ada misalkan pada atribute yang merupakan primary key

2. 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?

   Method post biasanya digunakan untuk pengiriman data yang bersifat sensitif, karena pada method post data yang dikirimkan tidak tercantum pada url atau link. Sedangkan method get biasanya digunakan untuk pengiriman data yang tidak terlalu sensitif. Ya, butuh penangan yang berbeda. RequestMapping untuk post menggunakan RequestMethod.POST sedangkan method fget menggunakan RequestMethod.GET.

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

   Ya, satu method dapat menerima lebih dari satu jenis request method

- Method yang Anda buat pada Latihan Menambahkan Delete, jelaskan
  Pertama-tama saya menambahkan link pada page viewall.html untuk melakukan operasi delete. Setelah itu saya menambahkan method deleteStudent pada StudentMapper yang berisikan query untuk menghapus data student pada database.

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

Setelah itu melengkapi method deleteStudent pada StudentServiceDatabase menjadi

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

Lalu saya melengkapi method delete pada StudentController sekaligus memberikan validasi apakah npm tersebut terdaftar atau tidak.

```
 @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 {

         return "not-found";
      }

   }
```

- Method yang Anda buat pada Latihan Menambahkan Update, jelaskan
  Pertama-tama saya menambahkan method updateStudent pada class StudentMapper:

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

  Lalu saya menambahkan method updateStudent pada StudentService:

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

  Setelah itu saya menambahkan method updateStudent pada StudentServiceDatabase

  ```
  @Override
     public void updateStudent (StudentModel student)
  ```

```
    {
                log.info ("student " + student + " updated");
                studentMapper.updateStudent(student);
    }
```

Setelah itu saya menambahkan link pada page viewall.html untuk melakukan operasi update student:

```
<a th:href="'/student/update/' + ${student.npm}"> Update Data</a><br/>
```

Lalu saya menambahkan halaman form-update.html yang berisikan form untuk memasukan data update student, dan success-html yang berisikan untuk menunjukkan bahwa update berhasil dilakukan.

Lalu saya menambahkan method pada StudentController beserta validasi apakah npm terdaftar atau tidak.

```
@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 {

            return "not-found";
        }
```

Serta membuat method updateSubmit yang berfungsi untuk melakukan operasi submit setelah data-data dimasukan  pada form-update.html

```
@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 = studentDAO.selectStudent (npm);
                        student.setName(name);
                        student.setGpa(gpa);
```

```
        studentDAO.updateStudent(student);
        return "success-update";
    }
```

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

Pada latihan menggunakan object sebagai Parameter, saya menambahkan th:object="${student}" pada tag <form>, can menambahakan th:field="*{nama_field} untuk input npm name dan gpa . Dan saya mengubah method updateSubmit menjadi

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