

Penjelasan MVC

Sebelum memulai mengerjakan tugas, saya terlebih dahulu melakukan perbaikan pada application properties agar database terhubung dengan tabel pada siduk.sql(tugas1). Setelah itu membuat model untuk semua tabel beserta variable dan paramaternya.

1. Tampilkan Data Penduduk Berdasarkan Nik

Untuk Mengerjakan tugas ini yang saya lakukan adalah membuat mapping sesuai dengan permintaan soal. Kemudian membuat mapper select untuk mengambil data penduduk melalui nik.

```
@Select("select id, nik, nama, status_dalam_keluarga, tempat_lahir, id_keluarga, tanggal_lahir, jenis_kelamin, golongan_darah, agama, " +
"from penduduk where nik = #{nik}")
@Results(value = { @Result(property = "nik", column = "nik"), @Result(property = "nama", column = "nama"),
@Result(property = "tempat_lahir", column = "tempat_lahir"), @Result(property = "tanggal_lahir", column = "tanggal_lahir"),
@Result(property = "jenis_kelamin", column = "jenis_kelamin"), @Result(property = "golongan_darah", column = "golongan_darah"),
@Result(property = "agama", column = "agama"), @Result(property = "is_wni", column = "is_wni"),
@Result(property = "status_perkawinan", column = "status_perkawinan"),
@Result(property = "id", column = "id"),
@Result(property = "status_dalam_keluarga", column = "status_dalam_keluarga"),
@Result(property = "pekerjaan", column = "pekerjaan"), @Result(property = "is_wafat", column = "is_wafat"),
@Result(property = "id_keluarga", column = "id_keluarga") })
PendudukModel selectpenduduk(@Param("nik") String nik);

@Select("select id,id_kelurahan, alamat, rt, rw " +
"from keluarga where id = #{id_keluarga}")
@Results(value = { @Result(property = "alamat", column = "alamat"),
@Result(property = "rt", column = "rt"),
@Result(property = "rw", column = "rw") })
KeluargaModel selectkeluarga(@Param("id_keluarga") String id_keluarga);

@Select("select * " +
"from kecamatan where id = #{id_kecamatan}")
@Results(value = { @Result(property = "nama_kecamatan", column = "nama_kecamatan"),
@Result(property = "id", column = "id") })
KecamatanModel selectkecamatan(@Param("id_kecamatan") String id_kecamatan);

@Select("select id,nama_kota " +
"from kota where id = #{id_kota}")
@Results(value = { @Result(property = "nama_kota", column = "nama_kota") })
KotaModel selectkota(@Param("id_kota") String id_kota);
```

Ini adalah mapper yang digunakan untuk mengerjakan soal no 1. Kemudian baru saya membuat service database dimana berisi logic untuk menghubungkan tabel. Ini adalah fungsi dari servicedatabase yang digunakan untuk mengerjakan soal 1. Setelah itu saya membuat service untuk menentukan fungsi apa saja yang diperlukan oleh servicedatabase

```
//soal 1
@Override
public PendudukModel selectpenduduk (String nik) {
    PendudukModel penduduk = pendudukMapper.selectpenduduk(nik);
    KeluargaModel keluarga = pendudukMapper.selectkeluarga(penduduk.getId_keluarga());
    KelurahanModel kelurahan = pendudukMapper.selectkelurahan(keluarga.getId_kelurahan());
    KecamatanModel kecamatan = pendudukMapper.selectkecamatan(kelurahan.getId_kecamatan());
    KotaModel kota = pendudukMapper.selectkota(kecamatan.getId_kota());
    penduduk.setKeluarga(keluarga);
    keluarga.setKelurahan(kelurahan);
    kelurahan.setKecamatan(kecamatan);
    kecamatan.setKota(kota);
    return penduduk;
}
```

Baru setelah itu membuat tampilan, tampilan yang digunakan hanya sekedar untuk mengetahui bahwa fungsi berjalan atau tidak.

SiPenduduk

Home

List Penduduk

Tambah Penduduk

Tambah Keluarga

Search

Home

Masukan NIK

Cari Penduduk

Masukan NKK

Cari Keluarga

Gambar diatas adalah fitur setelah menerapkan bootstrap dengan cara fragmentasi. Setelah cari penduduk maka akan keluar tampilan sebagai berikut.

Lihat Data Penduduk - 3101015405170003

NIK 3101015405170003

Nama Heru Haryanto

Tempat/Tanggal lahir Jakarta,2017-05-14

Alamat Ds. Adisumarmo No. 43

RT/RW 079/025

Kelurahan/Desa PULAU TIDUNG

Kecamatan KEPULAUAN SERIBU SELATAN

Kota KABUPATEN KEPULAUAN SERIBU

Golongan Darah A+

Agama Islam

Status Perkawinan Belum Kawin

Pekerjaan BELUM/TIDAK BEKERJA

WNA

Status Kematian Hidup

Ubah Status Kematian

Berikut adalah fragments serta index html, dan viewPenduduk yang digunakan untuk soal 1

```
3 <head>
4 <title>View Penduduk by NIK</title>
5 <link rel="stylesheet" href="/css/bootstrap.min.css" />
6 <link rel="stylesheet" href="https://cdn.datatables.net/v/dt-1.10.16/datatables.min.css" />
7 <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
8 <script src="https://cdn.datatables.net/1.10.7/js/jquery.dataTables.min.js"></script>
9 <script type="text/javascript" src="/js/bootstrap.min.js"></script>
10 <script type="text/javascript">
11 $(document).ready(function() {
12     $('#table').DataTable();
13 });
14 </script>
15 </head>
16 <body>
17 <div th:replace="fragments/fragment :: header"></div>
18 <h2 th:text="'Lihat Data Penduduk - ' + ${penduduk.nik}"></h2>
19 <h3 th:text="'NIK ' + ${penduduk.nik}">Penduduk NIK</h3>
20 <h3 th:text="'Nama ' + ${penduduk.nama}">Penduduk Name</h3>
21 <h3 th:text="'Tempat/Tanggal lahir ' + ${penduduk.tempat_lahir} + ', ' + ${penduduk.tanggal_lahir}">Penduduk BirthDate</h3>
22 <h3 th:text="'Alamat ' + ${penduduk.keluarga.getAlamat()}">Penduduk Alamat</h3>
23 <h3 th:text="'RT/RW ' + ${penduduk.keluarga.getRt() + '/' + ${penduduk.keluarga.getRw()}">Penduduk Religion</h3>
24 <h3 th:text="'Kelurahan/Desa ' + ${penduduk.keluarga.kelurahan.getNama_kelurahan()}">Penduduk Religion</h3>
25 <h3 th:text="'Kecamatan ' + ${penduduk.keluarga.kelurahan.kecamatan.getNama_kecamatan()}">Penduduk Religion</h3>
26 <h3 th:text="'Kota ' + ${penduduk.keluarga.kelurahan.kecamatan.kota.getNama_kota()}">Penduduk Religion</h3>
27 <h3 th:text="'Golongan Darah ' + ${penduduk.golongan_darah}">Penduduk Blood Type</h3>
28 <h3 th:text="'Agama ' + ${penduduk.agama}">Penduduk Religion</h3>
29 <h3 th:text="'Status Perkawinan ' + ${penduduk.status_perkawinan}">Penduduk Marriage Status</h3>
30 <h3 th:text="'Pekerjaan ' + ${penduduk.pekerjaan}">Penduduk Job</h3>
31 <h3 th:text="'Kewarganegaraan ' + ${penduduk.is_wni} == 1 ? 'WNI' : 'WNA'">Penduduk Kewarganegaraan</h3>
32 <h3 th:text="'Status Kematian ' + ${penduduk.is_wafat} == 0 ? 'Hidup' : 'Mati'">Penduduk Alive</h3>
33
34 <form action="/penduduk/mati" method="post" >
35 <input type="hidden" name="nik" th:value="${penduduk.nik}" />
36 <div>
37 <button type="submit">Ubah Status Kematian</button>
38 </div>
39 </form>
40
```

Index html

```

1 <!DOCTYPE HTML>
2 <html xmlns="http://www.w3.org/1999/xhtml"
3     xmlns:th="http://www.thymeleaf.org">
4 <head>
5
6 <title>Home
7 </title>
8 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
9 <link rel="stylesheet" href="/css/bootstrap.min.css" />
10 <link rel="stylesheet" href="https://cdn.datatables.net/v/dt-1.10.16/datatables.min.css" />
11 <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
12 <script src="https://cdn.datatables.net/1.10.7/js/jquery.dataTables.min.js"></script>
13 <script type="text/javascript" src="/js/bootstrap.min.js"></script>
14 <script type="text/javascript">
15     $(document).ready(function() {
16         $('#table').DataTable();
17     });
18 </script>
19 </head>
20
21 <body>
22 <div th:replace="fragments/fragment :: header"></div>
23 <h1 class="page-header">Home</h1>
24
25 <form action="/penduduk" method="get">
26 <div>
27     <label for="nik">Masukan NIK</label> <input type="text" name="nik" />
28 </div>
29 <div>
30     <button type="submit">Cari Penduduk</button>
31 </div>
32 </form>
33
34 <form action="/keluarga" method="get">
35 <div>
36     <label for="nik">Masukan NIK</label> <input type="text" name="nik" />
37 </div>
38 <div>

```

Fragments

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 </head>
5 <body>
6 <div th:fragment="header">
7 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
8 <a class="navbar-brand" href="/penduduk/cari">SiPenduduk</a>
9 <button class="navbar-toggler" type="button" data-toggle="collapse"
10 data-target="#navbarSupportedContent"
11 aria-controls="navbarSupportedContent" aria-expanded="false"
12 aria-label="Toggle navigation">
13 <span class="navbar-toggler-icon"></span>
14 </button>
15 <div class="collapse navbar-collapse" id="navbarSupportedContent">
16 <ul class="navbar-nav mr-auto">
17 <li class="nav-item">
18 <a class="nav-link" href="/">Home</a>
19 </li>
20 <li class="nav-item">
21 <a class="nav-link" href="/penduduk/cari">List Penduduk</a>
22 </li>
23 <li class="nav-item">
24 <a class="nav-link" href="/penduduk/tambah">Tambah Penduduk</a>
25 </li>
26 <li class="nav-item">
27 <a class="nav-link" href="/keluarga/tambah">Tambah Keluarga</a>
28 </li>
29 </ul>
30 <form class="form-inline my-2 my-lg-0">
31 <input class="form-control mr-sm-2"
32 type="text"
33 placeholder="Search"
34 aria-label="Search" />
35 <button class="btn btn-outline-success my-2 my-sm-0" type="submit">
36 Search
37 </button>
38 </form>

```

2. Tampilkan Daftar Penduduk Berdasarkan Nkk

Untuk Mengerjakan soal ini yang saya lakukan adalah membuat mapping sesuai dengan permintaan soal. Kemudian membuat mapper select untuk mengambil data penduduk melalui nkk. Dalam hal ini saya membutuhkan 2 mapper tambahan yang memiliki fungsi untuk mencari informasi id keluarga berdasarkan nkk. Dan mendapatkan list penduduk dengan id keluarga. Ini adalah mapper yang digunakan untuk mengerjakan soal no 2.

```
//Soal 2
@Select("select nik, nama, tempat_lahir, id_keluarga, tanggal_lahir, jenis_kelamin, golongan_darah, agama, is_wni, status_perkawinan,
"from penduduk where id_keluarga = #{id}")
@Results(value = { @Result(property = "nik", column = "nik"), @Result(property = "nama", column = "nama"),
    @Result(property = "tempat_lahir", column = "tempat_lahir"), @Result(property = "tanggal_lahir", column = "tanggal_lahir"),
    @Result(property = "jenis_kelamin", column = "jenis_kelamin"), @Result(property = "golongan_darah", column = "golongan_darah"),
    @Result(property = "agama", column = "agama"), @Result(property = "is_wni", column = "is_wni"),
    @Result(property = "status_perkawinan", column = "status_perkawinan"),
    @Result(property = "pekerjaan", column = "pekerjaan"), @Result(property = "is_wafat", column = "is_wafat"),
    @Result(property = "id_keluarga", column = "id_keluarga") })
List<PendudukModel> selectpenduduks(@Param("id") String id);

@Select("select id, nomor_kk, id_kelurahan, alamat, rt, rw " +
"from keluarga where nomor_kk = #{nkk}")
@Results(value = { @Result(property = "alamat", column = "alamat"),
    @Result(property = "id", column = "id"),
    @Result(property = "nomor_kk", column = "nomor_kk"),
    @Result(property = "rt", column = "rt"),
    @Result(property = "rw", column = "rw")})
KeluargaModel selectkeluarga2(@Param("nkk") String nkk);
```

Setelah itu saya menambahkan service untuk membuat *interface* yang akan dipakai pada soal 2. Kemudian baru saya membuat service database dimana berisi logic untuk menghubungkan tabel. Ini adalah fungsi dari servicedatabase yang digunakan untuk mengerjakan soal 2. Ada beberapa fungsi dari mapper yang sudah dibuat pada soal no 1 yang bisa digunakan. Berikut adalah *servicedatabase* yang dibuat

```
//soal 2
@Override
public KeluargaModel selectkeluarga (String nkk) {
    KeluargaModel keluarga = pendudukMapper.selectkeluarga2(nkk);
    List<PendudukModel> penduduk = pendudukMapper.selectpenduduks(keluarga.getId());
    KelurahanModel kelurahan = pendudukMapper.selectkelurahan(keluarga.getId_kelurahan());
    KecamatanModel kecamatan = pendudukMapper.selectkecamatan(kelurahan.getId_kecamatan());
    KotaModel kota = pendudukMapper.selectkota(kecamatan.getId_kota());
    keluarga.setPenduduks(penduduk);
    keluarga.setKelurahan(kelurahan);
    kelurahan.setKecamatan(kecamatan);
    kecamatan.setKota(kota);
    return keluarga;
}
```

Baru setelah itu membuat tampilan, tampilan yang digunakan hanya sekedar untuk mengetahui bahwa fungsi berjalan atau tidak.

SiPenduduk
[Home](#)
[List Penduduk](#)
[Tambah Penduduk](#)
[Tambah Keluarga](#)

Home

Masukan NIK

Masukan NKK

Gambar diatas adalah fitur setelah menerapkan bootstrap dengan cara fragmentasi. Setelah cari penduduk maka akan keluar tampilan sebagai berikut.

SiPenduduk

Home

List Penduduk

Tambah Penduduk

Tambah Keluarga

Search

Search

NKK = 3175060709120001

Alamat = Ki. Gegerkalong Hilir No. 98

RT/RW = 001/001

Kelurahan/Desa = KALI BARU

Kecamatan = CILINCING

Kota = KOTA JAKARTA UTARA

Anggota Keluarga

Show 10 entries

Search:

No	Nama Lengkap	NIK	Jenis Kelamin	Tempat Lahir	Tanggal Lahir	Agama	Pekerjaan	Status Perkawinan	Status Dalam Keluarga	Kewarganegaraan
1	Rachel Agustina	3175066506930003	Perempuan	Jakarta	1993-06-25	Islam	BURUH TANI/PERKEBUNAN	Belum Kawin		WNI
2	Hesti Hastuti	3175065604800002	Perempuan	Jakarta	1980-04-16	Islam	TUKANG LAS/PANDAI BESI	Kawin		WNI

Contoh potongan file html yang ditambahkan pada soal ini

```
</script>
</head>
<body>
<div th:replace="fragments/fragment :: header"></div>
<h3 th:text="'NKK = ' + ${keluarga.nomor_kk}">NKK KELUARGA</h3>
<h3 th:text="'Alamat = ' + ${keluarga.alamat}">ALAMAT KELUARGA</h3>
<h3 th:text="'RT/RW = ' + ${keluarga.rt} + '/' + ${keluarga.rw}">RT/RW</h3>
<h3 th:text="'Kelurahan/Desa = ' + ${keluarga.kelurahan.getNama_kelurahan()}">KELURAHAN</h3>
<h3 th:text="'Kecamatan = ' + ${keluarga.kelurahan.kecamatan.getNama_kecamatan()}">KECAMATAN</h3>
<h3 th:text="'Kota = ' + ${keluarga.kelurahan.kecamatan.kota.getNama_kota()}">KOTA</h3>
<hr/>
<br/>
<h3>Anggota Keluarga</h3>
<br/>
<table id = "table">
<thead>
<tr>
<th>No</th>
<th>Nama Lengkap</th>
<th>NIK</th>
<th>Jenis Kelamin</th>
<th>Tempat Lahir</th>
<th>Tanggal Lahir</th>
<th>Agama</th>
<th>Pekerjaan</th>
<th>Status Perkawinan</th>
<th>Status Dalam Keluarga</th>
<th>Kewarganegaraan</th>
</tr>
</thead>
<tbody>
<tr th:each="penduduk, iterationStatus: ${keluarga.penduduks}">
<td th:text="${iterationStatus.count}">No. 1</td>
<td th:text="${penduduk.nama}">Nama Penduduk</td>
```

3. Fitur Add Penduduk dan
4. Fitur Add Keluarga

Sama halnya dengan cara pengerjaan no 1 dan no 2, yang dilakukan adalah membuat menambahkan mapper sesuai dengan kebutuhan dalam hal ini saya menggunakan *insert*

```
// soal 3
@Select("Select max(id) from penduduk")
String getMaxId();

//soal 4
@Select("Select max(id) from keluarga")
String getMaxIdKeluarga();

//soal 3
@Select("Select nik from penduduk where nik between #{min} and #{max} and jenis_kelamin = #{jenis_kelamin} Order by nik desc limit 1")
String getNikPenduduk(@Param("min") String min, @Param("max") String max, @Param("jenis_kelamin") String jenis_kelamin);

@Insert("insert into penduduk(id,nik,nama,tempat_lahir,tanggal_lahir,jenis_kelamin, is_wni, id_keluarga,agama,pekerjaan,status_perkawi
#{agama}, #{pekerjaan}, #{status_perkawinan}, #{status_dalam_keluarga}, #{golongan_darah}, #{is_wafat})")
void addpenduduks(PendudukModel penduduk);

// soal 4
@Insert("insert into keluarga(id,nomor_kk,alamat,rt,rw,id_kelurahan,is_tidak_berlaku) values (#{id}, #{nomor_kk}, #{alamat}, #{rt}, #{
    "#{is_tidak_berlaku}} ")")
void addkeluargas(KeluargaModel keluarga);
```

Fungsi baru yang digunakan adalah get max id, untuk mendapatkan id paling tinggi, digunakan untuk generate id dan fungsi untuk mencari 12 string awal yang sama.

Kemudian menambahkan method pada *servicedatabase* disini juga terjadi perhitungan construct nik sesuai permintaan soal. Method lain yang ditambahkan adalah add penduduk pada service database yang digunakan untuk melakukan insert ke database. Pada soal 3 dan soal 4 ini beberapa fungsi di mapper yang telah dibuat sebelumnya juga bisa diterapkan.

```
//soal 3
@Override
public PendudukModel addpenduduk (String nama, String tempat, String tanggal, String is_wni, String pekerjaan, String golongan_darah,
    KeluargaModel keluarga = pendudukMapper.selectkeluarga(id_keluarga);
    KelurahanModel kelurahan = pendudukMapper.selectkelurahan(keluarga.getId_kelurahan());
    KecamatanModel kecamatan = pendudukMapper.selectkecamatan(kelurahan.getId_kecamatan());
    String nik1 = kecamatan.getKode_kecamatan();
    String[] tanggals = tanggal.split("-");
    String tanggalf = tanggals[2] + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
    System.out.println(tanggalf);
    String tanggalf = tanggals[2] + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
    nik1 = nik1.substring(0, nik1.length() - 1);
    if(jenis_kelamin.equals("1")){
        int nik2 = Integer.parseInt(tanggals[2].substring(0,1));
        nik2 += 4;
        System.out.println(nik2);
        String temp = nik2 + tanggals[2].substring(1,2);
        tanggalf = temp + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
        System.out.println(tanggalf);
    }
    String nik = nik1 + tanggalf;
    String min = nik + "0001";
    String max = nik + "0999";
    String nik4 = pendudukMapper.getNikPenduduk(min,max, jenis_kelamin);
    String nikf = "";
    if(nik4 != null){
        nik4 = nik4.substring(12, nik4.length());
        int yes = Integer.parseInt(nik4) + 1;
        if (yes < 10 ){
            nikf = nik + "000" + yes;
        }
        else if (yes < 100 ){
            nikf = nik + "00" + yes;
        }
        else if (yes < 1000 ){
            nikf = nik + "0" + yes;
        }
    }
}
```

```
//soal 4
@Override
public KeluargaModel addKeluarga(String alamat, String rt, String rw, String nama_kelurahan){
    String awal6 = pendudukMapper.selectkodekelurahan(nama_kelurahan);
    String id_kelurahan = pendudukMapper.selectidkelurahan(nama_kelurahan);
    System.out.println(id_kelurahan);
    awal6 = awal6.substring(0,6);
    System.out.println(awal6);
    String date = new SimpleDateFormat("yyyy-MM-dd").format(new Date());
    System.out.println(date);
    String[] tanggals = date.split("-");
    String tanggalf = tanggals[2] + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
    String nkk = awal6 + tanggalf;
    String min = nkk + "0001";
    System.out.println(nkk);
    String max = nkk + "0999";
    System.out.println(max);
    String nkkf = pendudukMapper.getNkkKeluarga(min,max);
    if(nkkf != null){
        System.out.println(nkkf);
        nkkf = nkkf.substring(12, nkkf.length());
        int yes = Integer.parseInt(nkkf) +1;
        if (yes < 10 ){
            nkkf =nkk + "000"+ yes;
        }
        else if (yes < 100 ){
            nkkf =nkk + "00"+ yes;
        }
        else if (yes < 1000 ){
            nkkf =nkk + "0"+ yes;
        }
    }
    else{
        nkkf = nkk + "0001";
    }
}
```

Dan tampilan

SiPenduduk
[Home](#)
[List Penduduk](#)
[Tambah Penduduk](#)
[Tambah Keluarga](#)

Tambah Penduduk

Nama
Tempat Lahir
Tanggal Lahir
Golongan Darah
Agama
Status Perkawinan
Jenis Kelamin
Pekerjaan
Kewarganegaraan
Status Kematian
Status dalam Keluarga
Id Keluarga

Tambah Keluarga

Alamat

RT

RW

Kelurahan/Desa

Kecamatan

Kota

Setelah berhasil add maka akan mucul tampilan

Data berhasil ditambahkan

Keluarga dengan Nkk 3101012210170002 berhasil ditambahkan

5. Fitur 5 dan 6 update

Pada mapper ditambahkan

```
// soal 5
@Update("update penduduk SET nik = #{nik}, nama = #{nama}, tempat_lahir = #{tempat_lahir},
agama = #{agama}, pekerjaan = #{pekerjaan}, status_perkawinan = #{status_perkawinan}")
void updatePenduduk(PendudukModel penduduk);

//soal 6
@Update("update keluarga SET nomor_kk = #{nomor_kk}, alamat = #{alamat}, RT = #{RT}")
void updateKeluarga(KeluargaModel keluarga);
```

Pada Service Database

```
//soal 5
@Override
public PendudukModel uppenduduk (String id, String nama, String tempat, String tanggal, String is_wni, String pekerjaan, String jenis_kelamin) {
    KeluargaModel keluarga = pendudukMapper.selectkeluarga(id_keluarga);
    KelurahanModel kelurahan = pendudukMapper.selectkelurahan(keluarga.getId_kelurahan());
    KecamatanModel kecamatan = pendudukMapper.selectkecamatan(kelurahan.getId_kecamatan());
    String nik1 = kecamatan.getKode_kecamatan();
    String[] tanggals = tanggal.split("-");
    String tanggalf = tanggals[2] + tanggals[1] + tanggals[0].substring(2, tanggals[0].length());
    System.out.println(tanggalf);
    nik1 = nik1.substring(0, nik1.length() - 1);
    if(jenis_kelamin.equals("1")){
        int nik2 = Integer.parseInt(tanggals[2].substring(0,1));
        nik2 += 4;
        System.out.println(nik2);
        String temp = nik2 + tanggals[2].substring(1,2);
        tanggalf = temp + tanggals[1] + tanggals[0].substring(2, tanggals[0].length());
        System.out.println(tanggalf);
    }
    String nik = nik1 + tanggalf;
    String min = nik + "0001";
    String max = nik + "0999";
    String nik4 = pendudukMapper.getNikPenduduk(min, max, jenis_kelamin);
    String nikf = "";
    if(nik4 != null){
        nik4 = nik4.substring(12, nik4.length());
        int yes = Integer.parseInt(nik4) + 1;
        if (yes < 10 ){
            nikf = nik + "000" + yes;
        }
        else if (yes < 100 ){
            nikf = nik + "00" + yes;
        }
        else if (yes < 1000 ){
            nikf = nik + "0" + yes;
        }
    }
}
```

Sama halnya saat membuat penduduk dan keluarga baru hal yang berbeda adalah tidak perlu membuat id baru.

Untuk menyelesaikan soal ini diperlukan 2 mapping mapping /penduduk/update dan penduduk/update/submit. Sama halnya dengan soal 6 memerlukan 2 mapping. Mapping penduduk/update pertama digunakan untuk menampilkan form yang sudah diisi sesuai dengan database berikut hasil tampilan dan code

Update Penduduk

Nama

Tempat Lahir

Tanggal Lahir

Golongan Darah

Agama

Status Perkawinan

Jenis Kelamin

Pekerjaan

Kewarganegaraan

Status Kematian

Status dalam Keluarga

Id Keluarga

Update Keluarga

Alamat

RT

RW

kelurahan

Source Code

```

<?xml:th="http://www.thymeleaf.org">
<head>

<title>Update Keluarga</title>
<link rel="stylesheet" href="/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://cdn.datatables.net/v/dt-1.10.16/datatables.min.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdn.datatables.net/1.10.7/js/jquery.dataTables.min.js"></script>
<script type="text/javascript" src="/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#table').DataTable();
    });
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>
    <div th:replace="fragments/fragment :: header"></div>
    <h1 class="page-header">Update Keluarga</h1>

    <form action="/keluarga/update/submit" method="post" >
        <input type="hidden" name="nkk" th:value="${keluarga.nomor_kk}" />
        <div>
            <label for="nama">Alamat</label> <input type="text" name="nama" th:value="${keluarga.
        </div>
        <div>
            <label for="rt">RT</label> <input type="text" name="rt" th:value="${keluarga.rt}" th:
        </div>
    </form>

```

```

1 <!DOCTYPE HTML>
2 <html xmlns="http://www.w3.org/1999/xhtml"
3     xmlns:th="http://www.thymeleaf.org">
4 <head>
5
6 <title>Update Penduduk</title>
7 <link rel="stylesheet" href="/css/bootstrap.min.css" />
8 <link rel="stylesheet" href="https://cdn.datatables.net/v/dt-1.10.16/datatables.min.css" />
9 <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
10 <script src="//cdn.datatables.net/1.10.7/js/jquery.dataTables.min.js"></script>
11 <script type="text/javascript" src="/js/bootstrap.min.js"></script>
12 <script type="text/javascript">
13     $(document).ready(function() {
14         $('#table').DataTable();
15     });
16 </script>
17 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
18 </head>
19
20 <body>
21 <div th:replace="fragments/fragment :: header"></div>
22 <h1 class="page-header">Update Penduduk</h1>
23
24 <form action="/penduduk/update/submit" method="post" >
25     <input type="hidden" name="nik" th:value="${penduduk.nik}" />
26     <div>
27         <label for="nama">Nama</label> <input type="text" name="nama" th:value="${penduduk.nama}" th:field="${penduduk.nama}" />
28     </div>
29     <div>
30         <label for="tempat_lahir">Tempat Lahir</label> <input type="text" name="tempat_lahir" th:value="${penduduk.tempat_lahir}" th:field="${penduduk.tempat_lahir}" />
31     </div>
32     <div>
33         <label for="tanggal_lahir">Tanggal Lahir</label> <input type="text" name="tanggal_lahir" th:value="${penduduk.tanggal_lahir}" th:field="${penduduk.tanggal_lahir}" />
34     </div>
35     <div>
36         <label for="golongan_darah">Golongan Darah</label>
37         <select name="golongan_darah" th:field="${penduduk.golongan_darah}">

```

Mapping /submit digunakan untuk fungsi update ke database dan menampilkan halaman sukses

6. Update status kematian

- Menambahkan tombol ubah status pada penduduk?nik=
- Menambahkan fungsi di mapper
- Menambahkan fungsi servicedatabase
- Menambahkan fungsi di service
- Membuat view html
- Menambahkan membuat mapping

7. Mencari penduduk

- Membuat halaman html(4 halaman)
- Membuat mapping
- Menambahkan fungsi di mapper
- Menambahkan fungsi servicedatabase
- Menambahkan fungsi di service
- Implementasi datatable

Tampilan cari penduduk

SiPenduduk

HomeList PendudukTambah PendudukTambah Keluarga

Search

Tampilkan List Keluarga

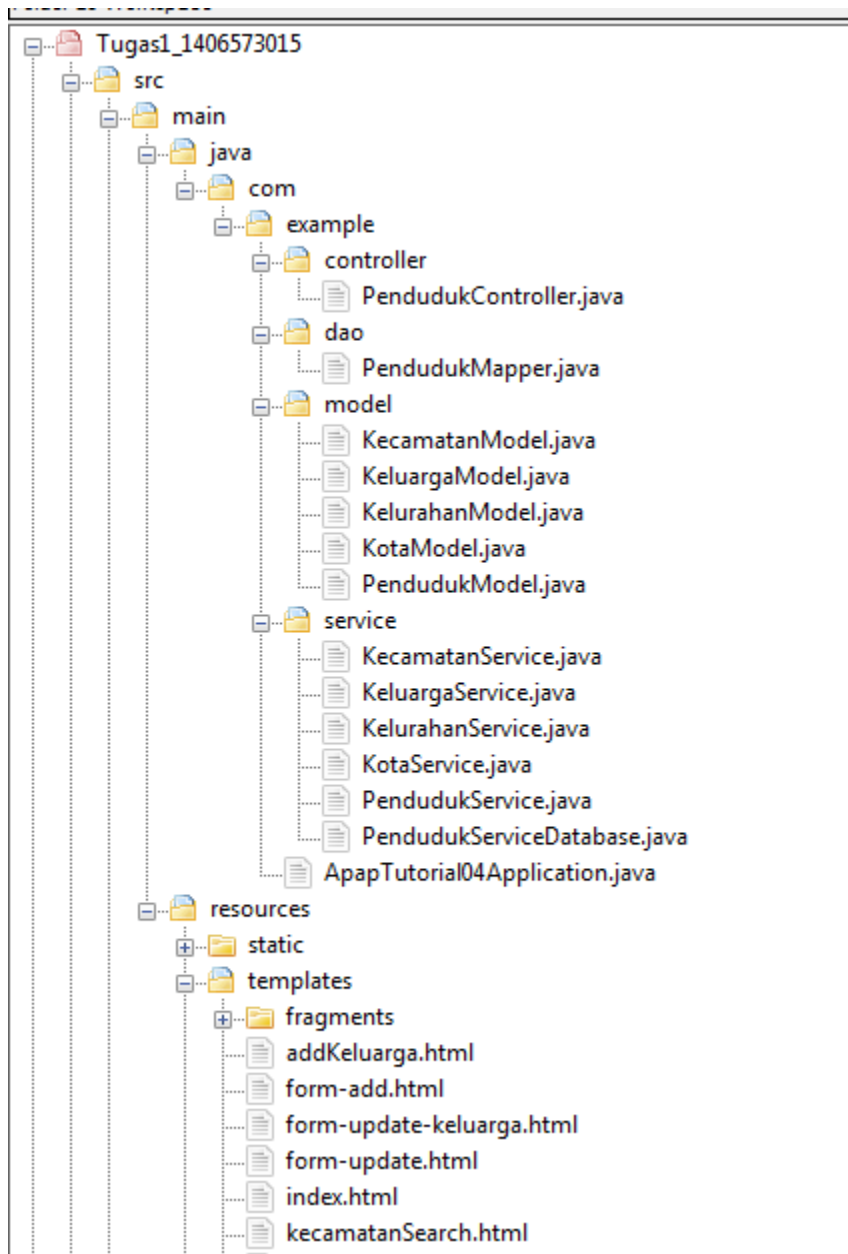
KotaKABUPATEN KEPULAUAN SERIBU KecamatanKEPULAUAN SERIBU SE KelurahanPULAU TIDUNG

Show10 entries

Search:

Nik	Nama	Jenis Kelamin
3101010101170001	Caraka Marbun	0
3101010102640001	Aris Edward Gunawan S.Sos	0
3101010103810001	Karma Samosir	0
3101010103920001	Wakiman Sitorus	0
3101010103940001	Janwi Sihombing S.EI	0
3101010104890001	Taswir Hutasoit	0
3101010104910001	Mahfud Setiawan M.TI.	0
3101010105890001	Radika Dabukke	0
3101010108140001	Dalimin Ihsan Manullang S.Kom	0
3101010109130001	Emong Pranowo M.M.	0

Struktur File



Dalam folder terdapat model, model merupakan sebuah objek yang merepresentasikan dan menyimpan informasi tertentu. Pada proyek ini ada 5 model yang dibuat sesuai dengan jumlah database dan setiap model mempunyai informasi yang berbeda. Kemudian terdapat service, service digunakan sebagai penghubung antara controller dan database. Didalam service terdapat fungsi logic untuk memanipulasi suatu informasi tertentu dari view, sehingga informasi tersebut bisa diolah oleh database, pada tugas yang dikerjakan hampir semua fungsi service-service yang penting ada dalam penduduk Servicedatabase. Sedangkan semua http request diolah pada penduduk controller. Controller berfungsi sebagai penghubung antara layer view dan service.

Kemudian terdapat mapper, mapper yang digunakan hanya ada 1 yaitu penduduk mapper. Mapper berfungsi untuk Create, Insert, Delete, Update. Kemudian ada view, semua view diletakan pada folder template. View di gunakan sebagai tampilan yang berfungsi sebagai media interaksi antara user dan sistem.

Stress Testing

Stress Testing dilakukan pada halaman <http://localhost:8080/penduduk?nik=3101011112160001> menggunakan Jmeter. Berikut adalah hasil yang didapatkan

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: Browse...

Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time	Status	Bytes	Sent Bytes	Latency	Connect Time
235	15:16:36.711	Thread Group ...	HTTP Request	3972	✓	2673	147	3972	1
236	15:16:37.409	Thread Group ...	HTTP Request	3283	✓	2673	147	3283	2
237	15:16:36.599	Thread Group ...	HTTP Request	4093	✓	2673	147	4093	3
238	15:16:38.799	Thread Group ...	HTTP Request	1908	✓	2673	147	1907	2
239	15:16:38.764	Thread Group ...	HTTP Request	1953	✓	2673	147	1951	3
240	15:16:38.731	Thread Group ...	HTTP Request	1992	✓	2673	147	1992	1
241	15:16:38.770	Thread Group ...	HTTP Request	1959	✓	2673	147	1959	1
242	15:16:35.945	Thread Group ...	HTTP Request	4796	✓	2673	147	4774	2
243	15:16:38.970	Thread Group ...	HTTP Request	1774	✓	2673	147	1774	2
244	15:16:38.962	Thread Group ...	HTTP Request	1787	✓	2673	147	1781	1
245	15:16:38.937	Thread Group ...	HTTP Request	1813	✓	2673	147	1813	2
246	15:16:38.199	Thread Group ...	HTTP Request	4565	✓	2673	147	4565	1
247	15:16:38.979	Thread Group ...	HTTP Request	1791	✓	2673	147	1790	1
248	15:16:38.966	Thread Group ...	HTTP Request	1807	✓	2673	147	1806	2
249	15:16:38.810	Thread Group ...	HTTP Request	1986	✓	2673	147	1985	2
250	15:16:39.055	Thread Group ...	HTTP Request	1749	✓	2673	147	1749	1
251	15:16:39.004	Thread Group ...	HTTP Request	1806	✓	2673	147	1806	1
252	15:16:39.052	Thread Group ...	HTTP Request	1764	✓	2673	147	1764	2
253	15:16:39.086	Thread Group ...	HTTP Request	1739	✓	2673	147	1739	2
254	15:16:39.005	Thread Group ...	HTTP Request	1828	✓	2673	147	1828	2
255	15:16:35.829	Thread Group ...	HTTP Request	5037	✓	2673	147	5036	1
256	15:16:36.299	Thread Group ...	HTTP Request	4568	✓	2673	147	4568	1
257	15:16:39.093	Thread Group ...	HTTP Request	1779	✓	2673	147	1777	2
258	15:16:36.159	Thread Group ...	HTTP Request	4730	✓	2673	147	4730	1
259	15:16:36.149	Thread Group ...	HTTP Request	4756	✓	2673	147	4756	2
260	15:16:36.169	Thread Group ...	HTTP Request	4739	✓	2673	147	4738	9
261	15:16:38.648	Thread Group ...	HTTP Request	2264	✓	2673	147	2264	2

☐ Scroll automatically? ☐ Child samples? No of Samples 1000 Latest Sample 2550 Average 3256 Deviation 2245

Dari hasil tersebut dapat dikatakan bahwa rata2 waktu yang diperlukan oleh user ketika mengakses halaman <http://localhost:8080/penduduk?nik=3101011112160001> rata-rata 3256ms atau sekitar 3 detik ketika ada 1000 user yang mengakses bersamaan. Dari hasil ini dapat disimpulkan bahwa akses yang dilakukan tergolong cepat, karena hanya membutuhkan 3 detik ketika terdapat 1000 user yang mengakses secara bersamaan. Untuk melakukan optimasi yang dilakukan adalah membuat index pada tabel penduduk kolom id dan nik serta tabel keluarga kolom id.