Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

Write Up Tutorial 4

Latihan Menambahkan Delete

- Method **deleteStudent** pada class **StudentMapper**

  Pada method **deleteStudent** pada class **StudentMapper,** ditambahkan method delete student yang menerima parameter NPM dan menambahkan annotation delete di atas method dan SQL untuk menghapus.

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

  Namun, sebelum menambahkan annotation delete, perlu mengimport annotation tersebut dengan **import** org.apache.ibatis.annotations.Delete; pada class **StudentMapper.**

- Method **deleteStudent** pada class **StudentService**

  Pada method **deleteStudent** pada class **StudentService,** ditambahkan method **delete Student** seperti yang ada di **StudentMapper**.

- Method **deleteStudent** pada class **StudentServiceDatabase**

  Pada method **deleteStudent** pada class **StudentServiceDatabase,** ditambahkan log untuk method tersebut dan memanggil method **deleteStudent** yang ada **di Student Mapper.**

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

Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

- Method **delete** pada class **StudentController**

  Pada method **deleteStudent** pada class **StudentController,** ditambahkan validasi agar jika mahasiswa tidak ditemukan tampilkan view not-found dan menambahkan method yang berfungsi untuk menhapus objek student.

```
@RequestMapping("/student/delete/{npm}")
public String delete (Model model, @PathVariable(value = "npm") String npm)
{
    if(studentDAO.selectStudent (npm) == null) {
        return "not-found";
    }
    studentDAO.deleteStudent (npm);

    return "delete";
}
```

  Sebelum run project, perlu untuk membuat database dan halaman delete.html, dan jika diperlukan, ubahlah username dan password pada data source application properies.

Latihan Menambahkan Update

- Method **updateStudent** pada class **StudentMapper**

  Pada method **updateStudent** pada class **StudentMapper,** ditambahkan method **updateStudent** yang menerima parameter NPM, nama, dan ipk dan menambahkan annotation update di atas method dan SQL untuk mengubah.

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

  Namun, sebelum menambahkan annotation update, perlu mengimport annotation tersebut dengan **import** org.apache.ibatis.annotations.Update; pada class **StudentMapper.**

- Method **updateStudent** pada class **StudentService**

  Pada method **updateStudent** pada class **StudentService,** ditambahkan method **updateStudent** seperti yang ada di **StudentMapper**.

Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

- Method **updateStudent** pada class **StudentServiceDatabase**

  Pada method **updateStudent** pada class **StudentServiceDatabase,** ditambahkan log untuk method tersebut dan memanggil method **updateStudent** yang ada **di Student Mapper.**

```
@Override
public void updateStudent (String npm, String name, double gpa)
{
    log . info ( "student " + npm + " updated" );
    studentMapper.updateStudent(npm, name, gpa);
}
```

- Method **update** pada class **StudentController**

  Pada method **updateStudent** pada class **StudentController,** ditambahkan request mapping ke /student/update/{npm}, lalu tambahkan validasi agar jika mahasiswa tidak ditemukan tampilkan view not-found dan menambahkan method yang berfungsi untuk mengubah objek student.

```
@RequestMapping("/student/update/{npm}")
public String update (Model model, @PathVariable(value = "npm") String npm)
{
    if(studentDAO.selectStudent (npm) == null) {
        return "not-found";
    }

    StudentModel student = studentDAO.selectStudent (npm);

    model.addAttribute("student", student);

    return "form-update";
}
```

- Method **updateSubmit** pada class **StudentController**

  Pada method **updateSubmit** pada class **StudentController,** perlu mengubah bentuk request mapping seperti di gambar di bawah ini untuk menerima request method POST dari view form-update yang menerima parameter npm, name dan gpa. Lalu akan berpindah ke halaman notifikasi bahawa update yang dilakukan telah sukses

```
@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)
{
    studentDAO.updateStudent (npm, name, gpa);
    return "success-update";
}
```

  Sebelum run project, perlu untuk membuat halaman form-update.html dan success-update.html, dan jika diperlukan, ubahlah username dan password pada data source application properies.

Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

Latihan Menggunakan Object Sebagai Parameter

- Method **updateStudent** pada class **StudentMapper**

  Pada method **updateStudent** pada class **StudentMapper,** ditambahkan method **updateStudent** yang menerima parameter **StudentModel** dan menambahkan annotation update di atas method dan SQL untuk mengubah.

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

  Namun, sebelum menambahkan annotation update, perlu mengimport annotation tersebut dengan **import** org.apache.ibatis.annotations.Update; pada class **StudentMapper.**

- Method **updateStudent** pada class **StudentService**

  Pada method **updateStudent** pada class **StudentService,** ditambahkan method **updateStudent** seperti yang ada di **StudentMapper**.

- Method **updateStudent** pada class **StudentServiceDatabase**

  Pada method **updateStudent** pada class **StudentServiceDatabase,** ditambahkan log untuk method tersebut dan memanggil method **updateStudent** seperti yang ada **di Student Mapper.**

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

- Method **update** pada class **StudentController**

  Pada method **updateStudent** pada class **StudentController,** ditambahkan request mapping ke /student/update/{npm}, lalu tambahkan validasi agar jika mahasiswa tidak ditemukan tampilkan view not-found dan menambahkan method yang berfungsi untuk mengubah objek student.

Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

```
@RequestMapping("/student/update/{npm}")
public String update (Model model, @PathVariable(value = "npm") String npm)
{
    if(studentDAO.selectStudent (npm) == null) {
        return "not-found";
    }

    StudentModel student = studentDAO.selectStudent (npm);

    model.addAttribute("student", student);

    return "form-update";
}
```

- Method **updateSubmit** pada class **StudentController**

  Pada method **updateSubmit** pada class **StudentController,**.

```
@RequestMapping("/student/update/submit")
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";
}
```

  Sebelum run project, perlu untuk membuat halaman form-update.html dan success-update.html, dan jika diperlukan, ubahlah username dan password pada data source application properies.

Menjawab Pertanyaan

Jika menggunakan Object sebagai parameter pada form POST, bagaimana caranya melakukan validasi input yang optional dan input yang required? Apakah validasi diperlukan?

A: Cara melakukan validasi input yang optional dan required dengan cara membandingkan input POST dengan database dan keduanya dibandingkan berdasarkan input yang berada di database. Validasi diperlukan untuk menghandling jika input tidak sesuai dengan data yang berada di database.

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 tidak menampilkan input pengguna di url . sifat ini membuat POST method lebih aman dibandingkan dengan method GET yang menampilkan input pengguna di url. perlu penangan berbeda untuk beberapa kasus.

Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

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

A: Memungkinkan untuk satu method menerima lebih dari satu jenis request method. Pada Spring, cukup menggunakan “method = {RequestMethod.POST, RequestMethod.GET}” pada request mapping.