Nama: Haryo Parigroho
NPM: 1506689603
Kelas: B

Pertanyaan

1. Cara melakukan validasi input yang optional dan input yang required jika menggunakan Object sebagai parameter pada form POST adalah dengan menggunakan annotation-annotation dari class **javax.validation.constraints.*.**

```
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class StudentModel
{
    @NotNull
    @Size(min=1)
    private String npm;

    @NotNull
    @Size(min=1)
    private String name;

    private double gpa;
```

   Annotation @NotNull dan @Size digunakan jika attribute suatu object tidak boleh null dan tidak boleh kosong (required = true) sedangkan untuk yang optional tidak usah menggunakan annotation.

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

   Lalu tambahkan annotation @Valid dan @ModelAttribute pada parameter Object-nya, tambahkan juga BindingResult sebagai parameter. @Valid digunakan untuk menandakan adanya validation sehingga ketika terdapat error maka result.hasErrors() bernilai true dan mengalihkan ke form update lagi.

2. Form submit biasanya menggunakan method POST karena method POST lebih aman daripada method GET. Form submit yang menggunakan method GET akan menampilkan data-data yang disubmit pada URL dan data-data tersebut juga tersimpan di history browser.

3. Satu method dapat menerima lebih dari satu jenis request method.

Nama: Haryo Parigroho
NPM: 1506689603
Kelas: B

```java
@RequestMapping(value={"/student/add/submit", "/student/add/submitpost"})
public String addSubmit (
        @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.addStudent (student);

    return "success-add";
}
```

/student/add/submitpost adalah mapping untuk form dengan method post.

```html
<form action="/student/add/submitpost" method="post">
    <div>
        <label for="npm">NPM</label> <input type="text" name="npm" />
    </div>
    <div>
        <label for="name">Name</label> <input type="text" name="name" />
    </div>
    <div>
        <label for="gpa">GPA</label> <input type="text" name="gpa" />
    </div>

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

Latihan Menambahkan Delete

Method deleteStudent pada class StudentMapper. Method ini adalah method yang mengeksekusi query pada database. (DELETE FROM student WHERE npm = #{npm})

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

Method deleteStudent pada class StudentServiceDatabase. Method ini memanggil method deleteStudent pada class studentMapper.

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

Method delete pada class StudentController. Method ini menghandle mapping /student/delete/{npm}, jika tidak ada student dengan npm yang ditentukan maka membuka halaman not-found. Tetapi jika terdapat student dengan npm tersebut maka memanggil method deleteStudent ada class StudentServiceDatabase.

Nama: Haryo Parigroho
NPM: 1506689603
Kelas: B

```
@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 {
        model.addAttribute ("npm", npm);
        return "not-found";
    }

}
```

Latihan Menambahkan Update

Method updateStudent pada class StudentMapper. Method ini mengeksekusi query UPDATE student SET npm = #{npm}, name = #{name}, gpa = #{gpa} WHERE npm = #{npm}

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

Method updateStudent pada interface StudentService.

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

Method updateStudent pada class StudentServiceDatabase. Mengoverride dari interface StudentService. Method ini memanggil updateStudent pada class StudentMapper.

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

Method update pada class StudentController. Method ini menghadle mapping /student/update/{npm}, jika student dengan npm yang ditentukan tidak ada maka akan menampilkan halaman not-found. Tetapi jika student ada maka akan menampilkan form untuk update.

```
@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 {
        model.addAttribute ("npm", npm);
        return "not-found";
    }
}
```

Nama: Haryo Parigroho
NPM: 1506689603
Kelas: B

Method updateSubmit pada class StudentController. Setelah mengisi form, method ini akan mensubmit form update. Pada method ini sudah saya tambahkan validasi dengan menggunakan parameter Object StudentModel. Jika validasi gagal maka menampilkan form update lagi tetapi jika validasi berhasil maka menjalankan method updateStudent pada class StudentServiceDatabase.

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

Latihan Menggunakan Object Sebagai Parameter

Method updateSubmit dengan Object sebagai parameter.

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