MUHAMMAD AKBAR SETIADI
APAP A
1506737810

Yang Saya Pelajari

Yang saya pelajari pada tutorial ini adalah mengenai konfigurasi java spring dengan database mysql yang ada di dalam phpmyadmin, seperti penggunaan insert, update dan delete row pada database. Kemudian saya baru tahu mengenai parameter objek dengan menggunakan *th:object* dan *th:field,* yang menurut saya cukup penting untuk efisiensi *code.*

Jawaban Pertanyaan

1. secara *default,* setiap variabel objek yang diinput berupa optional required (*required* = *false*). Tetapi, kita dapat mengubahnya menjadi *required* = *true,* dengan menambahkan setiap instance variable pada objek yang diinginkan dengan anotasi @NotNull

```
public class StudentModel
{
    @NotNull
    private String npm;
    @NotNull
    private String name;
    @NotNull
    private double gpa;

}
```

   Kemudian pada methodnya ditambahkan @Valid

```
@RequestMapping("/student/update/submit")
public String updateSubmit (
        @Valid @ModelAttribute StudentModel student)
{
    studentDAO.updateStudent(student);
    return "success-update";
}
```

2. Karena data yang dikirimkan oleh method *POST* tidak terbatas dan relatif aman, karena variabel data tidak ditampilkan di URL, seperti pada method *GET*

   Apakah perlu penanganan berbeda di header atau body method di controller jika form di post dikirim menggunakan method berbeda?

   Seperti yang saya pelajari pada tutorial ini, maka perlu untuk *best practice* nya, dengan menambahkan RequestMethod.POST pada @RequestMapping.

3. Tidak bisa, satu method satu jenis request method.

MUHAMMAD AKBAR SETIADI
APAP A
1506737810

Method pada Latihan Menambahkan Delete

1. menambahkan *code* di bawah pada viewall.html di dalam div

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

2. Menambahkan method deleteStudent di class StudentMapper

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

3. Melengkapi method deleteStudent di class StudentServiceDatabase

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

4. Melengkapi method delete di class StudentController

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

    if (student != null) {
        model.addAttribute ("student", student);
        studentDAO.deleteStudent (npm);
        return "delete";
    } else {
        return "not-found";
    }
}
```

5. Mencoba delete npm dengan menjalankan program dan berhasil

Method pada Latihan Menambahkan Update

1. menambahkan method *updateStudent* pada class StudentMapper

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

2. menambahkan method updateStudent pada interface StudentService

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

3. menambahkan implementasi method updateStudent pada class StudentServiceDatabase

MUHAMMAD AKBAR SETIADI
APAP A
1506737810

```
@Override
public void updateStudent (StudentModel student)
{
    Log.info ("student " + student.getNpm() + " updated");
    studentMapper.updateStudent(student);;
}
```

4. Menambahkan link Update Data pada viewall.html

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

5. Membuat file form-update.html dengan mengcopy file form-add.html

StudentController.java | form-update.html | StudentService.java | viewall.html | StudentMapper.java | StudentServiceDatabase.j

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>

<title>Add student</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>

    <h1 class="page-header">Problem Editor</h1>

    <form action="/student/update/submit" method="get" th:object="${student}" >
        <div>
            <label for="npm">NPM</label> <input type="text" name="npm" readonly="true" th:field="*{npm}"/>
        </div>
        <div>
            <label for="name">Name</label> <input type="text" name="name" th:field="*{name}" />
        </div>
        <div>
            <label for="gpa">GPA</label> <input type="text" name="gpa" th:field="*{gpa}" />
        </div>

        <div>
            <button type="submit" name="action" value="save">Save</button>
        </div>
    </form>

</body>

</html>
```

6. Membuat file success-update.html

StudentContro... | form-update.h... | StudentServic...

```
<html>
    <head>
        <title>Update</title>
    </head>
    <body>
        <h2>Data berhasil diubah</h2>
    </body>
</html>
```

7. Menambahkan method update di class StudentController
8. Menambahkan method updateSubmit di class StudentController
9. Menjalankan Spring Boot dan berhasil

MUHAMMAD AKBAR SETIADI
APAP A
1506737810

Method pada Latihan Menggunakan Object

1. Mengubah beberapa tag pada file form-update.html

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>

<title>Add student</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>

    <h1 class="page-header">Problem Editor</h1>

    <form action="/student/update/submit" method="get" th:object="${student}" >
        <div>
            <label for="npm">NPM</label> <input type="text" name="npm" readonly="true" th:field="*{npm}"/>
        </div>
        <div>
            <label for="name">Name</label> <input type="text" name="name" th:field="*{name}" />
        </div>
        <div>
            <label for="gpa">GPA</label> <input type="text" name="gpa" th:field="*{gpa}" />
        </div>

        <div>
            <button type="submit" name="action" value="save">Save</button>
        </div>
    </form>

</body>

</html>

```

2. Mengubah method updateSubmit pada StudentController yang hanya menerima parameter berupa StudentModel

```
}

@RequestMapping("/student/update/submit")
public String updateSubmit (
        @Valid @ModelAttribute StudentModel student)
{
    studentDAO.updateStudent(student);
    return "success-update";
}
```

3. Test aplikasi kembali dan berhasil