

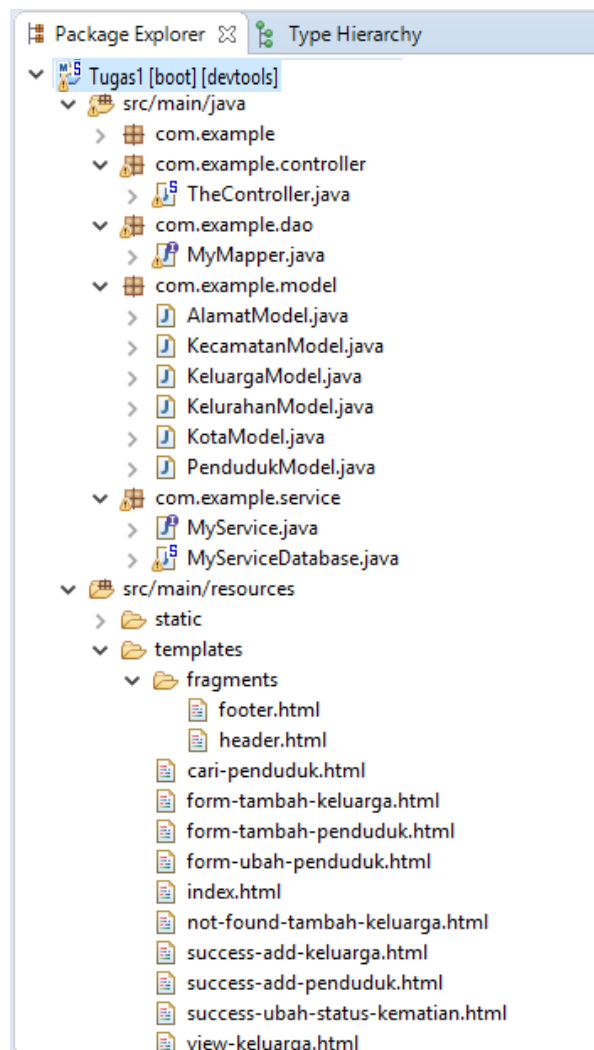
Write-up Tugas 1

Dayana Sheilla Agathy

- **Proses pengembangan Tugas 1**

Dalam mengembangkan Sistem Informasi Kependudukan ini, saya menerapkan konsep MVC atau Model-View-Controller. Hal pertama yang saya lakukan adalah dengan membuat package Controller, Model, Service, dan DAO. Kemudian, saya membuat class-class Model yang merupakan entitas dari setiap table yang terdapat di database. Class Model berisi atribut-atribut yang nantinya akan digunakan untuk mengembangkan tugas ini. Setelah membuat Model, saya membuat dua buah file yang berisi Interface Service dan Class Service pada package Service yang berisi logic code yang dibutuhkan. Setelah itu, saya membuat Mapper di dalam package DAO yang berisi query-query ke dalam database untuk mengambil data yang dibutuhkan untuk method yang digunakan pada Service. Kemudian terakhir saya membuat class Controller dalam package Controller. Di class Controller ini, berisi method yang akan menghubungkan method logic yang terdapat pada Service untuk dipetakan ke dalam url yang akan ditampilkan pada View. Pada tugas ini saya hanya membuat satu class Controller, Mapper, Service, dan class Model sesuai dengan entitas yang dibutuhkan. Setelah itu, ketika saya membuat pengempangan terhadap tugas ini saya melakukan tahapan yang telah saya jelaskan dengan berulang kali.

Berikut merupakan struktur file yang saya buat:



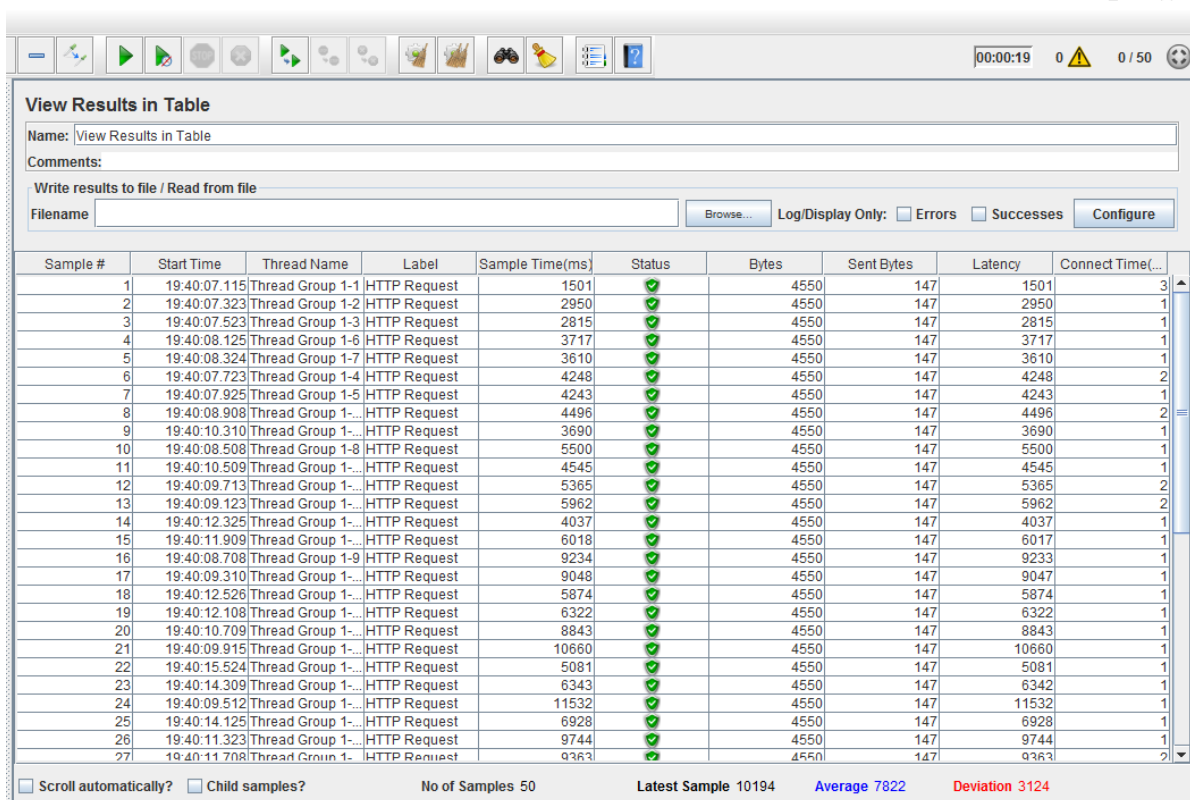
Selain itu, View terdapat pada folder resources. Di dalam folder tersebut terdapat dua folder yaitu folder static dan templates. Dimana folder static tempat saya menyimpan file-file css dan javascript dari framework Bootstrap. Sedangkan folder templates tempat saya menyimpan file html, selain itu terdapat folder fragment yang akan menyederhanakan pembuatan file html.

- **Optimasi database**

Pada database saya menambahkan primary key dan juga menghubungkan atribut yang berelasi dengan membuat foreign key pada atribut tersebut. Selain itu, saya melakukan beberapa perubahan yaitu dengan membuat id dari table Penduduk dan Keluarga menjadi autoincrement

- **Stress testing dengan Jmeter**

Untuk stress testing saya hanya menggunakan sampel dari salah satu fitur saja. Saya akan mencoba melakukannya dengan menggunakan fitur untuk mencari penduduk berdasarkan kota, kecamatan, dan kelurahan, yaitu <http://localhost:8080/penduduk?nik=3101011108170002>. Hal tersebut, dikarenakan pencarian untuk data tersebut cukup berat dikarenakan data penduduk yang cukup banyak. Saya mencoba melakukan stress testing dengan menggunakan 50 thread dan periode ramp-up selama 10 detik. Berikut hasil yang didapat sebelum saya melakukan optimasi.



View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: Browse...

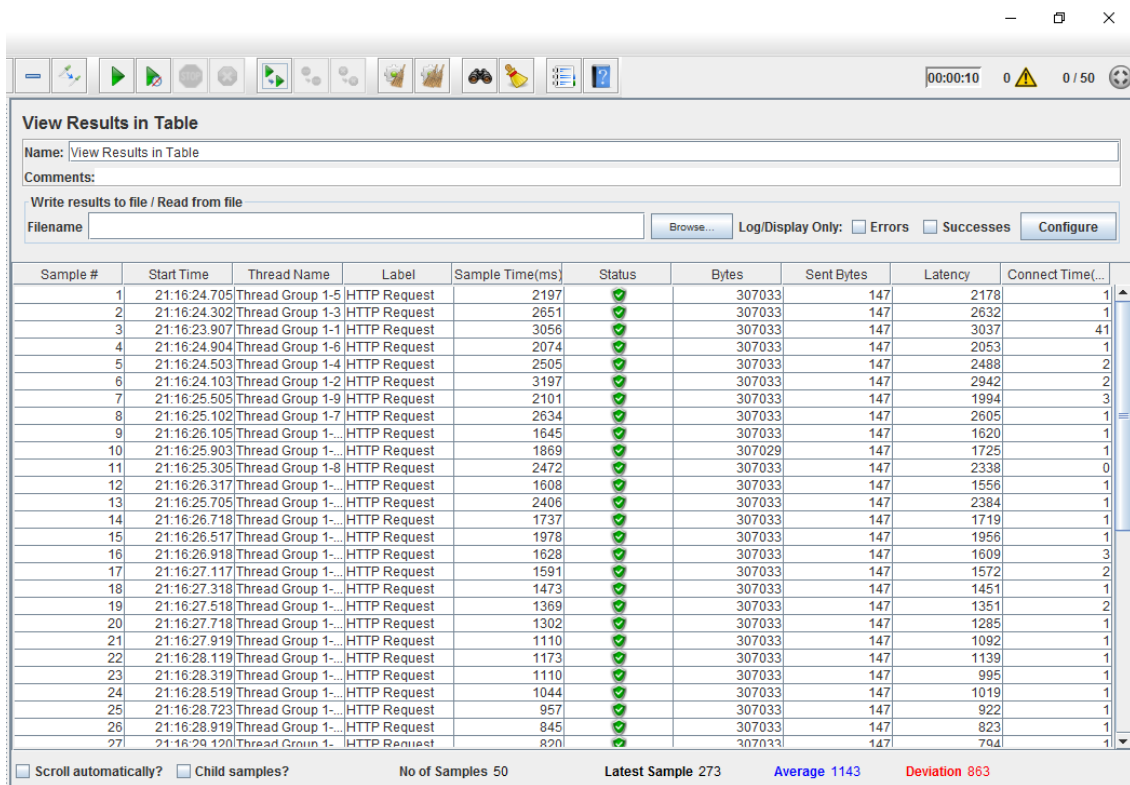
Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	19:40:07.115	Thread Group 1-1	HTTP Request	1501	✓	4550	147	1501	3
2	19:40:07.323	Thread Group 1-2	HTTP Request	2950	✓	4550	147	2950	1
3	19:40:07.523	Thread Group 1-3	HTTP Request	2815	✓	4550	147	2815	1
4	19:40:08.125	Thread Group 1-6	HTTP Request	3717	✓	4550	147	3717	1
5	19:40:08.324	Thread Group 1-7	HTTP Request	3610	✓	4550	147	3610	1
6	19:40:07.723	Thread Group 1-4	HTTP Request	4248	✓	4550	147	4248	2
7	19:40:07.925	Thread Group 1-5	HTTP Request	4243	✓	4550	147	4243	1
8	19:40:08.908	Thread Group 1-...	HTTP Request	4496	✓	4550	147	4496	2
9	19:40:10.310	Thread Group 1-...	HTTP Request	3690	✓	4550	147	3690	1
10	19:40:08.508	Thread Group 1-8	HTTP Request	5500	✓	4550	147	5500	1
11	19:40:10.509	Thread Group 1-...	HTTP Request	4545	✓	4550	147	4545	1
12	19:40:09.713	Thread Group 1-...	HTTP Request	5365	✓	4550	147	5365	2
13	19:40:09.123	Thread Group 1-...	HTTP Request	5962	✓	4550	147	5962	2
14	19:40:12.325	Thread Group 1-...	HTTP Request	4037	✓	4550	147	4037	1
15	19:40:11.909	Thread Group 1-...	HTTP Request	6018	✓	4550	147	6017	1
16	19:40:08.708	Thread Group 1-9	HTTP Request	9234	✓	4550	147	9233	1
17	19:40:09.310	Thread Group 1-...	HTTP Request	9048	✓	4550	147	9047	1
18	19:40:12.526	Thread Group 1-...	HTTP Request	5874	✓	4550	147	5874	1
19	19:40:12.108	Thread Group 1-...	HTTP Request	6322	✓	4550	147	6322	1
20	19:40:10.709	Thread Group 1-...	HTTP Request	8843	✓	4550	147	8843	1
21	19:40:09.915	Thread Group 1-...	HTTP Request	10660	✓	4550	147	10660	1
22	19:40:15.524	Thread Group 1-...	HTTP Request	5081	✓	4550	147	5081	1
23	19:40:14.309	Thread Group 1-...	HTTP Request	6343	✓	4550	147	6342	1
24	19:40:09.512	Thread Group 1-...	HTTP Request	11532	✓	4550	147	11532	1
25	19:40:14.125	Thread Group 1-...	HTTP Request	6928	✓	4550	147	6928	1
26	19:40:11.323	Thread Group 1-...	HTTP Request	9744	✓	4550	147	9744	1
27	19:40:11.708	Thread Group 1-...	HTTP Request	9363	✓	4550	147	9363	2

☐ Scroll automatically? ☐ Child samples? No of Samples 50 Latest Sample 10194 Average 7822 Deviation 3124

Dari hasil tersebut bisa dilihat rata-rata waktu untuk mengakses fitur ini adalah 7822 ms atau 7,8 detik dengan deviasi 3124 ms atau 3,1 detik, hal tersebut menunjukkan respon yang cukup lama. Kemudian saya melakukan optimasi yang seperti sudah saya jelaskan sebelumnya, bahwa saya membuat beberapa perubahan di struktur database di phpmyadmin. Setelah itu, dengan jumlah threads masih sama yaitu 50 didapatkan hasil yang cukup memuaskan. Response time menjadi turun drastis dibanding dengan response time sebelum dioptimasi dengan rata-rata waktu untuk mendapat respons 1143 ms atau sekitar 1 detik.

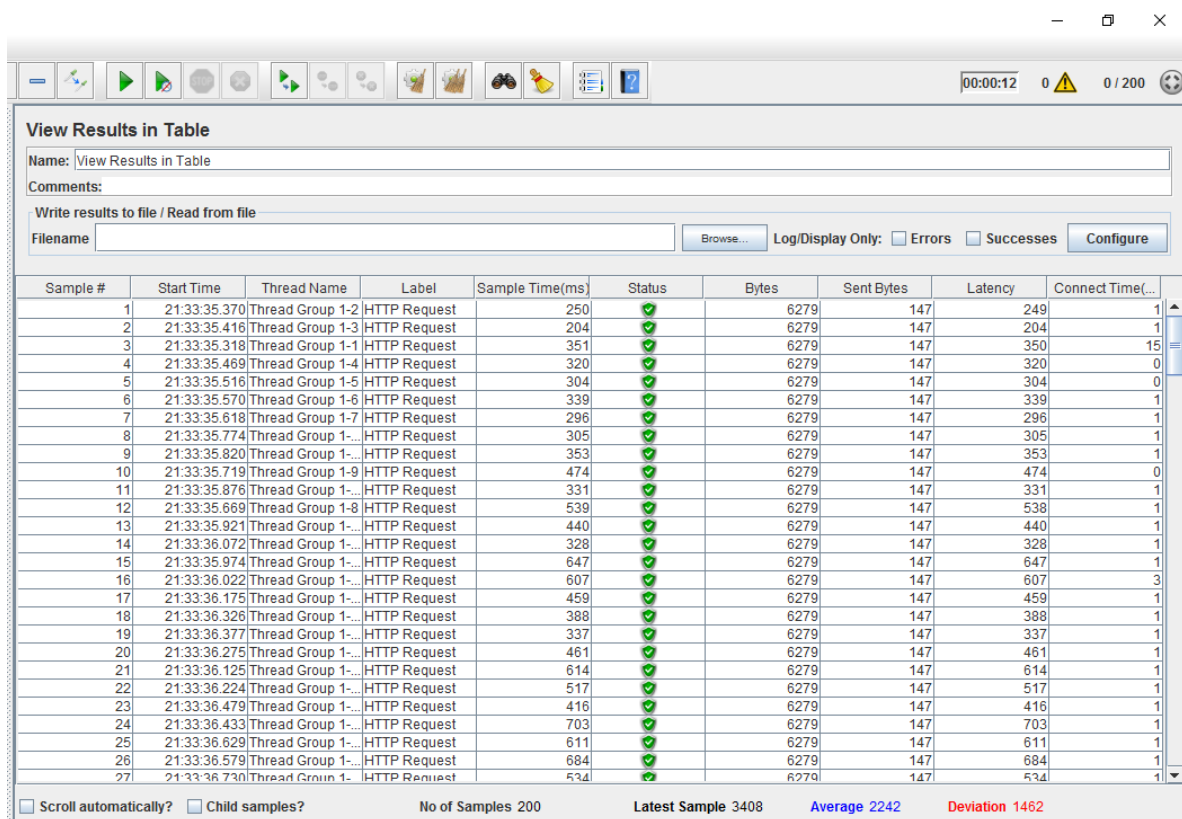
Dayana Sheilla Agathy
14065726330
Tugas 1 APAP



The screenshot shows the JMeter 'View Results in Table' window. The table displays 50 samples of HTTP requests. The status for all samples is 'Success' (green checkmark). The average latency is 1143 ms and the deviation is 863 ms.

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time...
1	21:16:24.705	Thread Group 1-5	HTTP Request	2197	Success	307033	147	2178	1
2	21:16:24.302	Thread Group 1-3	HTTP Request	2651	Success	307033	147	2632	1
3	21:16:23.907	Thread Group 1-1	HTTP Request	3056	Success	307033	147	3037	41
4	21:16:24.904	Thread Group 1-6	HTTP Request	2074	Success	307033	147	2053	1
5	21:16:24.503	Thread Group 1-4	HTTP Request	2505	Success	307033	147	2488	2
6	21:16:24.103	Thread Group 1-2	HTTP Request	3197	Success	307033	147	2942	2
7	21:16:25.505	Thread Group 1-9	HTTP Request	2101	Success	307033	147	1994	3
8	21:16:25.102	Thread Group 1-7	HTTP Request	2634	Success	307033	147	2605	1
9	21:16:26.105	Thread Group 1-...	HTTP Request	1645	Success	307033	147	1620	1
10	21:16:25.903	Thread Group 1-...	HTTP Request	1869	Success	307029	147	1725	1
11	21:16:25.305	Thread Group 1-8	HTTP Request	2472	Success	307033	147	2338	0
12	21:16:26.317	Thread Group 1-...	HTTP Request	1608	Success	307033	147	1556	1
13	21:16:25.705	Thread Group 1-...	HTTP Request	2406	Success	307033	147	2384	1
14	21:16:26.718	Thread Group 1-...	HTTP Request	1737	Success	307033	147	1719	1
15	21:16:26.517	Thread Group 1-...	HTTP Request	1978	Success	307033	147	1956	1
16	21:16:26.918	Thread Group 1-...	HTTP Request	1628	Success	307033	147	1609	3
17	21:16:27.117	Thread Group 1-...	HTTP Request	1591	Success	307033	147	1572	2
18	21:16:27.318	Thread Group 1-...	HTTP Request	1473	Success	307033	147	1451	1
19	21:16:27.518	Thread Group 1-...	HTTP Request	1369	Success	307033	147	1351	2
20	21:16:27.718	Thread Group 1-...	HTTP Request	1302	Success	307033	147	1285	1
21	21:16:27.919	Thread Group 1-...	HTTP Request	1110	Success	307033	147	1092	1
22	21:16:28.119	Thread Group 1-...	HTTP Request	1173	Success	307033	147	1139	1
23	21:16:28.319	Thread Group 1-...	HTTP Request	1110	Success	307033	147	995	1
24	21:16:28.519	Thread Group 1-...	HTTP Request	1044	Success	307033	147	1019	1
25	21:16:28.723	Thread Group 1-...	HTTP Request	957	Success	307033	147	922	1
26	21:16:28.919	Thread Group 1-...	HTTP Request	845	Success	307033	147	823	1
27	21:16:29.120	Thread Group 1-...	HTTP Request	820	Success	307033	147	794	1

Kemudian saya mencoba menaikkan jumlah thread menjadi 200.

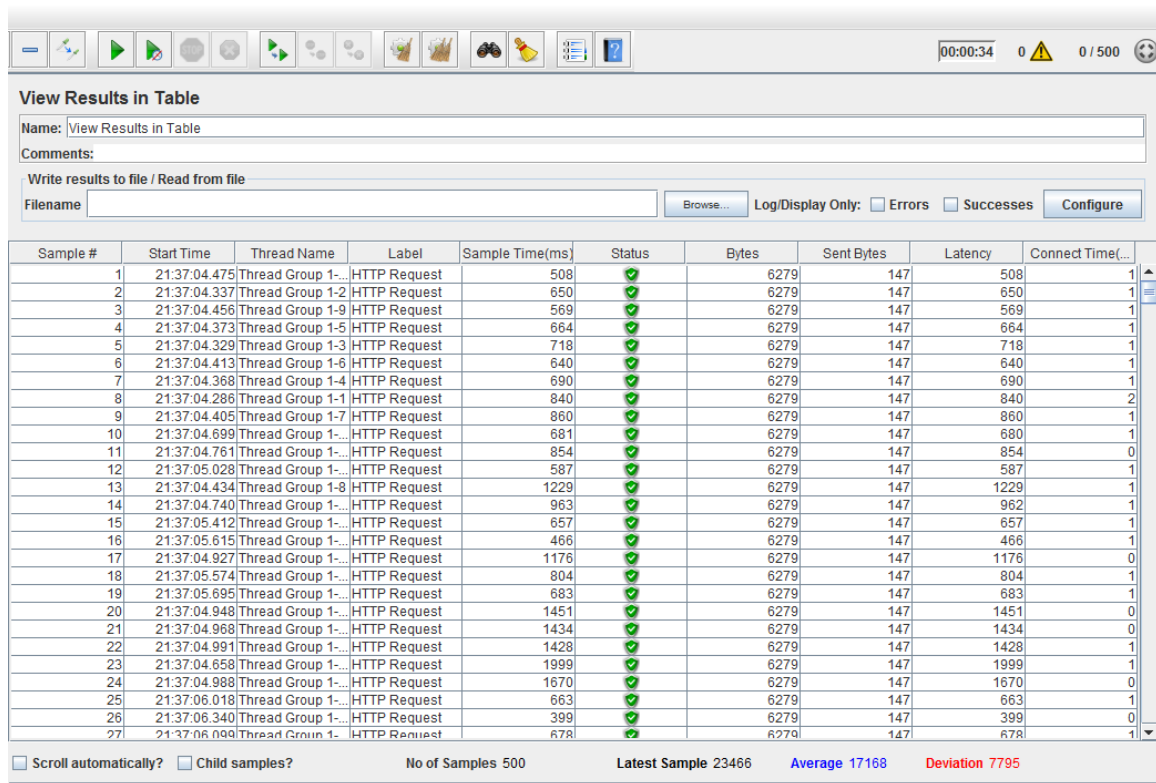


The screenshot shows the JMeter 'View Results in Table' window after increasing the number of threads to 200. The table displays 200 samples of HTTP requests. The status for all samples is 'Success' (green checkmark). The average latency is 2242 ms and the deviation is 1462 ms.

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time...
1	21:33:35.370	Thread Group 1-2	HTTP Request	250	Success	6279	147	249	1
2	21:33:35.416	Thread Group 1-3	HTTP Request	204	Success	6279	147	204	1
3	21:33:35.318	Thread Group 1-1	HTTP Request	351	Success	6279	147	350	15
4	21:33:35.469	Thread Group 1-4	HTTP Request	320	Success	6279	147	320	0
5	21:33:35.516	Thread Group 1-5	HTTP Request	304	Success	6279	147	304	0
6	21:33:35.570	Thread Group 1-6	HTTP Request	339	Success	6279	147	339	1
7	21:33:35.618	Thread Group 1-7	HTTP Request	296	Success	6279	147	296	1
8	21:33:35.774	Thread Group 1-...	HTTP Request	305	Success	6279	147	305	1
9	21:33:35.820	Thread Group 1-...	HTTP Request	353	Success	6279	147	353	1
10	21:33:35.719	Thread Group 1-9	HTTP Request	474	Success	6279	147	474	0
11	21:33:35.876	Thread Group 1-...	HTTP Request	331	Success	6279	147	331	1
12	21:33:35.669	Thread Group 1-8	HTTP Request	539	Success	6279	147	538	1
13	21:33:35.921	Thread Group 1-...	HTTP Request	440	Success	6279	147	440	1
14	21:33:36.072	Thread Group 1-...	HTTP Request	328	Success	6279	147	328	1
15	21:33:35.974	Thread Group 1-...	HTTP Request	647	Success	6279	147	647	1
16	21:33:36.022	Thread Group 1-...	HTTP Request	607	Success	6279	147	607	3
17	21:33:36.175	Thread Group 1-...	HTTP Request	459	Success	6279	147	459	1
18	21:33:36.326	Thread Group 1-...	HTTP Request	388	Success	6279	147	388	1
19	21:33:36.377	Thread Group 1-...	HTTP Request	337	Success	6279	147	337	1
20	21:33:36.275	Thread Group 1-...	HTTP Request	461	Success	6279	147	461	1
21	21:33:36.125	Thread Group 1-...	HTTP Request	614	Success	6279	147	614	1
22	21:33:36.224	Thread Group 1-...	HTTP Request	517	Success	6279	147	517	1
23	21:33:36.479	Thread Group 1-...	HTTP Request	416	Success	6279	147	416	1
24	21:33:36.433	Thread Group 1-...	HTTP Request	703	Success	6279	147	703	1
25	21:33:36.629	Thread Group 1-...	HTTP Request	611	Success	6279	147	611	1
26	21:33:36.579	Thread Group 1-...	HTTP Request	684	Success	6279	147	684	1
27	21:33:36.730	Thread Group 1-...	HTTP Request	534	Success	6279	147	534	1

Dayana Sheilla Agathy
14065726330
Tugas 1 APAP

Hasilnyapun masih cepat dengan rata-rata untuk mendapat respons 2242 ms atau sekitar 2 detik. Kemudian saya mencoba untuk menaikkan kembali jumlah thread menjadi 500.



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	21:37:04.475	Thread Group 1-...	HTTP Request	508	✓	6279	147	508	1
2	21:37:04.337	Thread Group 1-2	HTTP Request	650	✓	6279	147	650	1
3	21:37:04.456	Thread Group 1-9	HTTP Request	569	✓	6279	147	569	1
4	21:37:04.373	Thread Group 1-5	HTTP Request	664	✓	6279	147	664	1
5	21:37:04.329	Thread Group 1-3	HTTP Request	718	✓	6279	147	718	1
6	21:37:04.413	Thread Group 1-6	HTTP Request	640	✓	6279	147	640	1
7	21:37:04.368	Thread Group 1-4	HTTP Request	690	✓	6279	147	690	1
8	21:37:04.286	Thread Group 1-1	HTTP Request	840	✓	6279	147	840	2
9	21:37:04.405	Thread Group 1-7	HTTP Request	860	✓	6279	147	860	1
10	21:37:04.699	Thread Group 1-...	HTTP Request	681	✓	6279	147	680	1
11	21:37:04.761	Thread Group 1-...	HTTP Request	854	✓	6279	147	854	0
12	21:37:05.028	Thread Group 1-...	HTTP Request	587	✓	6279	147	587	1
13	21:37:04.434	Thread Group 1-8	HTTP Request	1229	✓	6279	147	1229	1
14	21:37:04.740	Thread Group 1-...	HTTP Request	963	✓	6279	147	962	1
15	21:37:05.412	Thread Group 1-...	HTTP Request	657	✓	6279	147	657	1
16	21:37:05.615	Thread Group 1-...	HTTP Request	466	✓	6279	147	466	1
17	21:37:04.927	Thread Group 1-...	HTTP Request	1176	✓	6279	147	1176	0
18	21:37:05.574	Thread Group 1-...	HTTP Request	804	✓	6279	147	804	1
19	21:37:05.695	Thread Group 1-...	HTTP Request	683	✓	6279	147	683	1
20	21:37:04.948	Thread Group 1-...	HTTP Request	1451	✓	6279	147	1451	0
21	21:37:04.968	Thread Group 1-...	HTTP Request	1434	✓	6279	147	1434	0
22	21:37:04.991	Thread Group 1-...	HTTP Request	1428	✓	6279	147	1428	1
23	21:37:04.658	Thread Group 1-...	HTTP Request	1999	✓	6279	147	1999	1
24	21:37:04.988	Thread Group 1-...	HTTP Request	1670	✓	6279	147	1670	0
25	21:37:06.018	Thread Group 1-...	HTTP Request	663	✓	6279	147	663	1
26	21:37:06.340	Thread Group 1-...	HTTP Request	399	✓	6279	147	399	0
27	21:37:06.099	Thread Group 1-...	HTTP Request	678	✓	6279	147	678	1

Scroll automatically? Child samples? No of Samples 500 Latest Sample 23466 Average 17168 Deviation 7795

Semakin threads meningkat maka respons time juga semakin lama. Pada jumlah threads sampai dengan 500 ini diperlukan waktu selama 17168 ms atau sekitar 17 detik untuk mendapat respons. Rata-rata tersebut masih cukup cepat karena mengingat jumlah threads yang cukup banyak.

- Fitur Tambahan

Saya mengerjakan salah satu fitur bonus yang ditawarkan, yaitu untuk **menambahkan fitur penduduk termuda dan penduduk tertua.**

SI Kependudukan	Home	Tambah Data ▾	Ubah Data ▾	Cari Data
-----------------	------	---------------	-------------	-----------

Lihat data penduduk di KOTA JAKARTA SELATAN Kecamatan PASAR MINGGU Kelurahan RAGUNAN

Penduduk Termuda		Penduduk Tertua	
NIK	3171022009170002	NIK	3171026310520001
Nama	Hairyanto Cecep Narpati	Nama	Almira Hassanah S.E.
Tanggal Lahir	2017-09-20	Tanggal Lahir	1952-10-23

Pada Mapper saya membuat query yang memanggil nik, nama dan tanggal lahir dari semua penduduk yang berada di kelurahan yang telah dipilih. Kemudian, untuk mendapatkan usia penduduk yang temuda saya melakukan "order by" tanggal lahir tersebut secara **descending** sehingga penduduk akan terurut dari yang mempunyai tanggal lahir yang paling besar hingga terkecil. Setelah itu saya melakukan limit 1 yang berarti hanya satu data saja yang akan saya ambil dan otomatis data paling atas yang akan diambil.

Sebaliknya, yang saya lakukan untuk mendapatkan usia penduduk paling tua adalah dengan melakukan "order by" tanggal lahir secara **ascending**, sehingga data penduduk akan terurut dari tanggal lahir yang paling kecil. Kemudian menggunakan limit 1 sehingga data yang diambil hanya satu dengan data yang berada yang paling atas. Berikut query yang saya lakukan.

```
@Select("select nik, nama, tanggal_lahir from penduduk JOIN "
+ "(select id from keluarga where id_kelurahan = #{id_kelurahan}) AS keluarga "
+ "ON keluarga.id = penduduk.id_keluarga "
+ "ORDER BY tanggal_lahir DESC "
+ "LIMIT 1")
PendudukModel getPendudukTermudaSekelurahan(int id_kelurahan);

@Select("select nik, nama, tanggal_lahir from penduduk JOIN "
+ "(select id from keluarga where id_kelurahan = #{id_kelurahan}) AS keluarga "
+ "ON keluarga.id = penduduk.id_keluarga "
+ "ORDER BY tanggal_lahir ASC "
+ "LIMIT 1")
PendudukModel getPendudukTertuaSekelurahan(int id_kelurahan);
```

Setelah itu, saya mengimplementasikan method dari Mapper tersebut ke dalam class Service, seperti gambar di bawah.

```
@Override
public PendudukModel getPendudukTermudaSekelurahan(int id_kelurahan) {
    return myMapper.getPendudukTermudaSekelurahan(id_kelurahan);
}

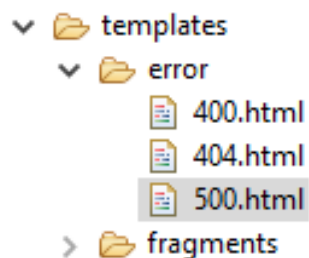
@Override
public PendudukModel getPendudukTertuaSekelurahan(int id_kelurahan) {
    return myMapper.getPendudukTertuaSekelurahan(id_kelurahan);
}
```

Kemudian, method tersebut saya implementasikan di dalam class Controller pada method cariPenduduk, sehingga dapat ditampilkan pada View. Berikut method CariPenduduk pada class Controller

Dayana Sheilla Agathy
14065726330
Tugas 1 APAP

```
@RequestMapping(value = "/penduduk/cari")
public String cariPenduduk (Model model,
    @RequestParam(value = "kt", required = false, defaultValue = "0") int id_kota,
    @RequestParam(value = "kc", required = false, defaultValue = "0") int id_kecamatan,
    @RequestParam(value = "kl", required = false, defaultValue = "0") int id_kelurahan){
    List<KotaModel> kota_list = myService.selectKotaList();
    model.addAttribute("kota_list", kota_list);
    System.out.println(kota_list);
    if (id_kelurahan != 0) {
        model.addAttribute("nama_kota", myService.selectKotaByID(id_kota).getNama_kota());
        model.addAttribute("id_kota", id_kota);
        model.addAttribute("nama_kecamatan", myService.selectKecamatanByID(id_kecamatan).getNama_kecamatan());
        model.addAttribute("id_kecamatan", id_kecamatan);
        model.addAttribute("nama_kelurahan", myService.selectKelurahanByID(id_kelurahan).getNama_kelurahan());
        model.addAttribute("id_kelurahan", id_kelurahan);
        model.addAttribute("view", "view");
        model.addAttribute("penduduk_list", myService.selectPendudukByIdKelurahan(id_kelurahan));
        model.addAttribute("penduduk_termuda", myService.getPendudukTermudaSekelurahan(id_kelurahan));
        model.addAttribute("penduduk_tertua", myService.getPendudukTertuaSekelurahan(id_kelurahan));
    } else if (id_kecamatan != 0) {
        model.addAttribute("nama_kota", myService.selectKotaByID(id_kota).getNama_kota());
        model.addAttribute("id_kota", id_kota);
        model.addAttribute("nama_kecamatan", myService.selectKecamatanByID(id_kecamatan).getNama_kecamatan());
        model.addAttribute("id_kecamatan", id_kecamatan);
        model.addAttribute("kelurahan_list", myService.selectKelurahanList(id_kecamatan));
    } else if (id_kota != 0) {
        model.addAttribute("nama_kota", myService.selectKotaByID(id_kota).getNama_kota());
        model.addAttribute("id_kota", id_kota);
        model.addAttribute("kecamatan_list", myService.selectKecamatanList(id_kota));
    }
    return "cari-penduduk";
}
```

Selain itu saya juga menambahkan **error page**.



Selain fitur bonus diatas, pada tugas kali ini saya hanya mengerjakan 6 fitur utama yaitu,

- **Fitur 1** Tampilan Data Penduduk Berdasarkan NIK
- **Fitur 2** Tampilan Data Keluarga Beserta Daftar Anggotanya Berdasarkan Nomor KK
- **Fitur 3** Menambah Penduduk Baru Sebagai Anggota Keluarga
- **Fitur 4** Menambah Keluarga Baru
-
- **Fitur 7** Mengubah Status Kematian Penduduk
- **Fitur 8** Tampilan Data Penduduk Berdasarkan Kota/Kabupaten, Kecamatan, dan Kelurahan Tertentu