

Automation Performance Test

JMeter



Name: Dian Permana

Phone: +62 89603901473

Email: Permana.dian2012@gmail.com

Table of Contents

Automation performance Test menggunakan JMeter.....	5
0. Performance Testing.....	5
0.1 Tipe Performance Testing.....	5
0.2 Permasalah utama Performance Testing.....	6
0.3 KPI Performance Testing.....	6
1. Pengantar JMeter.....	7
1.1 Download JMeter.....	8
1.2 Install Java™.....	8
1.3 Install JMeter.....	9
1.4 Menambahkan Plugin di JMeter (Library Eksternal).....	9
1.5 Launch JMeter.....	9
1.5.1 Launch JMeter GUI Mode.....	10
1.5.2 Launch JMeter Command Line Mode.....	10
2. Mengenal Element JMeter.....	11
2.1 Thread.....	11
2.1.1 Thread Group.....	12
2.1.2 Concurrency Thread Group.....	14
2.2 Logic Controllers.....	15
2.2.1 Loop Controller.....	15
2.2.2. Module Controller.....	15
2.2.3. Switch Controller.....	15
2.2.4. Transaction Controller.....	16
2.2.5. ForEach Controller.....	16
2.3 Samplers.....	16
2.3.1. HTTP Request.....	16
2.3.2. Debug Sampler.....	18
2.4 Listeners.....	18
2.4.1 Graph.....	18
2.4.2 View Result Tree.....	19
2.4.3 Summary Report.....	19
2.5 Configuration Element.....	20
2.5.1. CSV Data Set Config.....	20
2.5.2. HTTP Header Manager.....	20
2.6 Assertions.....	21
2.6.1 Response Assertion.....	21
2.6.2 JSON Assertion.....	21
2.7 Processor.....	22
2.7.1 Post Processor.....	22
2.7.1.1 JSON Extractor.....	22
2.7.1.2 BeanShell PostProcessor.....	23
2.7.2. Pre Processor.....	23
2.8 Test Fragment.....	23
3. Definisi Test Plan dan Workbeanch.....	24
3.1 Menambahkan Elements.....	24
3.2 Menyimpan Test Plan.....	24
3.3 Menjalankan Test Plan.....	24
3.4 Menghentikan Test Plan.....	24
4. Menggunakan JMeter dengan Data+Keyword Driven untuk API Performance Test.....	25
4.1 Menambahkan Test Fragment.....	25
4.2 Menambahkan Switch Controller.....	26
4.3 Menambahkan Transaction Controller.....	26
4.4 Menambahkan HTTP Request (Login).....	27
4.5 Menambahkan CSV Data Set Config.....	27
4.6 Menambahkan JSON Extractor.....	28
4.7 Menambahkan JSON Asseration.....	28
4.8 Menambahkan Beanshell Post Processor.....	29

4.9 Menambahkan Debug Sampler.....	29
4.10 Menambahkan HTTP Request (Logout).....	30
4.11 Menambahkan HTTP Header Manager (Logout).....	30
4.12 Menambahkan Concurrency Thread Group.....	31
4.13 Menambahkan CSV data Set (Keyword Driven).....	31
4.14 Menambahkan ForEach Controller.....	32
4.15 Menambahkan elemente Controller.....	33
5. Menambahkan Report berbentuk Graph dan Table.....	33
5.1 Menambahkan Graph.....	33
5.2 Menambahkan View Results Tree.....	34
5.3 Menambahkan Summary Report.....	34
6. Analisis Data dan Pembahasan.....	35
6.1 Grafik Hasil.....	35
6.2 View Result Tree.....	37
6.3 Summary Report.....	39
6.4 Beanshell processor.....	39
7. Menghasilkan laporan dasboard.....	40
8. Diskusi dan Tanya Jawab.....	43
9. Kesimpulan.....	43
Refference.....	44
Thank You.....	44

Gambar 0.1 : Tipe Performance Testing.....	5
Gambar 0.2 : Pola grafik tipe Performance Testing.....	6
Gambar 1.1 : Halaman utama download JMeter.....	8
Gambar 1.2 : Cek versi Java melalui command prompt.....	8
Gambar 1.5.1 : Halaman utama JMeter versi 5.0.....	10
Gambar 2 : JMeter Elements.....	11
Gambar 2.1 : Konsep Thread pada JMeter.....	11
Gambar 2.1.1 : Panel kontrol Thread Group.....	13
Gambar 2.1.2 : Panel kontrol Concurrency Thread Group.....	14
Gambar 2.2.1 : Panel kontrol Loop Controller.....	15
Gambar 2.2.2 : Panel kontrol Module Controller.....	15
Gambar 2.2.3 : Panel kontrol Switch Controller.....	15
Gambar 2.2.4 : Panel kontrol Transaction Controller.....	16
Gambar 2.2.5 : Panel kontrol ForEach Controller.....	16
Gambar 2.3.1.1 : Panel kontrol Basic HTTP Request.....	17
Gambar 2.3.1.2 : Panel kontrol Advance HTTP Request.....	17
Gambar 2.3.2 : Panel kontrol Debug Sampler.....	18
Gambar 2.4.1 : Panel kontrol Graph Respone Times Over Time.....	18
Gambar 2.4.2 : Panel kontrol View Results Tree.....	19
Gambar 2.4.3 : Panel kontrol Summary Report.....	19
Gambar 2.5.1 : Panel kontrol CSV data Set Config.....	20
Gambar 2.5.2 : HTTP Header Manager.....	20
Gambar 2.6.1 : Panel kontrol Response asseration.....	21
Gambar 2.6.2 : Panel kontrol JSON assersation.....	22
Gambar 2.7.1.1 : Panel kontrol JSON Extractor.....	22
Gambar 2.7.1.2 : Panel kontrol BeanShell PostProcessor.....	23
Gambar 2.8 : Panel kontrol Test Fragment.....	23
Gambar 4.1 : Menambahkan Test Fragment.....	25
Gambar 4.2 : Menambahkan Switch Controller.....	26
Gambar 4.3 : Menambahkan Transaction Controller.....	26
Gambar 4.9 : Menambahkan Debug Sampler.....	29
Gambar 4.13.1 : Menambahkan CSV Data Set Config (Keyword Driven).....	31
Gambar 4.13.2 : File TC001_Login_Multiple_Account.csv.....	32
Gambar 4.14 : Menambahkan ForEach Controller.....	32

Gambar 4.15 : Menambahkan elemente Controller.....	33
Gambar 5.1 : Menambahkan Graph Active Threads Overtime.....	33
Gambar 5.2 : Menambahkan View Results Tree.....	34
Gambar 5.3 : Menambahkan Summary Report.....	34
Gambar 6.1.1 : Graph Response Times Over Time (Grafik Respons Waktu).....	35
Gambar 6.1.2 : Graph Response Times Over Time Hold Target Rate Time 100 detik (Grafik Respons Waktu).....	35
Gambar 6.1.3 : Graph Active Threads Over Time (Grafik aktif thread).....	36
Gambar 6.1.4 : Graph Active Threads Over Time Hold Target Rate Time 100 detik (Grafik aktif thread).....	36
Gambar 6.1.5 : Graph Transactions per Second (Grafik Transaksi per Detik).....	37
Gambar 6.1.6 : Graph Transactions per Second Hold Target Rate Time 100 detik (Grafik Transaksi per Detik).....	37
Gambar 6.2.1 : View Results tree Keyword.....	38
Gambar 6.2.2 : View Results tree Login-Logout.....	38
Gambar 6.3 : Summary Report.....	39
Gambar 7.1 : Folder bin JMeter.....	40
Gambar 7.2 : Open Cmd prom langsung di folder bin JMeter.....	41
Gambar 7.3 : Generate dasboar JMeter via CMD Prompt.....	41
Gambar 7.4 : View Results Generate dasboard JMeter.....	42
Gambar 7.5 : Generate dasboar JMeter berbentuk folder.....	42

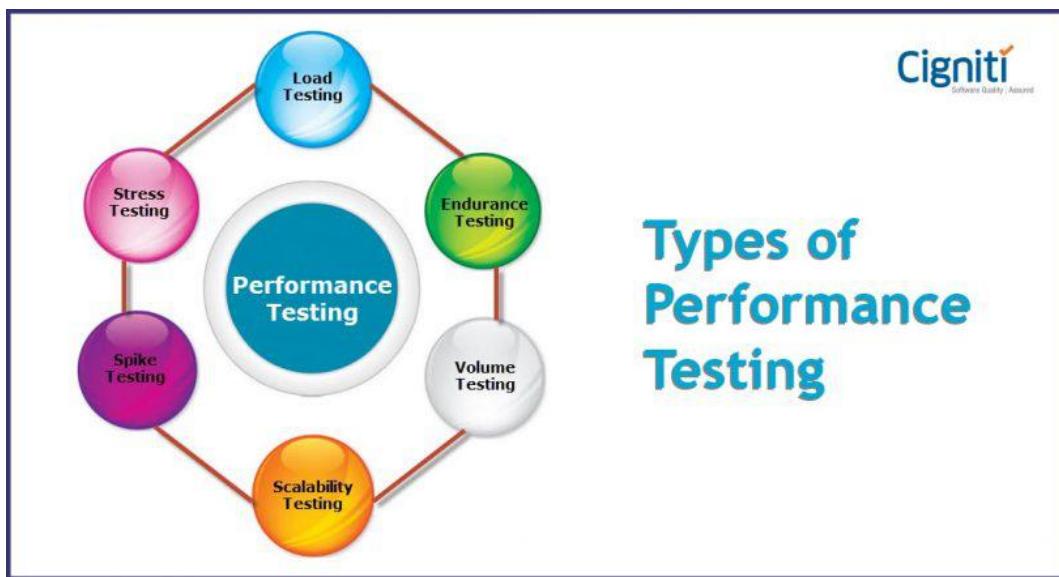
Automation performance Test menggunakan JMeter

0. Performance Testing

Performance testing adalah suatu bentuk pengujian perangkat lunak yang berfokus pada bagaimana suatu sistem yang menjalankan sistem berkinerja di bawah beban kerja yang di berikan/harapkan. Ini bukan tentang menemukan bug atau cacat perangkat lunak. Langkah langkah performance testing agar dilakukan sesuai dengan benchmark dan standar. Dan point penting dari dilakukannya performance testing adalah memberikan informasi kepada pengembang berdasarkan informasi diagnostik yang mereka butuhkan dari hasil performance testing yang telah dilakukan.

0.1 Tipe Performance Testing

Ada beberapa tipe performance testing berdasarkan teknik yang banyak digunakan, seperti terlihat pada gambar berikut:

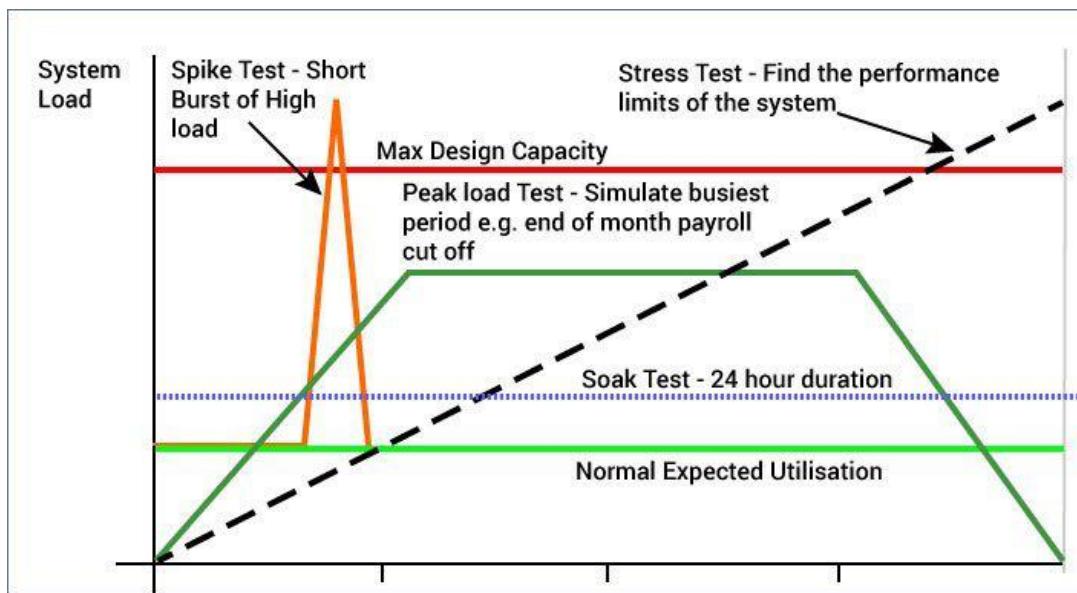


Gambar 0.1 : Tipe Performance Testing

- **Load testing** : Memeriksa kemampuan aplikasi untuk tampil di bawah beban pengguna yang diantisipasi. Tujuannya adalah untuk mengidentifikasi hambatan kinerja sebelum aplikasi perangkat lunak ditayangkan.
- **Stress testing** : Testing melibatkan sebuah aplikasi dibawah beban kerja ekstrim untuk melihat bagaimana menangani traffic tinggi atau memproses data. Tujuannya adalah untuk mengidentifikasi titik puncak suatu aplikasi dengan menggunakan lebih banyak users atau transaction.
- **Endurance testing** : atau di kenal dengan soak testing adalah testing ini dilakukan untuk memastikan perangkat lunak dapat menangani beban yang diharapkan dalam jangka waktu yang lama. Tujuan lainnya adalah untuk mengecek kebocoran memory (Kebocoran memori terjadi ketika suatu sistem gagal melepaskan memori yang dibuang. Kebocoran memori dapat merusak kinerja sistem atau menyebabkan nya gagal).
- **Spike testing** : Testing ini menguji reaksi perangkat lunak terhadap lonjakan besar tiba-tiba pada beban yang dihasilkan oleh pengguna.
- **Volume testing** : Testing ini utamnya ditujukan pada data base. Tujuannya adalah untuk memeriksa kinerja aplikasi perangkat lunak di bawah volume database yang bervariasi

- **Scalability testing** : Tujuan dari pengujian skalabilitas adalah untuk menentukan efektivitas aplikasi perangkat lunak dalam "meningkatkan / scaling-up" untuk mendukung peningkatan beban pengguna. Ini membantu merencanakan penambahan kapasitas untuk sistem perangkat lunak Anda.

Untuk mempermudah memahami antara tipe testing kita bisa melihat dari aspek visual grafik bagaimana testing ini berfungsi



Gambar 0.2 : Pola grafik tipe Performance Testing

Dengan pola garfik tipe performance testing ini kita bisa menganalisa berdasarkan point of view bagaimana performance testing ini dimaksudkan untuk tujuan yang berbeda antara tipe performance testing satu dengan yang lainnya. Dari banyaknya tipe performance testing yang ada dan masih banyak metode lainnya yang belum di jelaskan. Tetapi, point terpenting dari semua ini adalah bagaimana kita bisa melihat output kinerja dari aplikasinya dan memberikan informasi yang mudah dipahami oleh developer sehingga secara bersama sama bisa mengevaluasi hasil yang didapat dan meningkatkan performance aplikasi yang ada. bagaimanapun aspek ini hanya melihat bedasarkan metode yang secara 2 dimensi.

0.2 Permasalah utama Performance Testing

Utamanya semua permasalahan dari diadakanya sebuah performance testing adalah kecepatan, response time, load time, stabilitas dan skalabilitas. Permasalahan ini akan berdampak pada user yang tidak tertarik karena aplikasi dianggap terlalu lambat dalam memenuhi kebutuhannya. Maka dari itu performance testing merupakan aspek penting yang harus menjadi pertimbangan utama dalam memenuhi kebutuhan user. Salah satu aspek penting yang harus terpenuhi dalam performance testing adalah kita haru bisa memenuhi standar KPI's. harapan utamana hasil laporan performance testing ini nantinya akan bisa menjadi salah satu bahan evaluasi penting dalam bagaimana mengembangkan performance suatu aplikasi agar menjadi lebih baik lagi dalam aspek aspek permasalah yang ada.

0.3 KPI Performance Testing

KPI (Key Performance Indicator) adalah metrik yang memungkinkan pengukuran hasil dan sukses berdasarkan paramter yang relevan dan penting. dalam konteks load test, KPI menunjukkan pengukuran

pengguna dan lalu lintas untuk situs web atau aplikasi, untuk menentukan apakah dapat menahan sejumlah beban tertentu ke server backendnya. Setidaknya ada 7 KPI yang perlu diketahui :

- **Number of Users** : Untuk dapat meyakinkan website dapat menghandle banyak pengguna, kita bisa membuat simulasi aktivitas user berdasarkan account virtual seperti account nyata dalam melakukan aktivitas di sebuah website. Dengan mensimulasikan pengguna nyata, kita dapat menemukan hambatan yang dapat terjadi selama penggunaan harian real-time atau lonjakan lalu lintas.
- **Hit per Second** : Jumlah rata-rata sample yang diinisiasi perdetik. Dengan memahami hubungan antara Number of users dan Hit per Second, kita dapat mensimulasikan dan mengukur jenis dan beban penggunaan situs web (service), sehingga memvalidasi bahwa kemampuan kinerja situs memadai.
- **Error per Second** : di JMeter kita dapat melihat error per second, dari hal tersebut kita dapat mengetahui secara pasti dibagian mana yang harus di perbaiki. Sehingga analisa yang kita gunakan lebih bagus dalam memecahkan suatu masalah.
- **Respone Time** : Pengukuran ini mengukur jumlah waktu yang diambil dari byte pertama sebuah data sampai byte terakhir diterima oleh pengguna. Respone Time menunjukkan kinerja situs target dari sudut pandang pengguna, yang berarti berapa lama untuk menerima teks atau gambar yang diminta oleh tindakan pengguna.
- **Latency** : Pengukuran berapa lama dari sebelum mengirim permintaan sampai setelah menerima respon pertama. Latency membantu kita mengukur dan memahami keterlambatan jaringan yang melekat selama transmisi data dari klien ke server.
- **Connect Time** : Pengukuran berapa lama waktu yang dibutuhkan pengguna untuk terhubung ke server, dan server untuk merespons, termasuk jabat tangan SSL. Ini adalah bagian dari KPI Waktu Respons, tetapi penting untuk mengisolasi kinerja SSL sebagai hambatan.
- **Bytes/s (Throughput)** : Pengukuran konsumsi bandwidth rata-rata yang dihasilkan oleh tes per detik. Ini mengukur jumlah data yang mengalir ke dan dari server. Mengisolasi KPI ini penting untuk memastikan pengontrol antarmuka jaringan Anda bekerja dengan baik.

Masih banyak aspek lain yang bisa menjadi KPI, hanya saja karena disini kita akan menggunakan JMeter sebagai tools utama, maka kita fokus pada aspek penting yang terdapat di output JMeter sebagai bagian dari KPI.

1. Pengantar JMeter

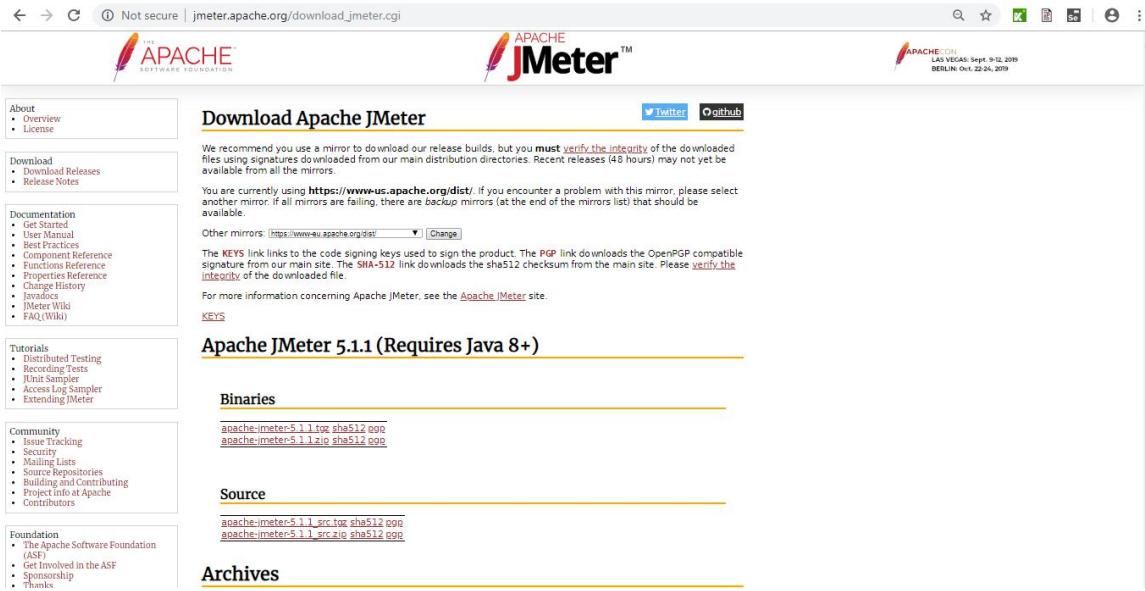
Aplikasi Apache JMeter™ adalah perangkat lunak open source, aplikasi Java 100% murni yang dirancang untuk memuat fungsional uji dan mengukur kinerja suatu aplikasi. Awalnya dirancang untuk menguji Aplikasi Web. JMeter sendiri dikembangkan oleh developer dari Apache Software Foundation dibawah license Apache License 2.0. Untuk memahami lebih detail mengenai informasi yang ada bisa mengunjungi situs resmi <https://JMeter.apache.org/>

JMeter sendiri saat ini sudah release sampai pada versi 5.1 pada tanggal 19/02/2019 menggunakan java 8+. Sama halnya dengan beberapa software lain, JMeter memiliki Plugins yang terpisah. JMeter Plugin adalah proyek independen untuk Apache JMeter. Setiap plugin memiliki tujuan yang berbeda dan mempercepat proses pembuatan dan pelaksanaan rencana uji JMeter. Pengguna dapat menginstal plugin melalui plugin manager (online) atau secara manual (Offline) dengan menambahkannya kebagian lib (lihat 1.4).

Sebagaimana software open source lainnya, kita bisa secara leluasa menggunakan software ini untuk kepentingan pendidikan, riset, profesional pekerjaan, maupun hal lainnya tanpa harus melanggar (membajak) baik itu secara hukum ataupun etika dalam keberlangsungan ekosistem suatu software.

1.1 Download JMeter

Untuk menggunakan JMeter kita bisa mengunjungi situs https://JMeter.apache.org/download_JMeter.cgi dan mendownloadnya.



Gambar 1.1 : Halaman utama download JMeter

Pilih versi terbaru yang bisa digunakan, dan perhatikan juga versi operating sistem (OS) yang digunakan apakah 32 atau 64 bit, karena seperti halnya kita tahu bahwa hal ini akan bedampak pada proses install software.

1.2 Install Java™

Karena JMeter merupakan sebuah aplikasi yang berbasis java, kita memerlukan dukungan dari Java SE Development Kit. Apabila sebelumnya kita sudah pernah install java, kita bisa melakukan pengecekan apakah java sudah terinstall dengan baik melalui command prompt (Windows + R > cmd), ketikkan perintah : java -version seperti berikut :

A screenshot of a Windows Command Prompt window titled "cmd C:\windows\system32\cmd.exe". The window displays the following text:

```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\pqqa04>java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)

C:\Users\pqqa04>
```

Gambar 1.2 : Cek versi Java melalui command prompt

Disini kita bisa melihat versi java yang telah terinstall di komputer. Jika masih menggunakan versi yang lama bisa melakukan update terlebih dahulu. Apabila belum pernah terinstall java sebelumnya, kita bisa

mengunjungi halaman berikut [ini](#) untuk melakukan download dan melakukan proses install java secara bertahap.

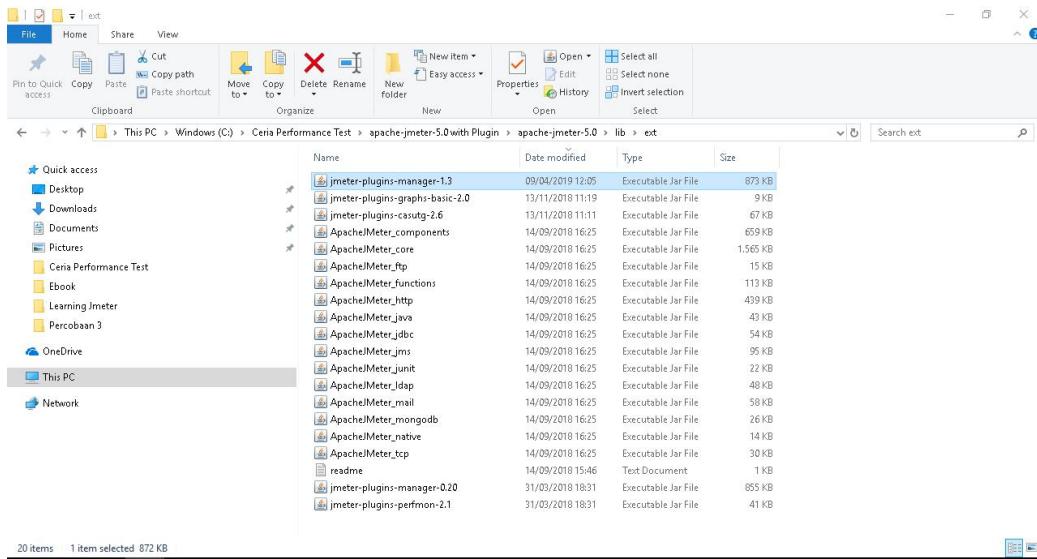
1.3 Install JMeter

Sangat mudah untuk menginstall JMeter, kita hanya perlu menyiapkan Folder khusus untuk menyimpan File download JMeter misalkan kita akan menyimpannya di : C:\Program Files\JMeter, kemudian Extract Folder "apache-jmeter-5.0.zip" dan selesai.

1.4 Menambahkan Plugin di JMeter (Library Eksternal)

Untuk menambahkan plugin kita bisa menggunakan beberapa pilihan, bisa dengan fitur online melalui Plugin Manager maupun offline yaitu dengan mengikuti beberapa langkah berikut :

- Download library eksternal melalui : <https://JMeter-plugins.org/wiki/Start/>
- Extract library hasil download lalu tambahkan di folder dimana JMeter disimpan sebagai contoh C:\Program Files\JMeter\apache-JMeter-5.0\lib\ext
- Pastikan JMeter untuk di close terlebih dahulu, lalu buka kembali JMeter dan cek apakah sudah tersedia atau belum.



Gambar 1.4 : Menambahkan Library Eksternal

Tiga File teratas merupakan plugin eksternal yang telah berhasil di tambahkan yaitu untuk thread group (Concurrency, Ultimate, Stepping) grafik (Active, Response, Transaction), dan plugin manager (update via online).

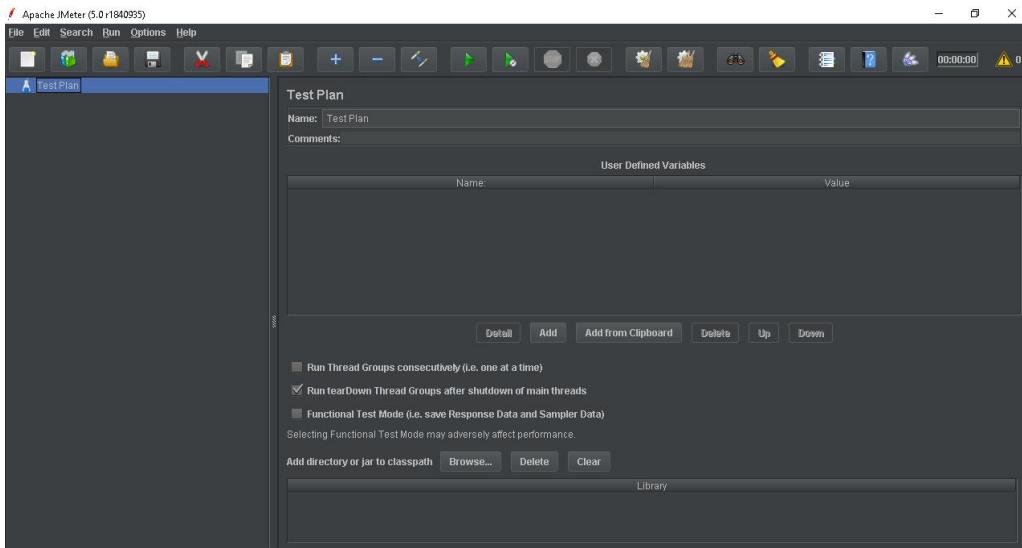
1.5 Launch JMeter

Setidaknya ada beberapa cara yang bisa pilih untuk bisa menjalankan JMeter :

- GUI Mode
- Command Line Mode
- Server Mode

1.5.1 Launch JMeter GUI Mode

Jika kita menggunakan JMeter di windows (7/8/10) maka kita akan mudah untuk memulainya yaitu dengan cara menjalankan file **/bin/jmeter.bat** maka akan muncul halaman utama JMeter mode GUI sebagai berikut :



Gambar 1.5.1 : Halaman utama JMeter versi 5.0

Untuk memahami lebih detail mengenai mode GUI ini, kita bisa memulai untuk mencoba beberapa fitur yang tersedia, sama hal nya dengan software yang lain mode ini sangat mudah untuk kita pelajari karena dibuat secara sederhana dan familiar seperti kebanyakan software lainnya.

1.5.2 Launch JMeter Command Line Mode

Command line hanya bisa digunakan jika kita telah berhasil membuat skenario testing melalui GUI Mode, hal ini karena JMeter hanya melakukan proses running via command line pada file.jmx. Untuk bisa memulai di mode ini kita perlu mengetahui perintah perintah khusus yang disediakan oleh JMeter agar aplikasi dapat memprosesnya. Berikut perintah yang digunakan untuk run JMeter via command line:

```
$jmeter -n -t Ebook.jmx -l log.jtl -H 127.0.0.1 -P 8000
```

Keterangan :

- JMeter -n : Spesifik JMeter untuk bisa run via Command line Mode
- t Ebook.jmx : Nama File yang berisi testplant (.jmx). (Tambahkan direktori penyimpanan)
- l log.jtl : Log File untuk menyimpan hasil running bisa berbentuk .jtl maupun .csv (Tambahkan direktori penyimpanan)
- H 127.0.0.1 -P 8000 : Proxy Server, Host name dan Port (Aplikasi).

Berikut contoh hasil running via Command line mode yang pernah di coba:

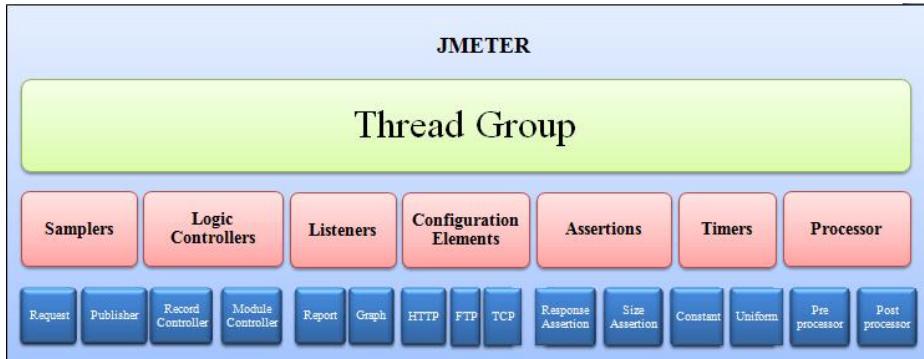
```
Select C:\Windows\System32\cmd.exe
C:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>jmeter -n -t C:\DATA\Dian\Ceria_Ebook.jmx -l C:\DATA\Dian\log_Ebook.csv -H 172.18.102.86 -P 8188
Creating summariser <summary>
Created the tree successfully using C:\DATA\Dian\Ceria_Ebook.jmx
Starting the test @ Thu Apr 25 18:26:28 ICT 2019 (1556162786954)
Waiting for possible Shutdown/StopTestNow/Heapdump message on port 4445
summary + 209 in 00:00:03 = 71.8/s Avg: 0 Min: 0 Max: 1 Err: 0 (0.00%) Active: 200 Started: 200 Finished: 200
summary + 182 in 00:00:06 = 31.8/s Avg: 0 Min: 0 Max: 1 Err: 0 (0.00%) Active: 0 Started: 200 Finished: 200
summary = 391 in 00:00:09 = 45.2/s Avg: 0 Min: 0 Max: 1 Err: 0 (0.00%)
Tidying up ... @ Thu Apr 25 18:26:35 ICT 2019 (1556162795978)
... end of run
```

Gambar 1.5.2 : Running via Command line Mode

Perlunya kita membandingkan hasil data yang dirunning via GUI dan command line mode, karena bisa saja terdapat perbedaan hasil data.

2. Mengenal Element JMeter

Untuk menggunakan JMeter kita perlu mengenal beberapa Element yang sering digunakan, hal ini menyesuaikan dengan kebutuhan testing yang akan dilakukan. Berikut ilustrasi mengenai element di JMeter.

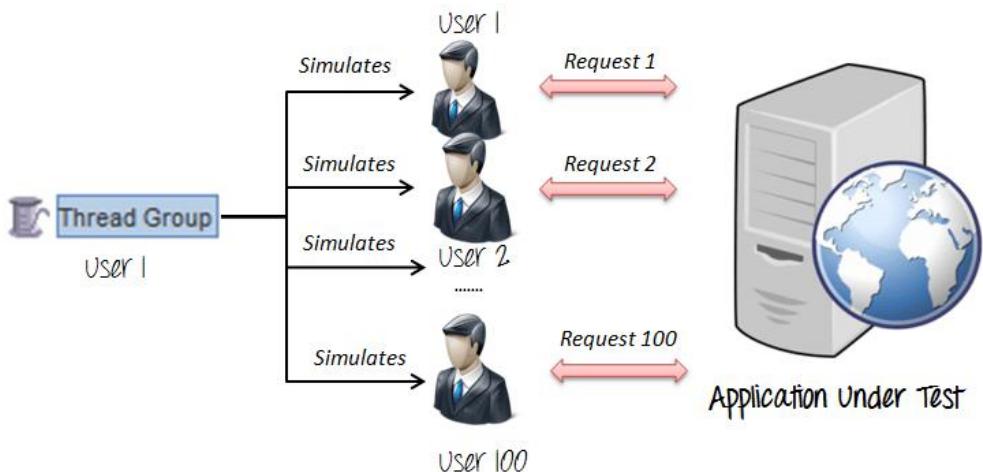


Gambar 2 : JMeter Elements¹

Di tulisan ini kita tidak menggunakan semua element yang tersedia di JMeter, hanya akan membahasa dan menggunakan beberapa element yang dibutuhkan.

2.1 Thread

Thread menentukan kelompok pengguna yang akan menjalankan uji kasus tertentu terhadap server. Kita dapat mengontrol jumlah pengguna yang disimulasikan (Number of thread/ Jumlah pengguna), waktu ramp up (berapa lama untuk memulai semua pengguna), berapa kali melakukan tes.



Gambar 2.1 : Konsep Thread pada JMeter²

Kita akan mempelajari secara sederhana bagaimana konsep thread bekerja. JMeter sendiri memiliki sebuah fitur yang tidak dimiliki oleh tools lain seperti katalon studio dimana konsep kerja ini kita bisa

¹ <https://www.guru99.com/JMeter-element-reference.html>

² <https://www.guru99.com/JMeter-element-reference.html>

mensimulasikan sebuah percobaan secara langsung dan melakukan sebuah proses secara bersamaan (Parallel).

Contoh : kita akan melakukan sebuah aktivitas login bersama-sama secara langsung dengan account yang sama (100 user account), maka kita bisa melakukan hal tersebut dengan setting thread group. Lalu bagaimana kita tahu bahwa hal ini bekerja secara bersama-sama dalam satuan waktu yang sama ? tentu kita bisa melihat hal tersebut pada bagian “sample results” yang ada pada view results tree yang telah dilakukan ujicoba seperti terlihat dibawah ini :

```
Thread Name: Login 1-57
Sample Start: 2019-04-23 08:47:21 ICT
Load time: 3718
Connect Time: 0
Latency: 0
Size in bytes: 6285
Sent bytes: 2664
Headers size in bytes: 939
Body size in bytes: 5346
Sample Count: 1
Error Count: 0
Data type ("text"|"bin"|""):
Response code: 200
Response message: Number of samples in transaction : 3, number of failing samples : 0
```

Untuk bisa membuktikan apakah thread bekerja bersama, kita setting thread group pada JMeter sebanyak 100 user account, maka seharusnya 100 user account tersebut akan dikirim secara bersama pada satuan waktu yang sama. Artinya akan ada **Sample Start: 2019-04-23 08:47:21 ICT** yang sama sebanyak 100 user account tetapi dengan **Thread Name: Login 1-57** yang berbeda (1-100).

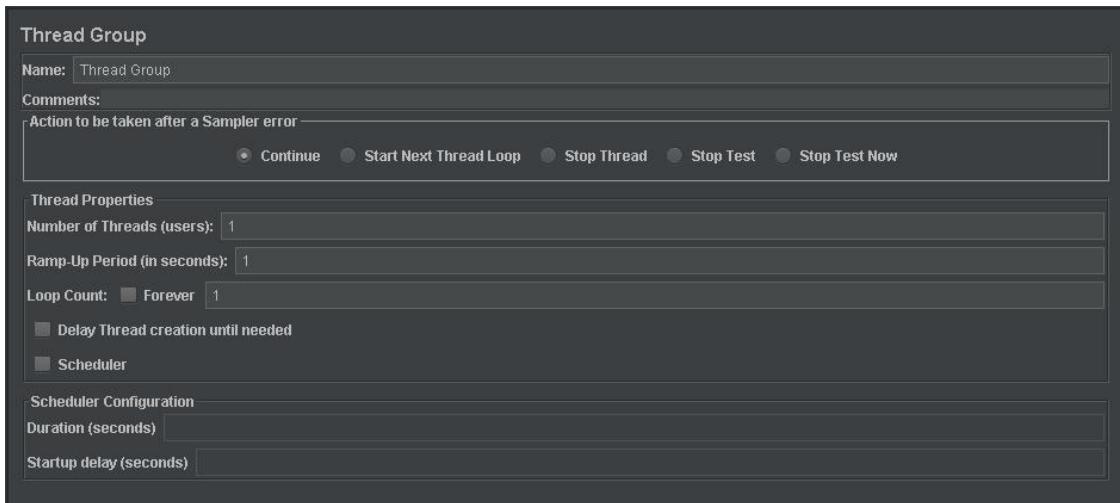
“Well, memang berdasarkan hasil ujicoba yang telah dilakukan, banyak aspek penting yang harus menjadi pertimbangan besar seperti spesifikasi Komputer/PC/Server dimana JMeter di install, trafic intranet/internet yang digunakan, banyaknya data yang di proses, Server dimana aplikasi di install, dan masih banyak hal lain. Namun secara hasil uji coba yang telah dilakukan penulis berpegang pada “jumlah total sample” yang didapat harus sesuai dengan teori dasar perhitungan matematis yang ada, seperti yang akan kita bahas dibagian thread group.

Terdapat beberapa thread yang bisa digunakan di JMeter salah satu yang akan dibahas adalah thread group dan Concurrency thread group. Ada berbagai keunggulan masing masing dari element ini sesuai dengan kebutuhan testing.

2.1.1 Thread Group

Konsep sederhana thread group telah di jelaskan sebelumnya, dimana kita harus benar benar memahami secara mendasar bagaimana konsep ini bekerja. JMeter menjalankan thread group sampai jumlah looping tercapai atau durasi / waktu akhir tercapai. Yang mana yang terjadi lebih dulu. Perhatikan bahwa kondisi hanya diperiksa antara sampel; ketika kondisi akhir tercapai, thread itu akan berhenti. JMeter tidak menginterupsi sampler yang sedang menunggu tanggapan, sehingga waktu akhir dapat ditunda secara bebas. Yang menjadi pebedaan mendasar pada thread group ini dengan concurrency thread group akan terlihat pada active thread overtime (meskipun number of thread di set lebih dari 1) hanya akan ada 1 thread yang aktif berbeda dengan

concurrency thread group dimana akan aktif secara bersama-sama. Hal yang terpenting dari thread group ini adalah kita harus menghitung dengan persamaan matematik agar mendapatkan perhitungan yang ideal.



Gambar 2.1.1 : Panel kontrol Thread Group

- **Number of Thread (Users)** : Mensimulasikan jumlah pengguna atau koneksi ke aplikasi server Anda
- **Ramp-Up Period (in seconds)** : Hal ini memberitahu seberapa lama JMeter untuk mengambil sampai selesai seluruh jumlah thread.
- **Loop Count**: Ini merupakan jumlah dari eksekusi untuk skrip.
- **Delay Thread Creation Until Needed**: Jika opsi ini dicentang, penundaan ramp-up dan penundaan startup dilakukan sebelum data thread dibuat. Jika tidak dicentang, semua data yang diperlukan untuk thread dibuat sebelum memulai pelaksanaan tes.
- **Scheduler**: Ini menjadwalkan tes. Anda dapat mengatur durasi khusus dan penundaan startup untuk membuat thread di bagian ini.

Kita bisa menggunakan persamaan matematik berikut untuk mendapatkan perhitungan delay antara user account satu dengan yang lainnya :

$$\text{delay between user account} = \frac{\text{Number of thread}}{\text{Ramp - up}}$$

Untuk mendapatkan Total Jumlah sample, maka :

$$\text{Sample} = \text{Number of thread} \times \text{Loopcount}$$

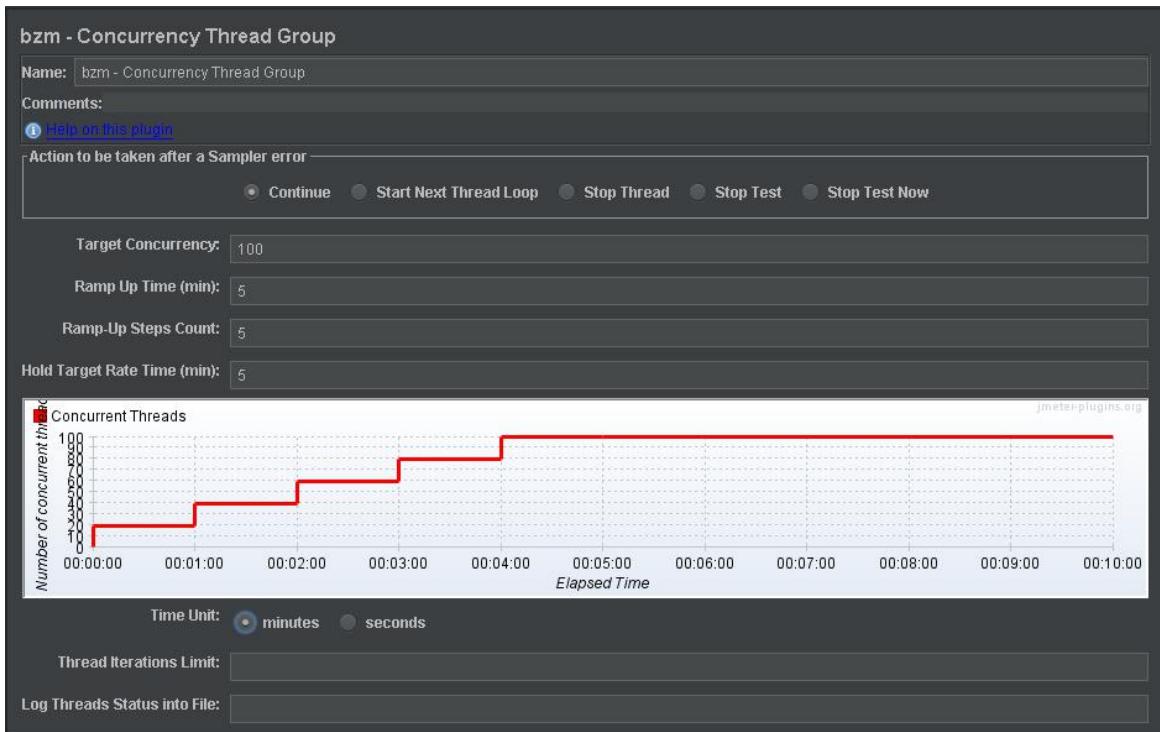
Untuk mempermudah memahaminya, kita bisa menggunakan contoh kasus sebagai berikut : Misalkan kita memiliki number of thread 100 user account dengan 100 ramp-up periode dan Loop count 6 berapa delay antara 1 user account dengan user account lainnya, dan berapa total jumlah sampel yang harus didapat oleh JMeter ?

Penyelesaian :

Maka dengan persamaan matematik kita akan mendapatkan : Setiap 1 detik ($100/100$) 6 thread / request akan di hit ke server, dari sini juga kita akan mendapatkan total waktu selama mengeksekusi thread tersebut yaitu 100 detik (00.01.40). Dan total jumlah sample yang harus di dapat adalah 600 (100×6) [Cek Threadgroup 100_100_6]

2.1.2 Concurrency Thread Group

Concurrency thread group merupakan element dari library eksternal, berbeda dengan thread group, element ini kita bisa menggunakan thread group secara bersamaan dalam satuan waktu yang sama tanpa adanya delay ketika melakukan hit.



Gambar 2.1.2 : Panel kontrol Concurrency Thread Group

Gambar 2.1.2 kita bisa menganalisa bahwa Setiap 1 menit, 20 pengguna akan ditambahkan hingga mencapai 100 pengguna. (5 menit dibagi 5 langkah sama dengan 1 menit per langkah. 100 pengguna dibagi 5 langkah sama dengan 20 pengguna per langkah. Total 20 pengguna setiap 1 menit). Setelah mencapai 100 thread, semuanya akan terus berjalan dan hit server bersama selama 5 menit

- **Target Concurrency:** Jumlah pengguna bersamaan yang harus di pertahankan setelah ramp-up.
- **Ramp Up Time:** Jangka waktu yang dibutuhkan untuk mencapai target tingkat target bersama.
- **Ramp Up Steps:** Ini mengacu pada rincian tingkat kenaikan bersama. Langkah-langkah lainnya menghasilkan pola yang lebih halus. Mengatur bidang ini ke 0 berarti bahwa ramp up akan mulus, dan thread akan mengalami penundaan yang seragam di antara keduanya.
- **Hold Target Rate Time:** Durasi untuk mempertahankan kebersamaan target sebelum mulai secara bertahap mematikan semua thread.
- **Thread Iterations Limit:** Membatasi jumlah iterasi.

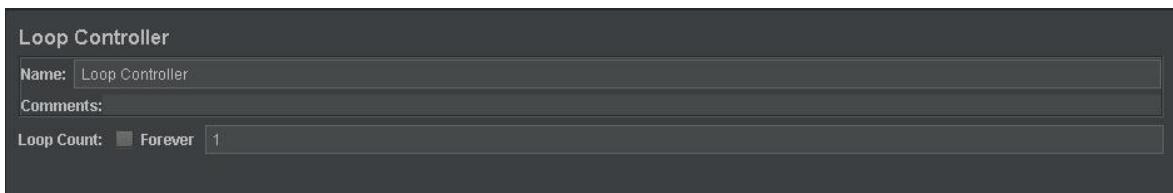
Concurrency thread group cocok untuk skenario berorientasi tujuan, tetapi tujuannya di sini adalah untuk memiliki kontrol atas jumlah pengguna bersama-sama, untuk menggunakan concurrency thread group kita perlu memahami konsep awal thread group bekerja, karena concurrency thread group merupakan pengembangan dari thread group default yang ada di JMeter.

2.2 Logic Controllers

Logic Controllers menentukan urutan di mana samplers diproses. Terdapat beberapa logic controller yang sering digunakan diantaranya sebagai berikut :

2.2.1 Loop Controller

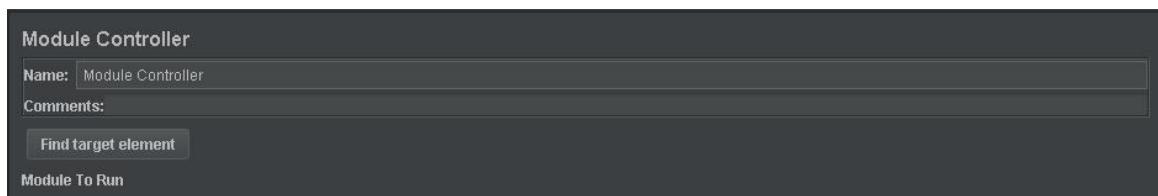
JMeter akan mengulanginya beberapa kali, selain nilai loop yang telah ditentukan untuk thread group. Misalnya, jika kita menambahkan satu permintaan HTTP ke Loop Controller dengan hitungan loop sama dengan 2, dan mengkonfigurasi jumlah loop thread group sama dengan 3, JMeter akan mengirimkan total $2 * 3 = 6$ Permintaan HTTP.



Gambar 2.2.1 : Panel kontrol Loop Controller

2.2.2. Module Controller

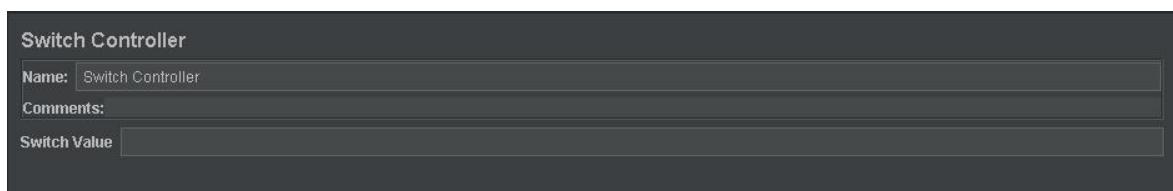
Module Controller menyediakan sebuah mekanisme untuk mengganti test plan fragment kedalam test plan yang sedang berjalan. Dimana sebuah Fragment terdiri dari sebuah controller dan semua test element. Selain itu, fragment juga dapat digunakan didalam sebuah thread group.



Gambar 2.2.2 : Panel kontrol Module Controller

2.2.3. Switch Controller

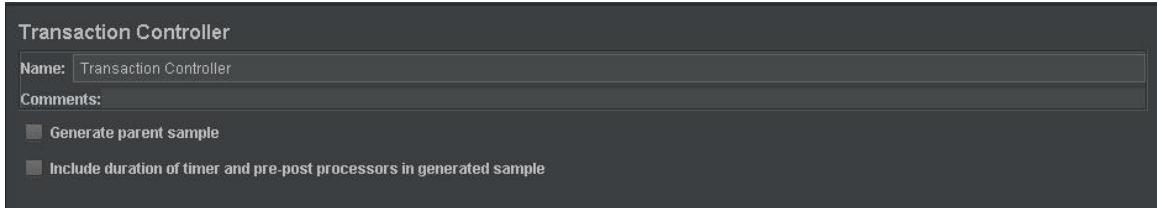
Switch Controller bertindak seperti Interleave Controller karena menjalankan salah satu elemen bawahan pada setiap iterasi, tetapi alih-alih menjalankannya secara berurutan, controller menjalankan elemen yang ditentukan oleh nilai switch.



Gambar 2.2.3 : Panel kontrol Switch Controller

2.2.4. Transaction Controller

Transaction Controller menghasilkan sampel tambahan yang mengukur keseluruhan waktu yang diperlukan untuk melakukan elemen uji.



Gambar 2.2.4 : Panel kontrol Transaction Controller

2.2.5. ForEach Controller

Melalui nilai-nilai dari set variabel terkait. Ketika kita menambahkan sampler (atau pengontrol) ke pengontrol ForEach, setiap sampel (atau pengontrol) dieksekusi satu kali atau lebih, di mana selama setiap loop variabel memiliki nilai baru. Input harus terdiri dari beberapa variabel, masing-masing diperluas dengan garis bawah dan angka. Setiap variabel tersebut harus memiliki nilai. Jadi misalnya ketika variabel input memiliki nama inputVar, variabel berikut seharusnya sudah didefinisikan:



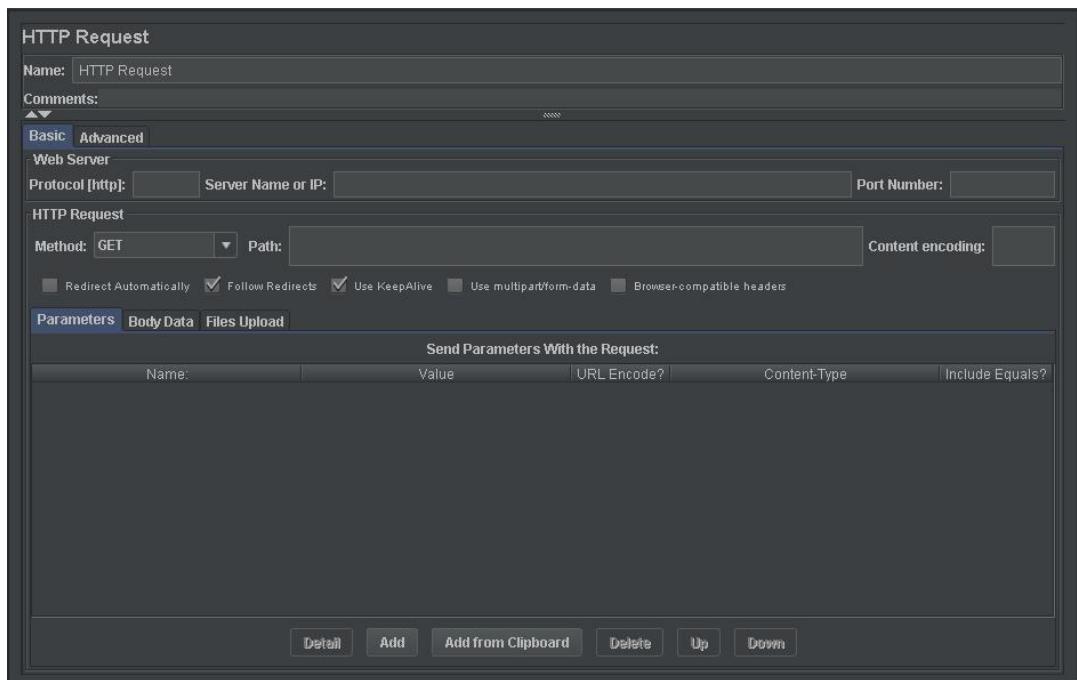
Gambar 2.2.5 : Panel kontrol ForEach Controller

2.3 Samplers

Samplers melakukan pekerjaan JMeter yang sebenarnya. Setiap sampler (kecuali aksi kontrol aliran) menghasilkan satu atau lebih hasil sampel. Hasil sampel memiliki berbagai atribut seperti sukses / gagal, waktu yang berlalu, ukuran data dll. Hasil samplers ini dapat dilihat dengan menggunakan “view results tree”.

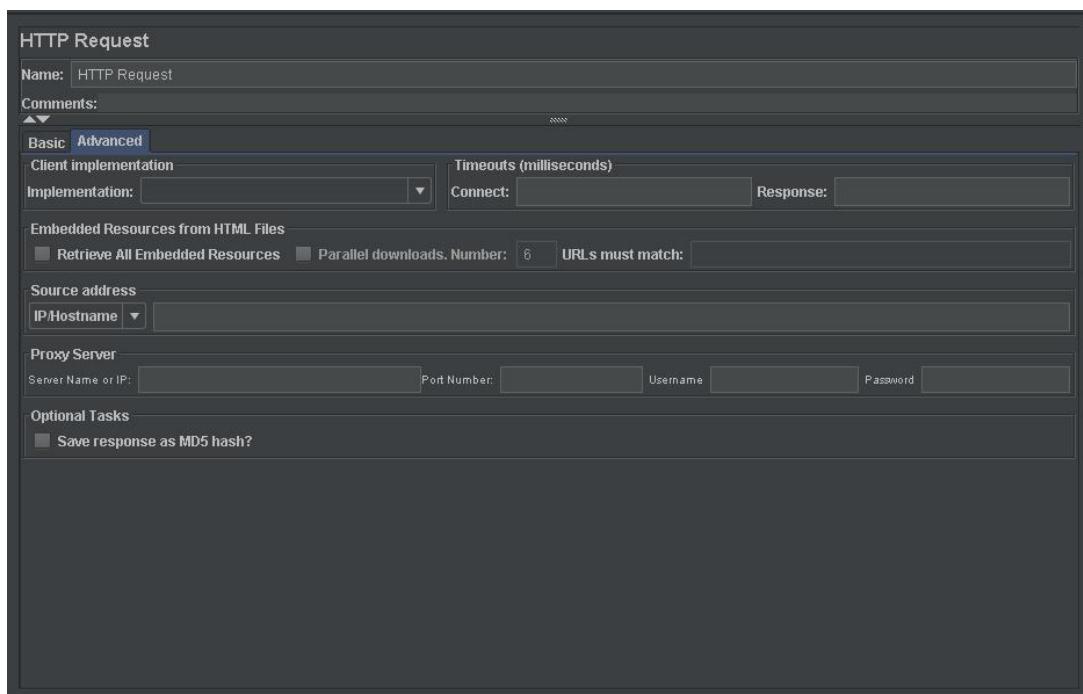
2.3.1. HTTP Request

Sampler Hypertext Transfer Protocol (HTTP) Request dirancang untuk memungkinkan komunikasi antara klien dan server. HTTP berfungsi sebagai protokol permintaan-respons antara klien dan server. Untuk dapat menggunakan element ini kita harus tahu secara detail mengenai konsep Restfull API (Application Programming Interface) dimana kita harus mengetahui 4 konsep dasar seperti : url design, Http verbs (Method, IP Addres, path, parameter, body data), Http Response (Kode standar dalam menginformasikan hasil request kepada client) code and Format response (XML, JSON)



Gambar 2.3.1.1 : Panel kontrol Basic HTTP Request

Secara fungsi JMeter sudah memiliki fitur standar yang bisa digunakan untuk Restfull API sama seperti tools lainnya. Kita bisa menggunakan fitur tersebut setalah memahami ke empat konsep dasar seperti yang dijelaskan diatas.



Gambar 2.3.1.2 : Panel kontrol Advance HTTP Request

Salah satu bagian penting dari Panel kontrol di JMeter ini terbilang sangat lengkap mulai dari parameter level basic - advance. Untuk advance setting kita bisa menambahkan timeout (milisecond) baik untuk connect maupun response.

2.3.2. Debug Sampler

Debug Sampler menghasilkan sebuah sample yang berisi sebuah nilai dari semua variabel JMeter dan atau properties. Untuk hasil debug, kita dapat melihatnya melalui “view result tree”.



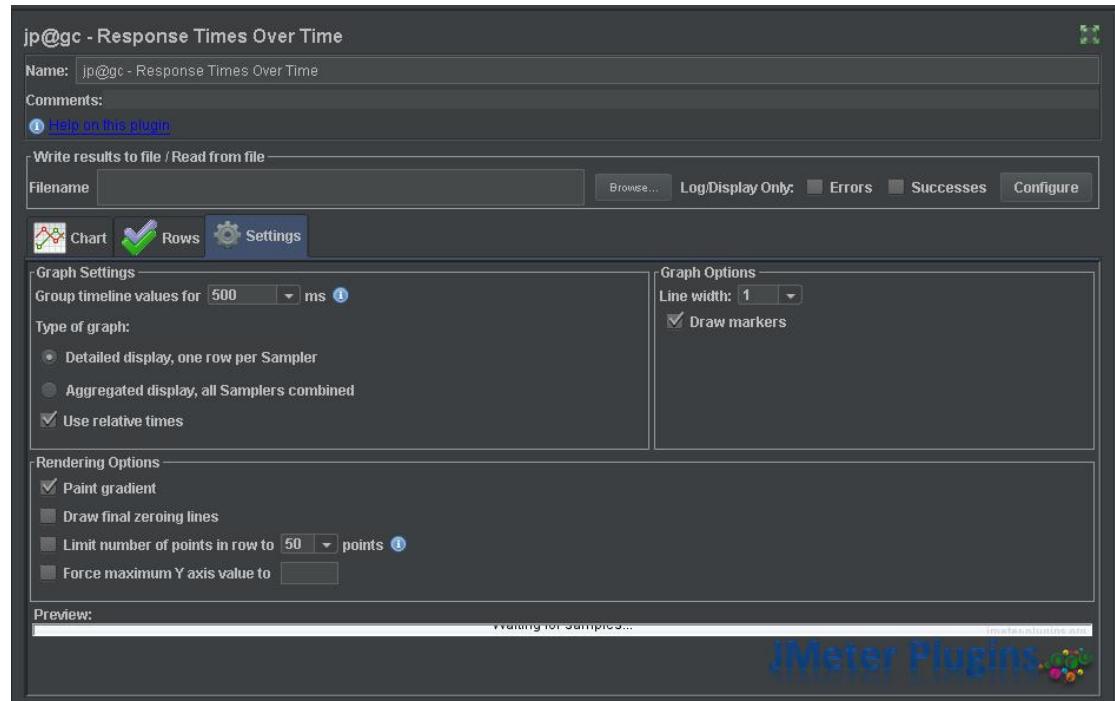
Gambar 2.3.2 : Panel kontrol Debug Sampler

2.4 Listeners

Listener memberikan akses ke informasi yang dikumpulkan JMeter tentang kasus uji sementara ketika JMeter berjalan. Terdapat beberapa element yang sering digunakan didalam listener.

2.4.1 Graph

Untuk dapat mengilustrasikan dalam bentuk grafik, JMeter menyediakan plot data result dengan element Graph. Terdapat beberapa tipe graph yang bisa digunakan salah satunya yaitu active thread overtime, transaction per second dan Response time over time yang telah di tambahkan dari eksternal libray sebelumnya.

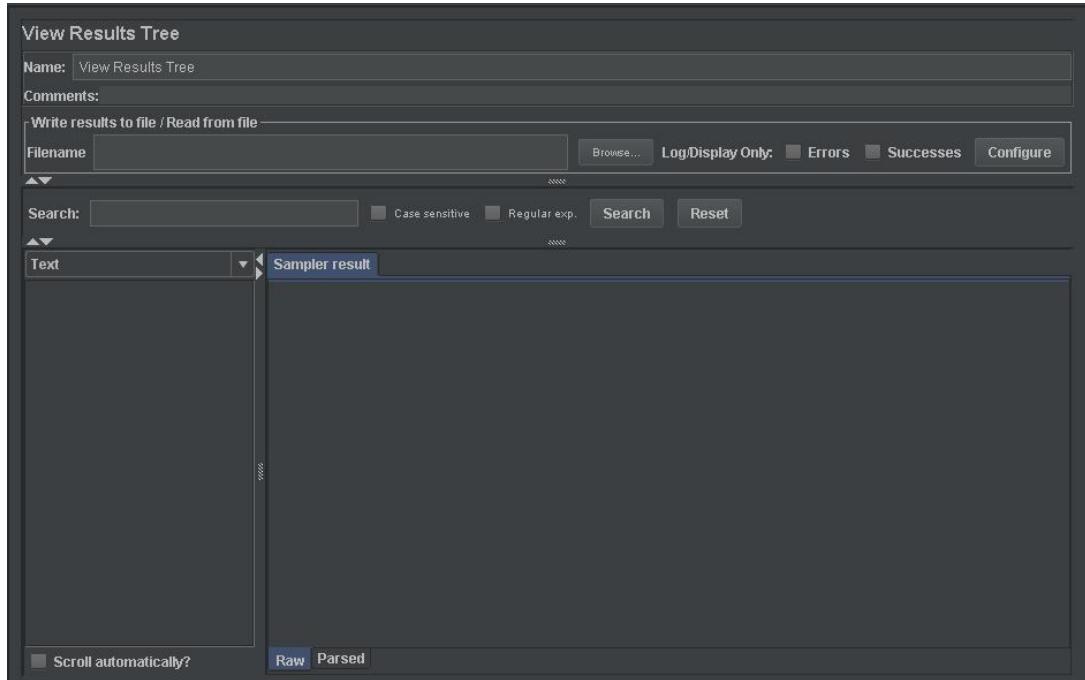


Gambar 2.4.1 : Panel kontrol Graph Response Times Over Time

Kita bisa melakukan setting pada panel kontrol tersebut menyesuaikan dengan kebutuhan yang ada, seperti terlihat pada Gambar 2.4.1

2.4.2 View Result Tree

View results tree menunjukkan semua sampel respon, mengizinkan kita untuk melihat response untuk sebuah sampel. Sebagai tambahan kita dapat melihat waktu untuk mendapatkan response ini dan beberapa response code lainnya.



Gambar 2.4.2 : Panel kontrol View Results Tree

Sebagai catatan tambahan View Result tree lebih efektif dan efisien hanya untuk functional testing atau test plan debug dan validasi, hal ini karena mengkonsumsi banyak resource seperti Memory dan CPU.

2.4.3 Summary Report

Summary Report membuat baris tabel untuk setiap permintaan dengan nama berbeda dalam pengujian. Seluruh proses testing yang dilakukan secara lebih sederhana dapat dilihat dibagian ini.

Summary Report											
Name:	Summary Report										
Comments:											
Write results to file / Read from file											
Filename											
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes	
TOTAL	0	0	9223372036...	-922337203...	0.00	0.00%	.0/hour	0.00	0.00	.0	

Gambar 2.4.3 : Panel kontrol Summary Report

Kita dapat melihat hasil seluruh aktivitas sampler baik berupa total sample, error, throughput, rata-rata transaksi, ukuran, dan dapat menyimpannya dalam format file dengan ekstensi.csv

2.5 Configuration Element

Configuration Element dapat digunakan untuk mengatur default dan variabel untuk digunakan oleh sampler. Perhatikan bahwa elemen-elemen ini diproses pada awal lingkup di mana mereka ditemukan, yaitu sebelum sampler dalam lingkup yang sama.

2.5.1. CSV Data Set Config

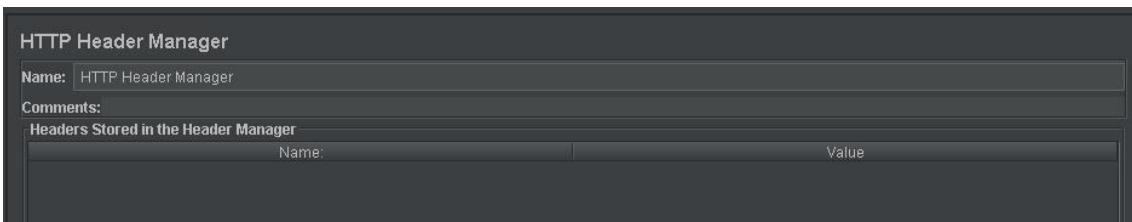
CSV Data Set Config digunakan untuk membaca line dari sebuah file. Dan memisahkannya kedalam variabel. Hal ini akan bermanfaat dalam sebuah testing menggunakan data random dan unik dalam jumlah yang sangat besar.



Gambar 2.5.1 : Panel kontrol CSV data Set Config.

Ketika sebuah variable sudah sangat banyak, maka element ini akan sangat bermanfaat sekali untuk digunakan agar mempermudah kita untuk mendefinisikan variabel dalam bentuk tabel sehingga mempermudah kita untuk membaca data. Ada beberapa set config yang perlu di perhatikan didalam element ini agar bisa digunakan sebagaimana semestinya.

2.5.2. HTTP Header Manager



Gambar 2.5.2 : HTTP Header Manager.

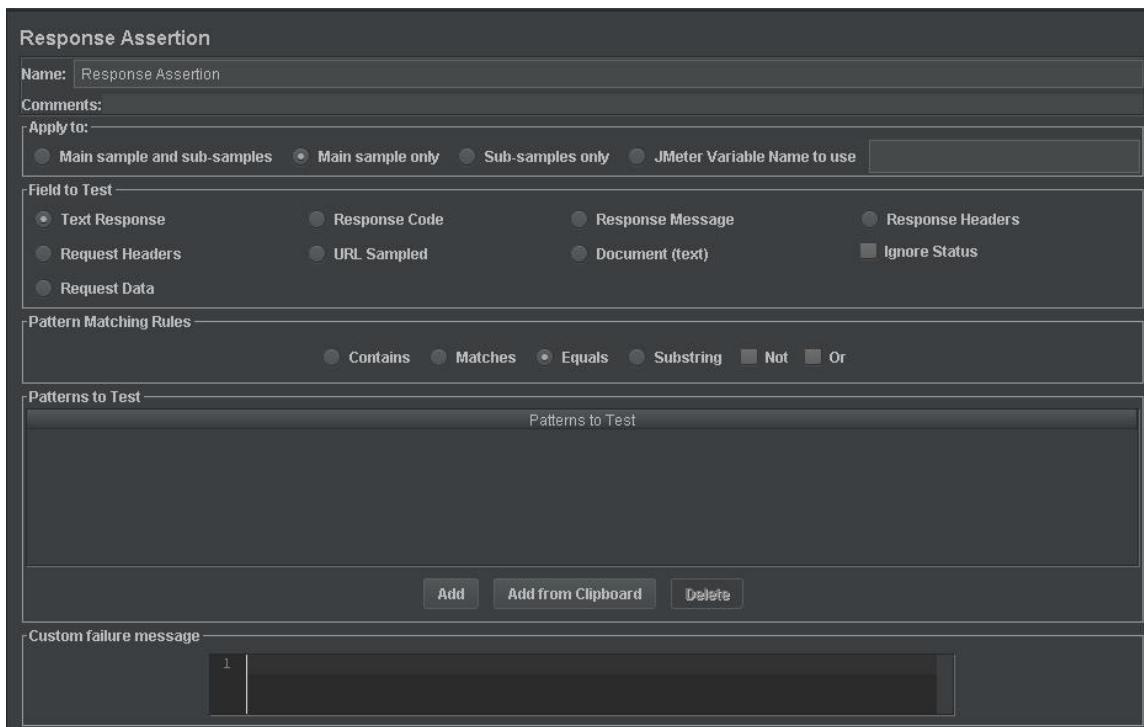
Http header Manager biasanya digunakan untuk menyimpan data dalam sebuah header yang dianggap sebagai konstanta yang tidak berubah, dalam hal ini kita akan menggunakan untuk mengambil sebuah refresh token yang didapatkan dari JSON response Login dan digunakan untuk Logout.

2.6 Assertions

Assertions digunakan untuk melakukan pemeriksaan tambahan pada sampler, dan diproses setelah setiap sampler dalam cakupan yang sama. Untuk memastikan bahwa Penegasan hanya diterapkan pada sampler tertentu

2.6.1 Response Assertion

Respon assertion memungkinkan kita untuk menambahkan pola string untuk dibandingkan dengan berbagai bidang atau respon, sebagai contoh yang sederhana untuk digunakan adalah Contains, Matches: Perl5-style regular expressions dan Equals, Substring: teks biasa, peka huruf besar-kecil dan



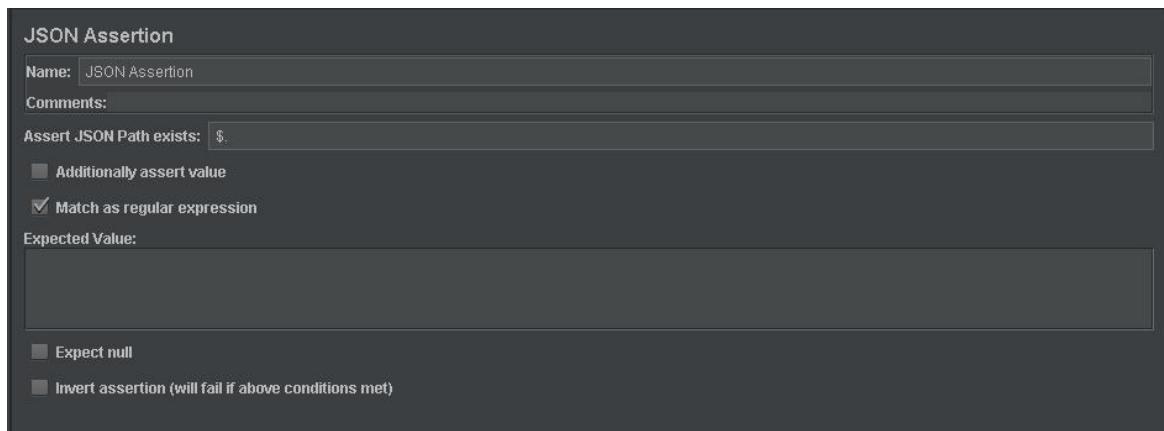
Gambar 2.6.1 : Panel kontrol Response asseration

2.6.2 JSON Assertion

Element ini memungkinkan kita untuk melakukan validasi/verifikasi dari sebuah JSON respon, flow proses kerja dari JSON assertion ini adalah sebagai berikut :

- Pertama, ini akan mengurai JSON dan gagal jika data bukan sebuah JSON.
- Kedua, ia akan mencari jalur yang ditentukan, menggunakan sintaks dari Jayway JsonPath 1.2.0. Jika jalan tidak ditemukan, itu akan gagal.
- Ketiga, jika jalur JSON ditemukan dalam dokumen, dan validasi terhadap nilai yang diharapkan diminta, itu akan melakukan validasi. Untuk nilai nol ada kotak centang khusus di GUI.

Perhatikan bahwa jika path akan mengembalikan objek array, itu akan diulang dan jika nilai yang diharapkan ditemukan, pernyataan akan berhasil. Untuk memvalidasi array kosong gunakan string `[]`. Juga, jika patch akan mengembalikan objek kamus, itu akan dikonversi ke string sebelum Perbandingan.



Gambar 2.6.2 : Panel kontrol JSON assersation

2.7 Processor

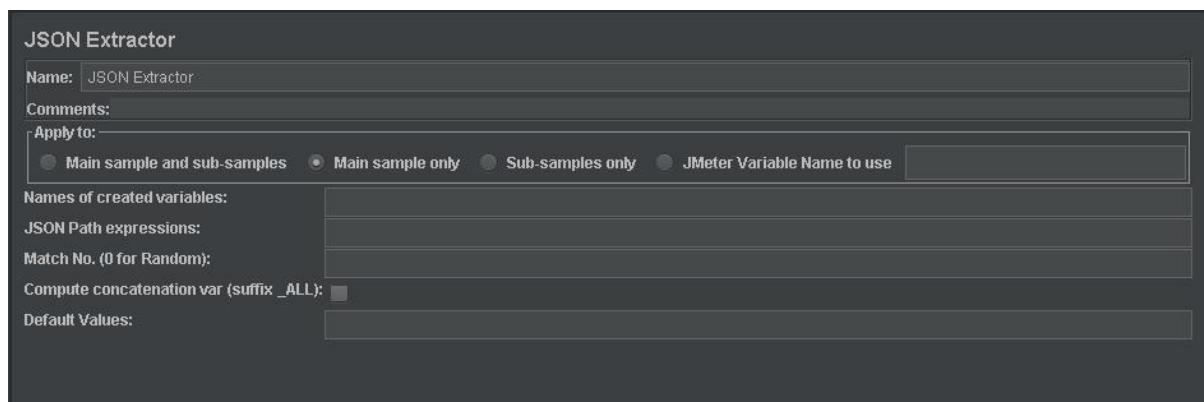
Sebuah processor digunakan untuk memodifikasi samplers didalam ruang lingkupnya. Sebagai contoh ketikan kita ingin mengolah data baik berupa response data maupun data lainnya yang ada dari hasil proses sebauh samplers maka kita bisa menggunakan element ini. Di JMeter setidaknya terbagi kedalam 2 tipe processor yaitu pre-processor dan post processor.

2.7.1 Post Processor

Post-Prosesor diterapkan setelah samplers. Perhatikan bahwa mereka diterapkan untuk semua sampler dalam cakupan yang sama, sehingga untuk memastikan bahwa post-prosesor hanya diterapkan pada sampler tertentu, tambahkan sebagai anak sampler.

2.7.1.1 JSON Extractor

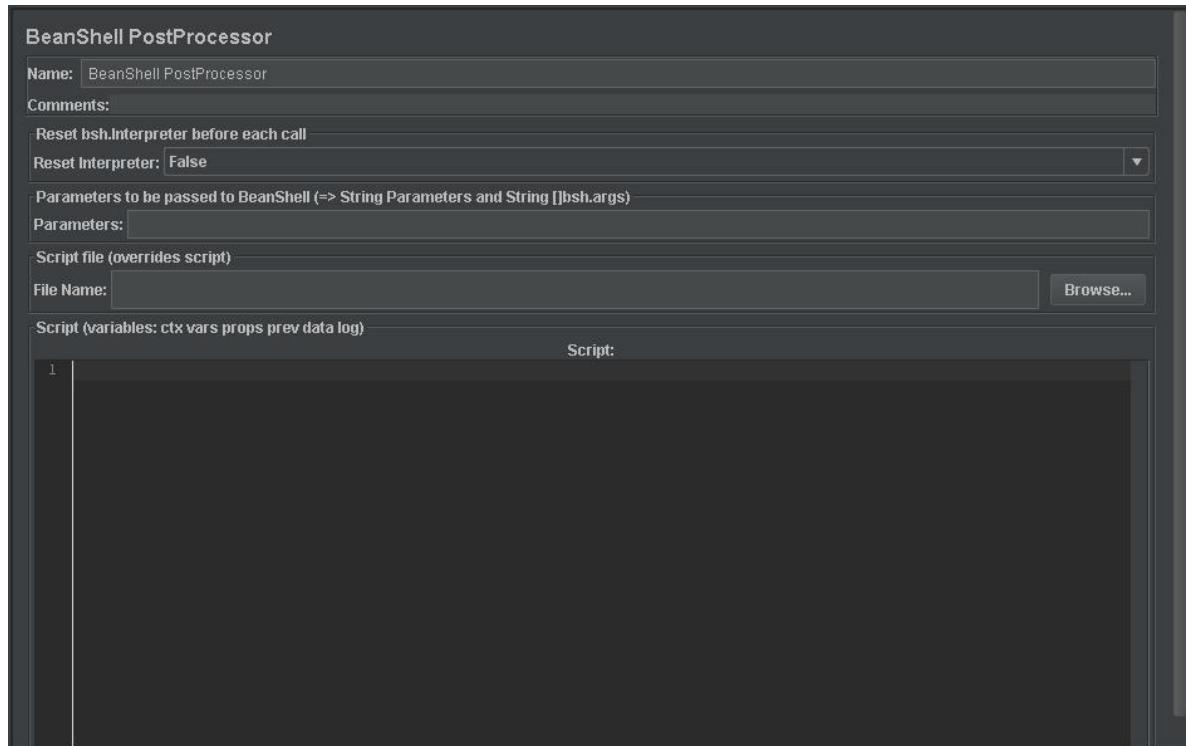
JSON Extractor memungkinkan kamu mengekstrak data dari respon JSON menggunakan JSON-PATH sintak. Post processor ini sangat mirip dengan extractor ekspresi reguler. Itu harus ditempatkan sebagai anak dari Sampler HTTP atau sampler lain yang memiliki respons. Ini akan memungkinkan Anda untuk mengekstrak konten teks dengan cara yang sangat mudah.



Gambar 2.7.1.1 : Panel kontrol JSON Extractor

2.7.1.2 BeanShell PostProcessor

The BeanShell PreProcessor memungkinkan kode arbitrer diterapkan setelah mengambil sampel. BeanShell Post-Processor tidak lagi mengabaikan sampel dengan data hasil nol panjang



Gambar 2.7.1.2 : Panel kontrol BeanShell PostProcessor

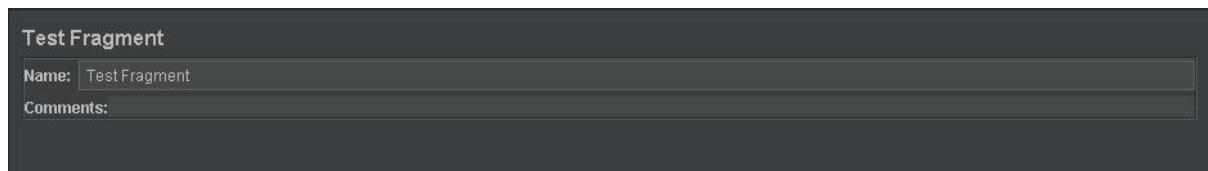
Salah satu yang dapat dilakukan yaitu kita dapat menulis semua data yang ada dari responde JSON diolah dan ditulis kedalam bentuk file dengan ekstention.csv.

2.7.2. Pre Processor

Dalam pre-processor kita tidak menggunakannya, namun secara fungsi element tidak jauh berbeda dengan post processor. hal ini dapat di pelajari lebih lanjut di

2.8 Test Fragment

Test Fragment biasanya digunakan bersama dengan Include Controller dan elemente Controller.



Gambar 2.8 : Panel kontrol Test Fragment

Secara Panel kontrol, test fragment tidak memiliki banyak fitur, hanya "name" saja yang bisa kita ganti untuk mendapatkan keterangan dari setiap fragment yang akan kita buat.

3. Definisi Test Plan dan Workbeanch

Sebuah test plan menjelaskan serangkaian langkah yang akan dijalankan JMeter saat dijalankan. Rencana pengujian lengkap akan terdiri dari satu atau lebih Grup Thread, logic controllers, sample generating controllers, listeners, timers, assertions, and configuration elements.

3.1 Menambahkan Elements

Untuk dapat menggunakan JMeter kita harus menambahkan beberapa element agar dapat di jalankan, setelah kita mengetahui berbagai fungsi element yang ada, kita bisa dengan mudah membuat skenario testing yang akan kita jalankan dengan menambahkan berbagai elemen yang akan kita gunakan berdasarkan kebutuhan.

3.2 Menyimpan Test Plan

Salah satu yang paling penting setelah kita menambahkan elements yaitu menyimpan hasil test plan yang sudah dibuat. JMeter menyediakan beberapa pilihan yang bisa digunakan. Untuk menyimpan test plan kita bisa menggunakan "save" jika file belum pernah dilakukan save dan save test plan as jika file sudah pernah digunakan dan kita bermaksud untuk menyimpan dengan nama yang baru.

3.3 Menjalankan Test Plan

Untuk menjalankan paket pengujian, kita bisa memulai dengan melakukan pilih "**Start**" (control+ r) dari item menu "**Run**". Ketika JMeter berjalan, kita dapat melihat kotak hijau kecil di ujung kanan bagian tepat di bawah menu bar. Kita juga dapat memeriksa menu "Run". Jika "Start" dinonaktifkan, dan "Stop" diaktifkan, maka JMeter menjalankan test plan.

Angka-angka di sebelah kiri kotak hijau adalah jumlah thread aktif / jumlah total thread. Ini hanya berlaku untuk tes yang dijalankan secara lokal; mereka tidak menyertakan thread yang dimulai pada sistem jarak jauh ketika menggunakan mode client-server.

- Continue : Jmeter akan mengabaikan kesalahan, melanjutkan eksekusi dan hanya sampler yang terpengaruh yang gagal dalam pendengar.
- Start Next Thread Loop: Jmeter akan mengabaikan kesalahan dan melanjutkan dengan eksekusi loop thread berikutnya.
- Stop Thread: Eksekusi Thread saat ini akan dihentikan, utas yang tersisa akan dieksekusi sebagaimana ditentukan.
- Stop Test: Seluruh Uji coba akan dihentikan jika sampler ada kesalahan.
- Stop Test Now: Eksekusi pengujian akan dihentikan secara mendadak dan semua sampler yang saat ini aktif akan terputus untuk mengakhiri eksekusi.

3.4 Menghentikan Test Plan

Ada dua type perintah yang mungkin digunakan untuk menghentikan sebuah test plan yang sedang berjalan yaitu stop (Control + .) dan shutdown (Control + ,)

4. Menggunakan JMeter dengan Data+Keyword Driven untuk API Performance Test

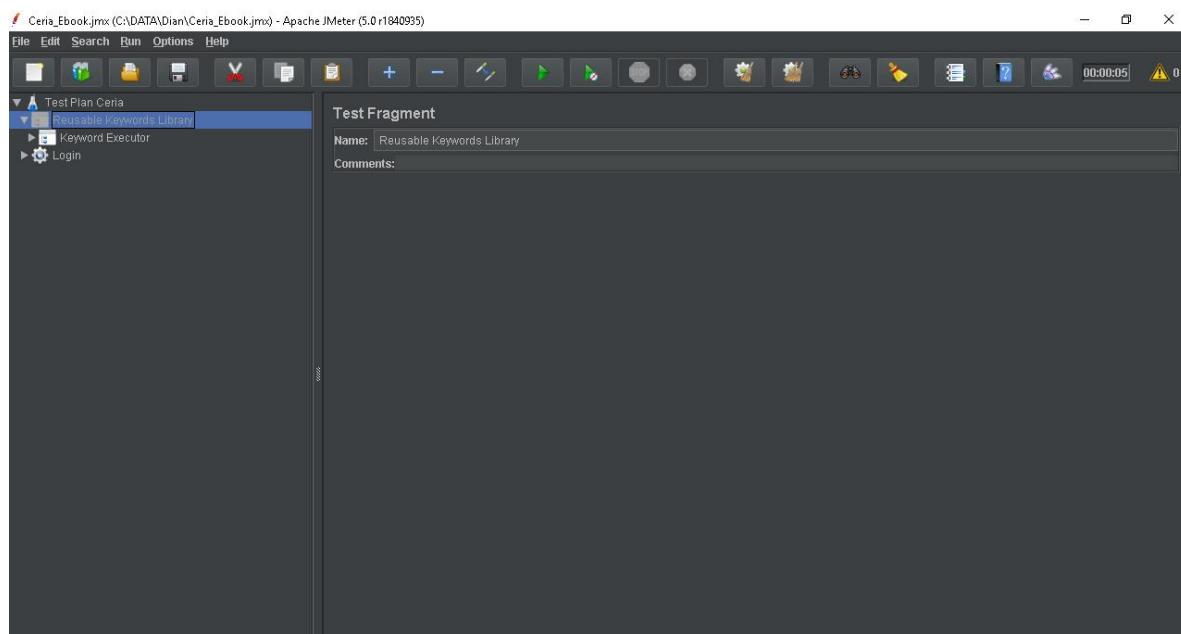
Keyword driven testing merupakan salah satu metodologi yang bisa digunakan untuk mempermudah proses testing dimana setiap fungsi fungsi kecil dibuat sedemikian rupa untuk dibuat lebih mudah dipahami berdasarkan kata kunci (Keyword) dari sisi kebutuhan testing. Salah satu aspek penting dari keyword driven testing yaitu menggunakan table untuk mempermudah pembaca melihat aspek berbagai parameter yang ada, sehingga mudah dipahami.

Pada Skenario test di JMeter yang akan kita buat ini, kita menggunakan 2 File.csv yang didesain dengan 2 kebutuhan yang berbeda, pertama csv file yang kita buat digunakan untuk menyimpan 200 user account yang berbeda dan disimpan dalam bentuk table yaitu untuk data PIN, nomor telephone, Device ID dll. Sementara csv File yang kedua ini kita gunakan untuk memilih keyword driven test yang telah kita definisikan hal ini dimaksudkan untuk mempermudah melihat flow proses apa yang kita buat didalam skenario tersebut dalam bentuk tabel.

4.1 Menambahkan Test Fragment

Secara default ketika kita membuka JMeter kita akan melihat terbentuknya sebuah test plan, dimana hal ini membantu mempermudah kita untuk membuat skenario testing yan akan dilakukan. Untuk menambahkan berbagai element JMeter kita bisa pilih test plan > "Klik kanan" lalu mulai memilih dan memilah element yang kita perlukan seperti yang akan kita lakukan sebagai berikut :

Add > Test Fragment > Test Fragment

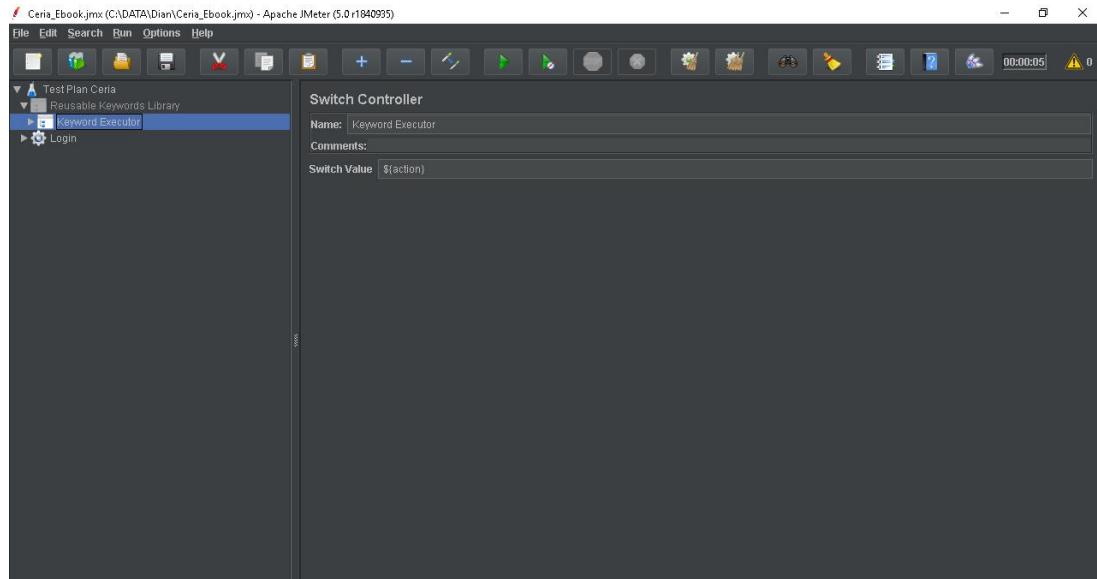


Gambar 4.1 : Menambahkan Test Fragment

Sesuaikan penamaan Test Fragment berdasarkan kebutuhan, dan jika di perlukan kita bisa mengisi kolom komen untuk memperjelas detail informasi yang akan diberikan, ataupun kita bisa mengosongkan komen tersebut.

4.2 Menambahkan Switch Controller

Add > Logic Controller > Switch Controller

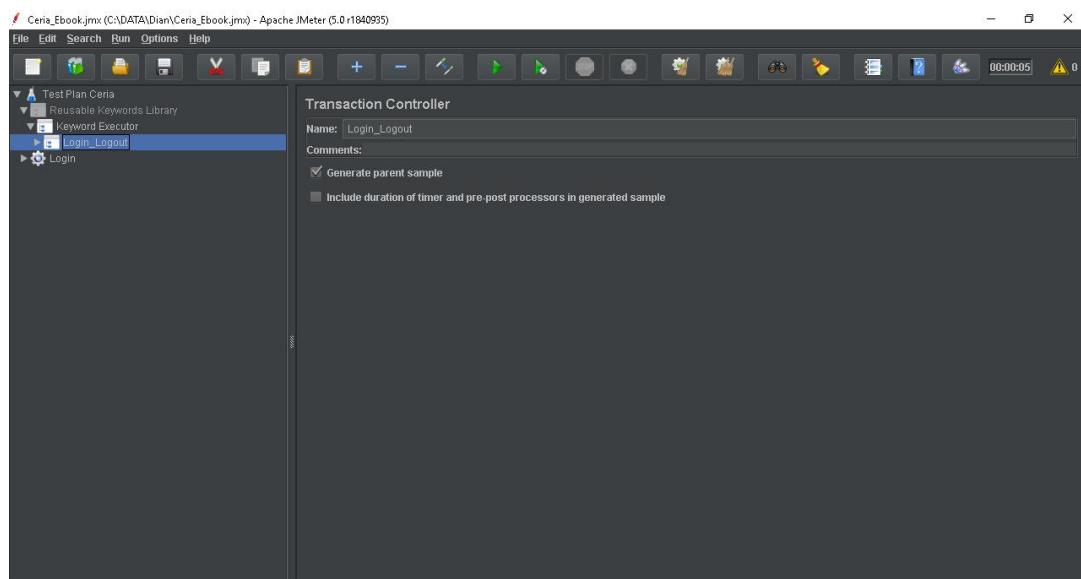


Gambar 4.2 : Menambahkan Switch Controller

Kita bisa menyesuaikan nama untuk switch controller sesuai dengan keinginan, disini kita menggantinya dengan nama Keyword Executor. Adapun untuk switch value kita bisa menyesuaikan.

4.3 Menambahkan Transaction Controller

Add > Logic Controller > Transaction Controller

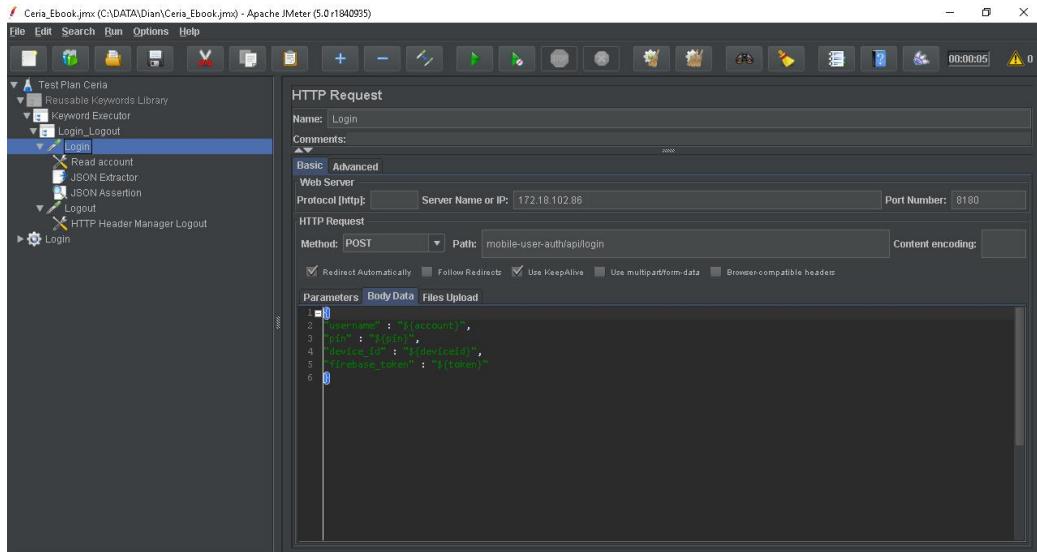


Gambar 4.3 : Menambahkan Transaction Controller

Transaction Controller dapat dieksekusi jika kita melakukan ceklis pada bagian Generate parent sampel yang ada, hal ini secara otomatis akan mengeksekusinya.

4.4 Menambahkan HTTP Request (Login)

Add > Samplers > Http Request

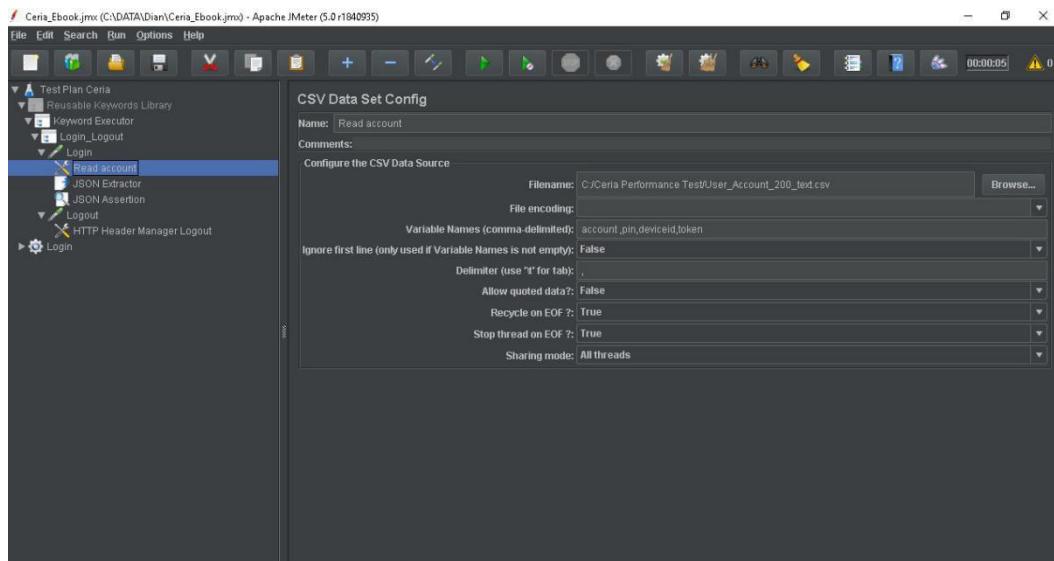


Gambar 4.4 : Menambahkan HTTP Request Login activity

Setting Method, IP address, port number, path and body data which will be used to perform the login process. Assume that account such as username, pin and device have been successfully registered and registered.

4.5 Menambahkan CSV Data Set Config

Add > Config element > CSV Data Set

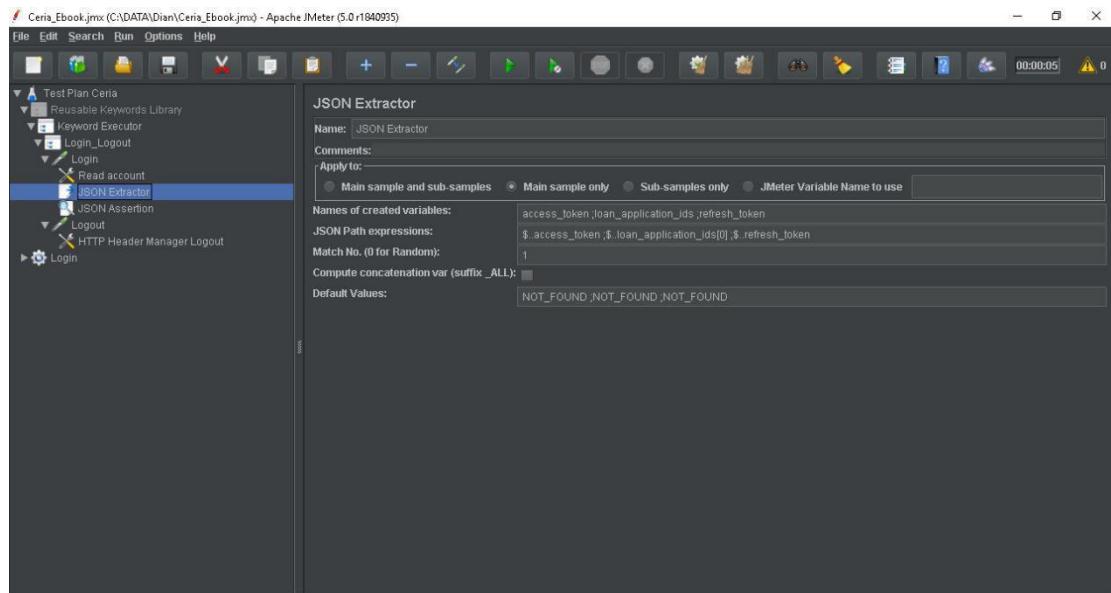


Gambar 4.5 : Menambahkan CSV Data Set Config

In the CSV data set config, we set several configurations such as seen in Figure 4.4.2. It is necessary to know that JMeter reads data in a "ROW/Baris" manner by adding a comma "," with a variable name that will be used in the body data part of the http request.

4.6 Menambahkan JSON Extractor

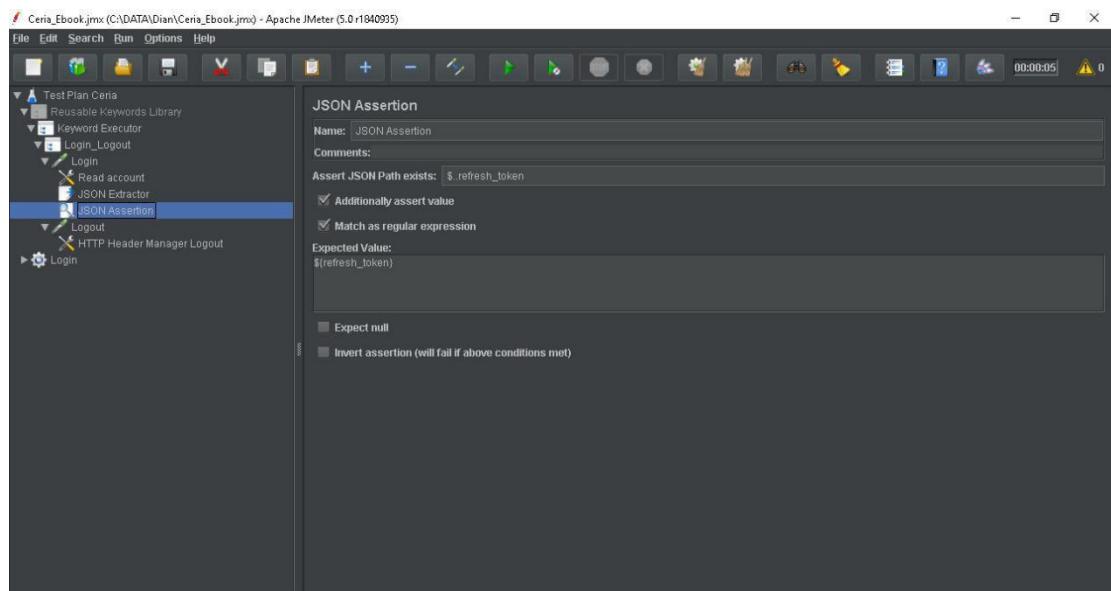
Add > Post Prosesor > JSON Extractor



Gambar 4.6 : Menambahkan JSON Extractor

Selanjutnya, kita akan mengekstrak JSON Response login untuk akses token, loan aplikasi, refresh token dan membuatnya kedalam sebuah variabel. Agar lebih memahami alangkah baiknya di coba secara sederhana terlebih dahulu dengan mengambil hanya 1 data terlebih dahulu.

4.7 Menambahkan JSON Assersation

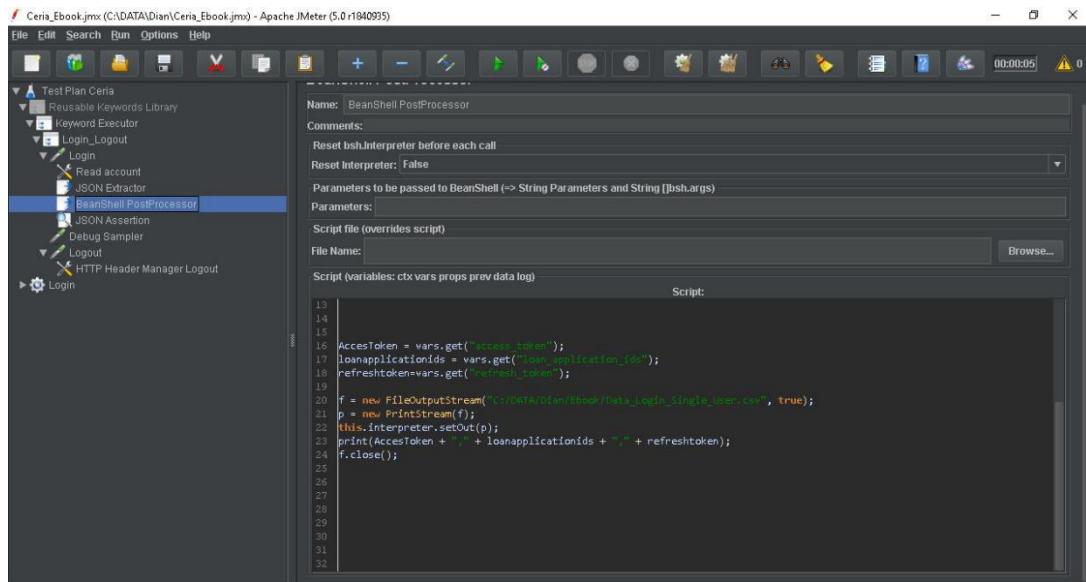


Gambar 4.7 : Menambahkan JSON Assertion

JSON Assertion digunakan untuk memverifikasi refresh token kemudian membandingkannya dengan refresh token yang didapat dari hasil login

4.8 Menambahkan Beanshell Post Processor

Add > Post Processor > Beanshell Post Processor

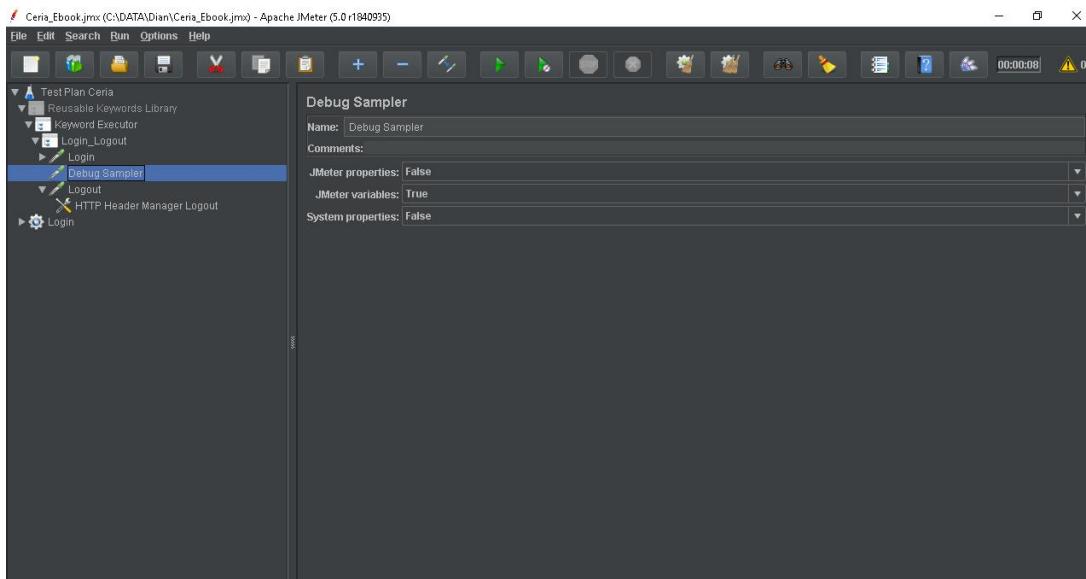


Gambar 4.8 : Menambahkan Beanshell Post Processor

Untuk menulis data yang didapatkan dari hasil Ekstrak JSON pada file dengan extension.csv kita gunakan skrip seperti terlihat pada Gambar 4.8.

4.9 Menambahkan Debug Sampler

Klik kanan pada Transaction Controller (Login_Logout) > Add > Sampler > Debug Samplers

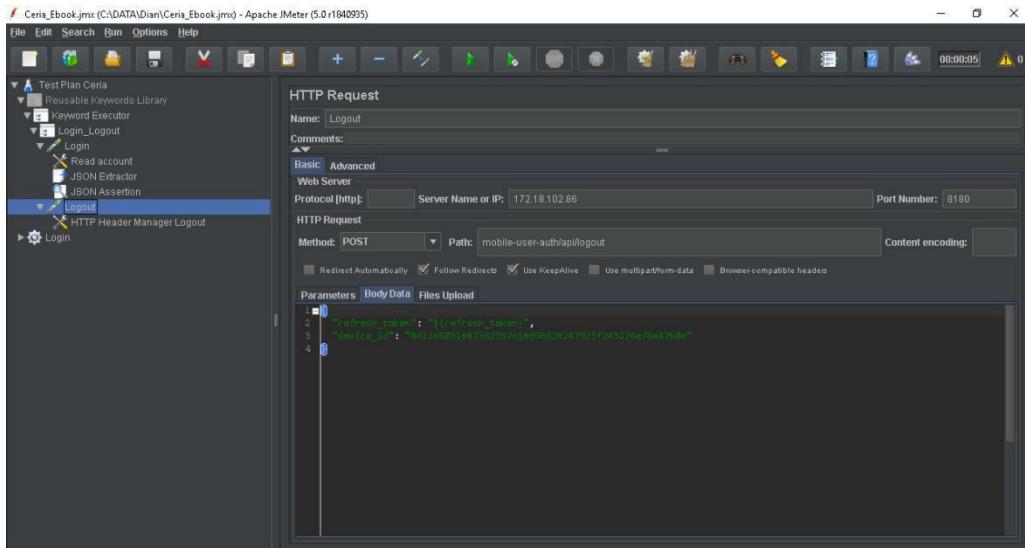


Gambar 4.9 : Menambahkan Debug Sampler

Sesuaikan dengan posisi dimana kita akan melakukan debug. Pada kasus ini kita akan melihat data debug pada proses login

4.10 Menambahkan HTTP Request (Logout)

Add > Samplers > Http Request

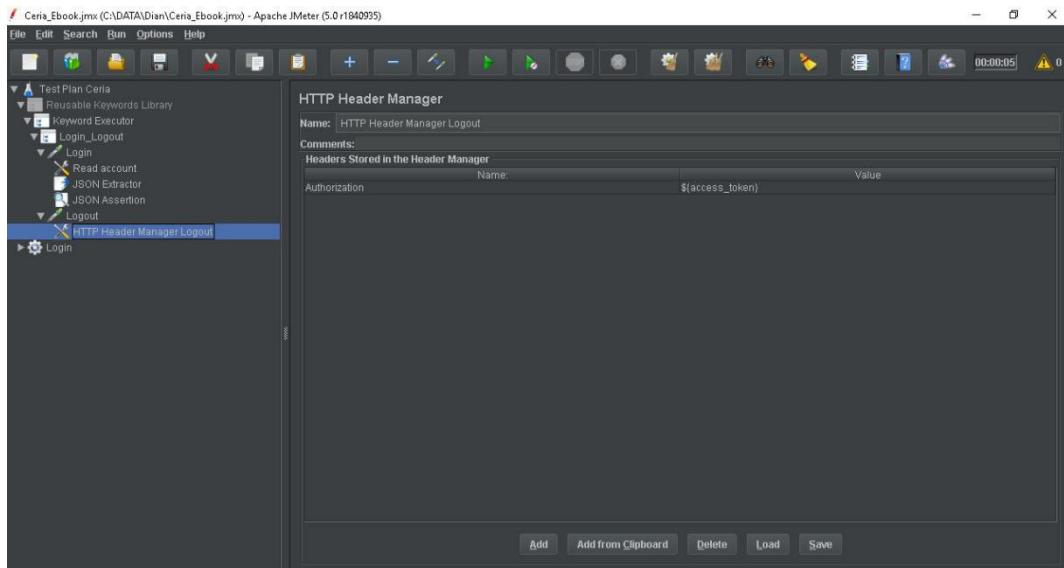


Gambar 4.10 : Menambahkan HTTP Request Logout

Untuk Logout, kita membutuhkan akses token, refresh token dan device id. Selain itu tentu kita harus melakukan setting configurasi yang sama seperti halnya login yaitu method, IP Address, port Number dan path dan data bosdy seperti terlihat pada Gambar 4.10.

4.11 Menambahkan HTTP Header Manager (Logout)

Add > Config Element> Http Header Manager

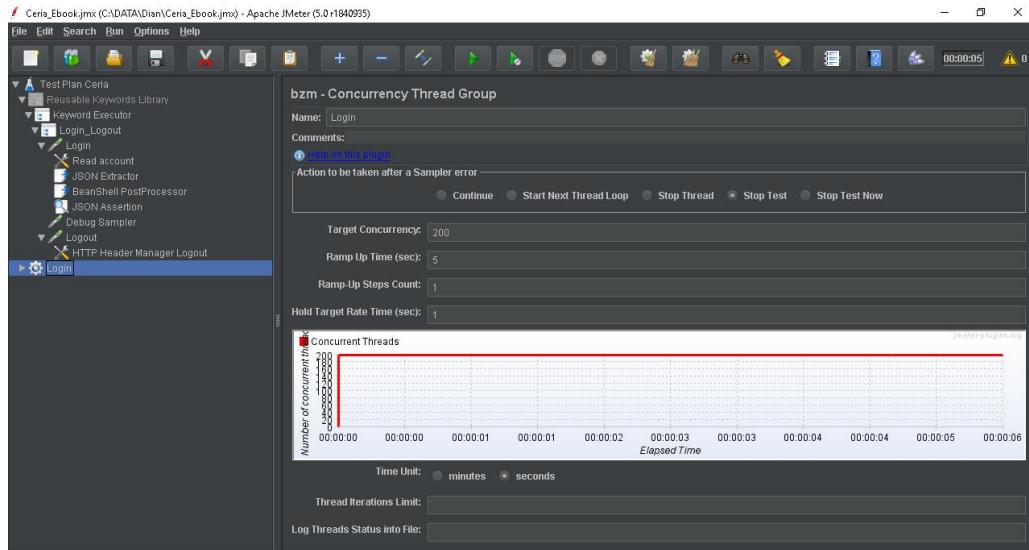


Gambar 4.11 : Menambahkan HTTP Header Manager Logout

Selain itu, kita juga membutuhkan authorization berupa akses token yang digunakan yang didapat dari hasil login yang di set pada HTTP Header Manager.

4.12 Menambahkan Concurrency Thread Group

Add > Thread > Concurrency Thread Group

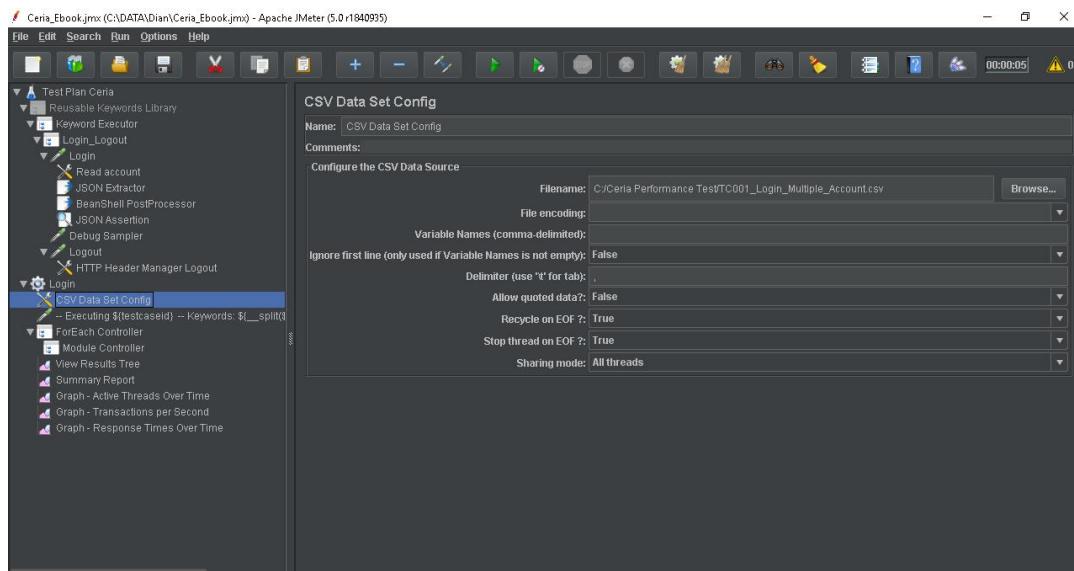


Gambar 4.12 : Menambahkan Concurrency Thread Group

Dari berbagai percobaan yang dilakukan oleh penulis angka ideal ini muncul berdasarkan hasil percobaan yang dilakukan. Dari parameter yang dipilih secara hitungan matematis kita harus mendapatkan sample sebanyak 200 concurrency thread group yang aktif.

4.13 Menambahkan CSV data Set (Keyword Driven)

Add > Config element > CSV Data Set



Gambar 4.13.1 : Menambahkan CSV Data Set Config (Keyword Driven)

Pada CSV Data Set Config kita melakukan pemilihan File yang berisi skenario seperti pada gambar dibawah ini

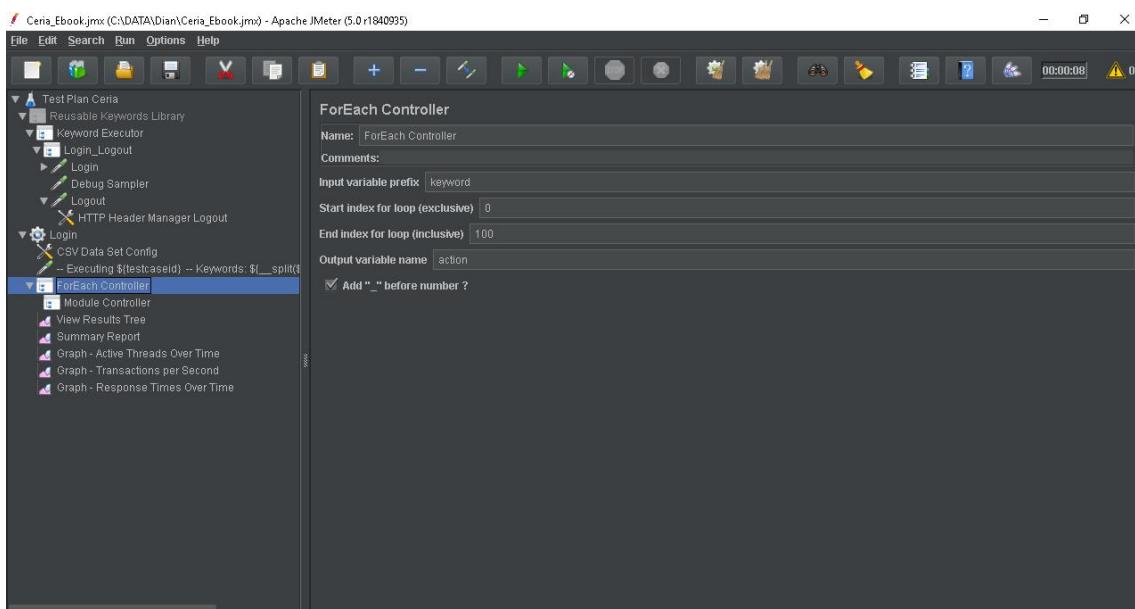
	A	B	C	D
1	testcaseid	description	keywords	cc
2	TC001	Check for Login multiple user	Login_Logout	TC001
3				

Gambar 4.13.2 : File TC001_Login_Multiple_Account.csv

Pada proses Foreach controller akan mencari sebuah keyword Login_Logout (transaction controller) dan menjalankan semua proses yang berada dibawahnya. Keyword driven testing seperti inilah yang secara visual kita bisa melihat lebih detail terkait inrmasi skenario testing yang dibuat sehingga mempermudah pembaca melihat informasi yang ada dengan cara membuat sederhana berbagai parameter yang tidak terlihat di bagian belakang sistem yang sedang bekerja.

4.14 Menambahkan ForEach Controller

Add > Logic Controller > Foreach Controller

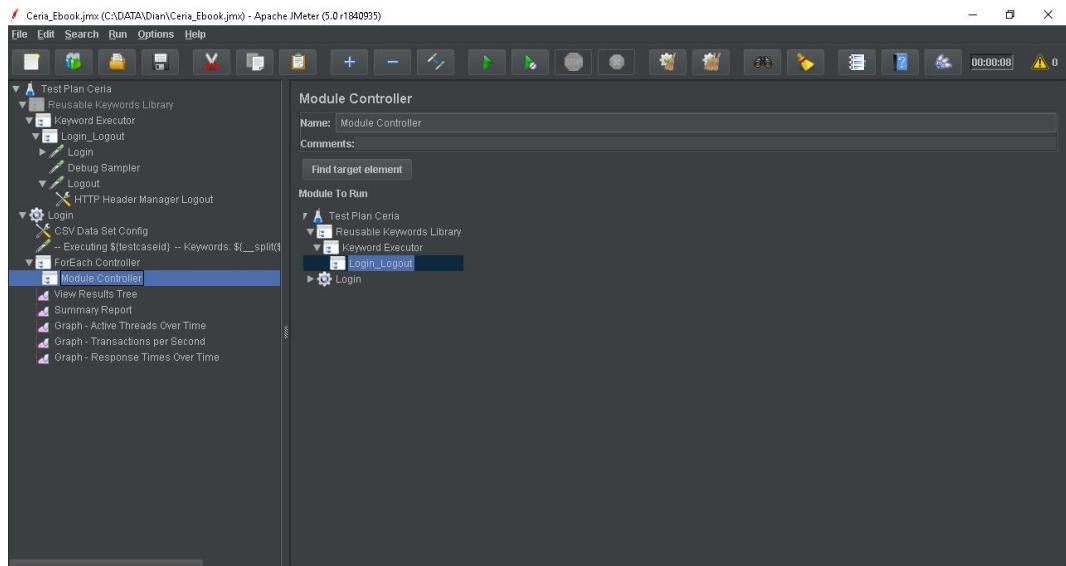


Gambar 4.14 : Menambahkan ForEach Controller

Input variable prefix yang digunakan adalah keyword yang dapatkan dari hasil debug sampler dan kita akan mulai dari index ke 0. sampai ke 100. dimana kita kaan mendefinisikannya dalam bentuk action.

4.15 Menambahkan elemente Controller

Add > Logic Controller > elemente Controller



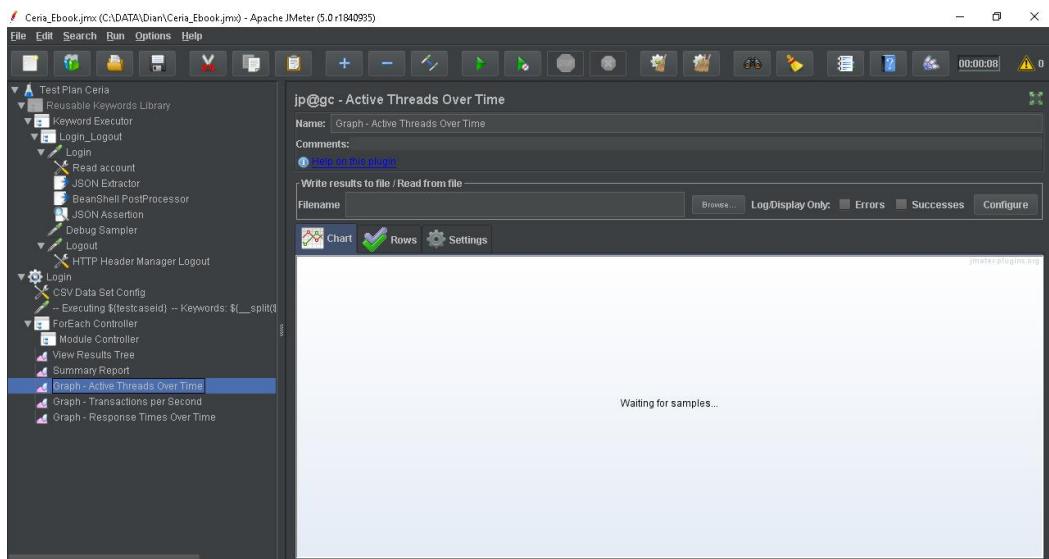
Gambar 4.15 : Menambahkan elemente Controller

Untuk bisa melakukan proses running, kita perlu melakukan pemilihan (Select) elemente controller yang akan kita running. Karena ketika ketika kita salah melakukan pemilihan maka yang akan dirunning oleh JMeter akan berbeda.

5. Menambahkan Report berbentuk Graph dan Table

5.1 Menambahkan Graph

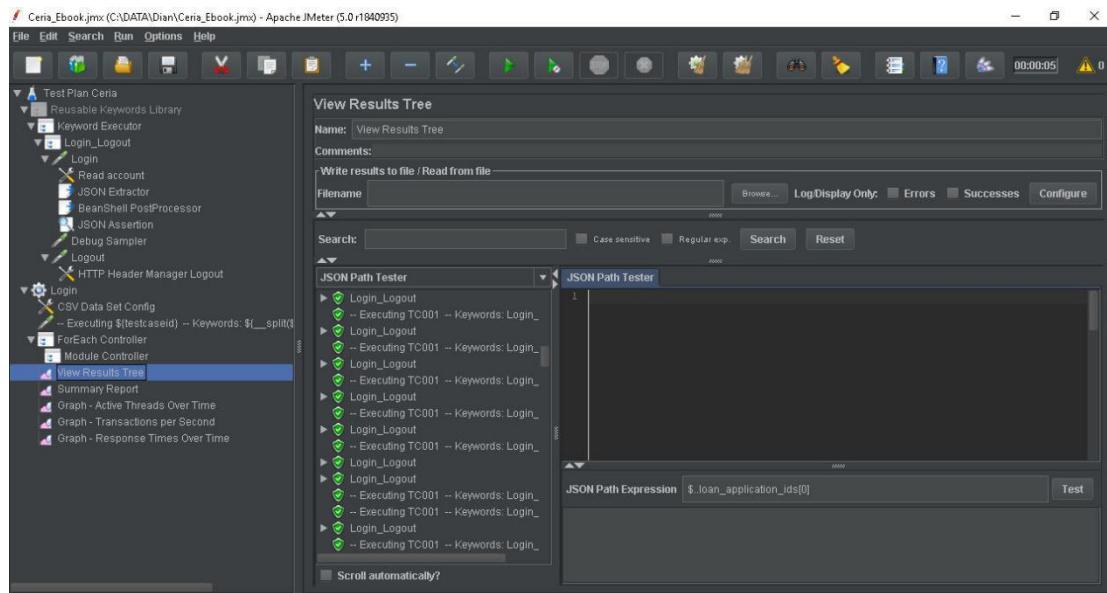
- Add > Listener > Graph - Active Thread Over Time; Graph - Response Time Over Time; Graph - Transaction Persecond



Gambar 5.1 : Menambahkan Graph Active Threads Overtime

5.2 Menambahkan View Results Tree

Add > Listener > View Results Tree

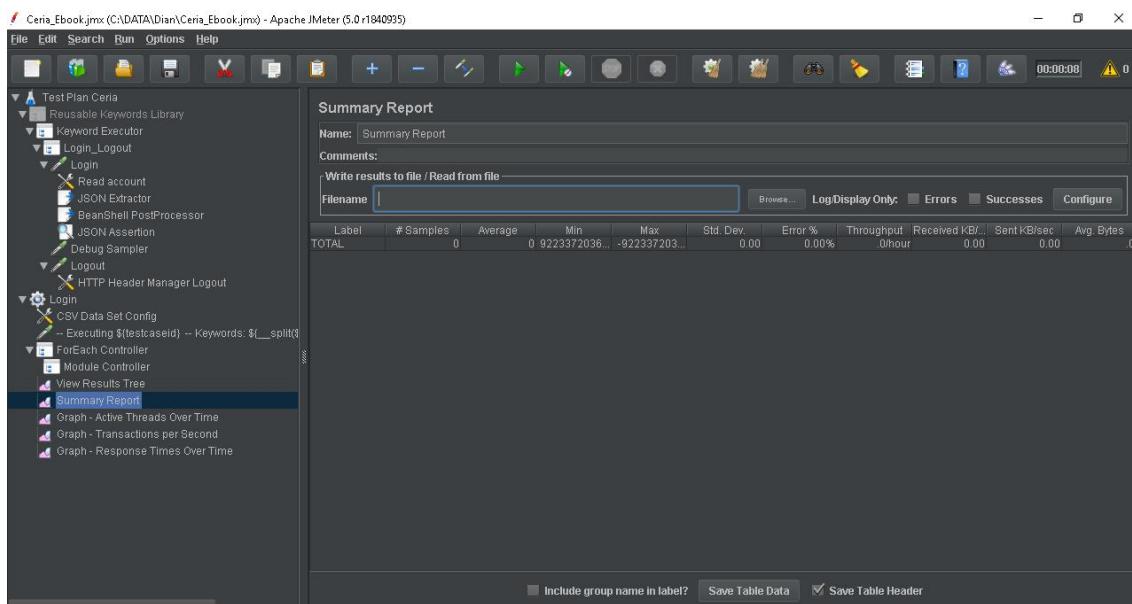


Gambar 5.2 : Menambahkan View Results Tree

Ini merupakan bagian terpenting setelah proses running skenario testing di JMeter, hal ini karena kita bisa melihat berbagai informasi yang ada secara lebih detail disini. Untuk lebih detail melihat informasi kita bisa memilih config dan ceklis berbagai informasi yang kita butuhkan untuk menganalisis data yang kita butuhkan dan jangan lupa untuk menyimpan file .csv karena file ini akan kita gunakan untuk generate dasboard

5.3 Menambahkan Summary Report

Add > Listener > Summary Report



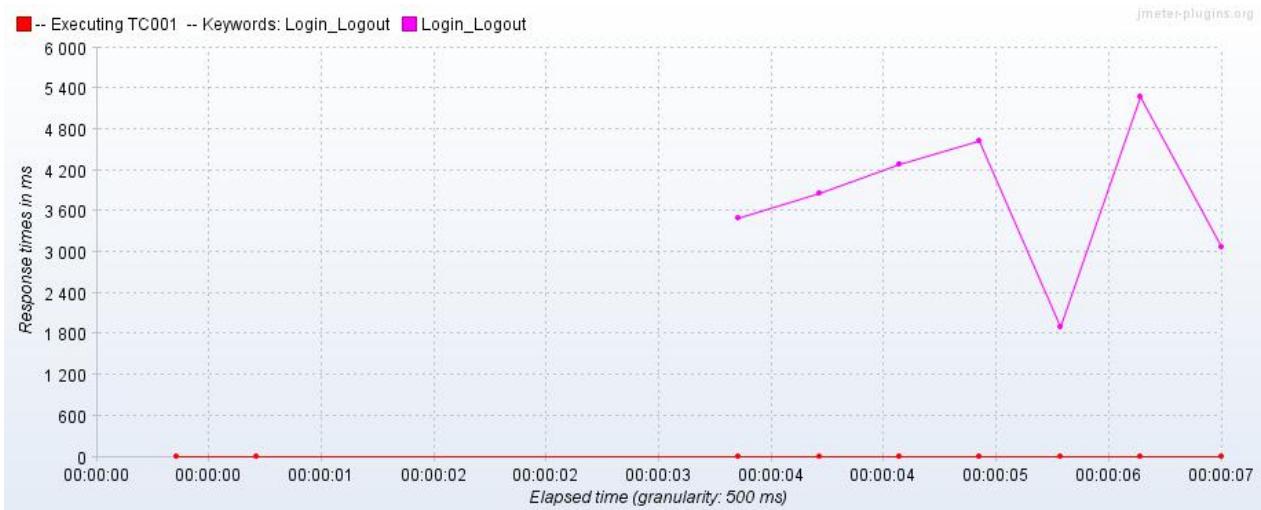
Gambar 5.3 : Menambahkan Summary Report

Disini kita bisa menyederhanakan berbagai informasi, sehingga mempermudah kita untuk memahaminya. Biasnya kita menyimpan summary results dalam file .csv

6. Analisis Data dan Pembahasan

6.1 Grafik Hasil

Adapun Hasil running test Login_Logout dapat dilihat dalam bentuk grafik sebagai berikut :



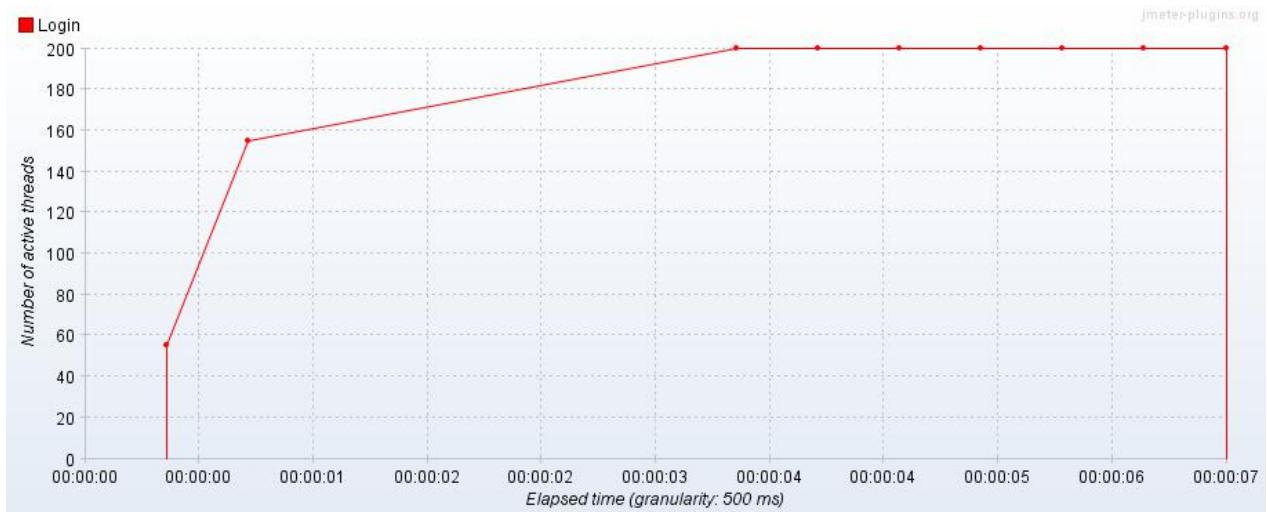
Gambar 6.1.1 : Graph Response Times Over Time (Grafik Respons Waktu)

Warna merah menunjukkan responde time dengan nilai “0” artinya JMeter memang tidak melakukan proses request ke server application, berbeda dengan warna ungu disana kita bisa melihat responde time over time dibawah 5.4 detik. Tentunya data ini akan berbeda jika kita setting Hold Target Rate Time yang lebih lama seperti yang telah dilakukan sebagai berikut :



Gambar 6.1.2 : Graph Response Times Over Time Hold Target Rate Time 100 detik (Grafik Respons Waktu)

Ini berdampak pada responde time yang lebih besar yaitu diatas 5.4 detik. Ini merupakan suatu uji test performance test yang lebih real karena melihat kemampuan server dengan mempertimbangkan banyaknya transaksi yang sedang berjalan.



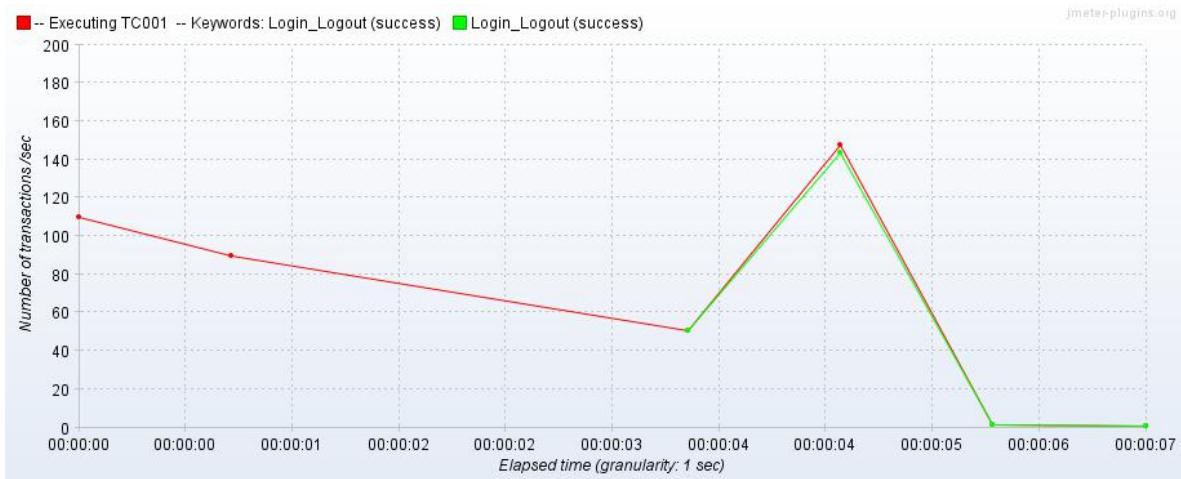
Gambar 6.1.3 : Graph Active Threads Over Time (Grafik aktif thread)

Aktive thread stabil ketika sudah diatas 3 detik, ini sangat normal karena kita melakukan setting dengan ramp-up mencapai 5 detik. Berbeda dengan kasus responde time ketika kita menambahkan hold target rate time selama 100 detik active thread terus stabil sampai pada detik terakhir, artinya tidak ada perubahan data yang membuat aktif thread ini terganggu.



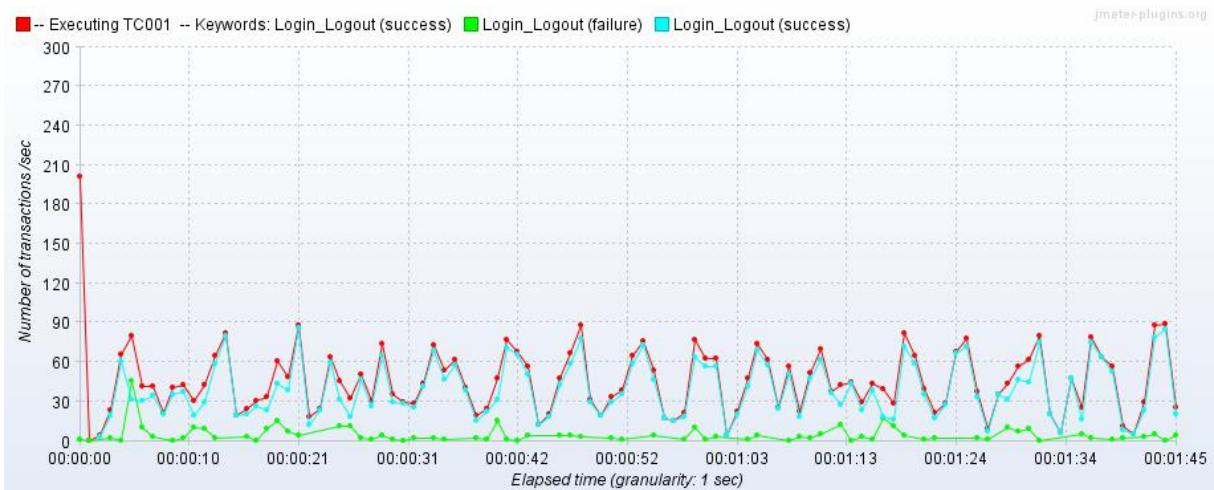
Gambar 6.1.4 : Graph Active Threads Over Time Hold Target Rate Time 100 detik (Grafik aktif thread)

Active thread ini yang menjadi dasar yang sangat penting apakah thread yang kita gunakan dalam keadaan aktif atau tidak, hal ini bisa kita lihat dari grapih ini, berdasarkan concurrency thread group kita menggunakan 200 user account yang berbeda atinya secara teknik harusnya ke 200 user account ini aktif secara bersama. Dan ini memnunjukkan bahwa ada proses yang bersamaan aktif secara lansung



Gambar 6.1.5 : Graph Transactions per Second (Grafik Transaksi per Detik)

Kita bisa melihat transaction per second dengan lebih detail dengan melihat data pada file.csv dan melihat maksimal transaksi yang didapatkan pada saat service berjalan. Untuk lebih detail memahami transaction per second itu ada baiknya kita melihat data rata-rata transaction, artinya kita harus memberikan Hold Target Rate Time yang lebih lama, agar dapat melihat behaviour transaction per second pada aplikasi yang sedang kita test, seperti pada gambar berikut :



Gambar 6.1.6 : Graph Transactions per Second Hold Target Rate Time 100 detik (Grafik Transaksi per Detik)

Cara ini akan lebih efektif untuk melihat behaviour transaction per second karena selain melakukan stress test data yang lebih banyak, kita bisa melihat dengan sangat jelas bahwa dengan 200 active thread yang sedang berjalan dan mempertahankannya selama 00:01:45 tidak semua transaksi yang sedang berjalan mampu di handle dengan baik Login_Logout (succes).

6.2 View Result Tree

Element penting yang harus selalu di perhatikan adalah view results three karena data ini merupakan hasil keseluruhan bagaimana proses running JMeter berjalan. Sukses atau tidaknya ini bisa dibuktikan dengan melihat detail informasi yang ada. Untuk lebih mudah melihatnya kita bisa melihat di file.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M
199	1.56E+12	0 -- Executing TC001 -- Keywords: Login_Logout		200 OK	Login 1-198	text	TRUE			350	0	198	198
200	1.56E+12	0 -- Executing TC001 -- Keywords: Login_Logout		200 OK	Login 1-199	text	TRUE			350	0	199	199
201	1.56E+12	0 -- Executing TC001 -- Keywords: Login_Logout		200 OK	Login 1-200	text	TRUE			350	0	200	200
202	1.56E+12	3493 Login_Logout		200 Number of Login 1-2			TRUE			6284	2664	200	200
203	1.56E+12	2595 Login		200 OK	Login 1-2	text	TRUE			3158	316	200	200
204	1.56E+12	0 Debug Sampler		200 OK	Login 1-2	text	TRUE			2637	0	200	200
205	1.56E+12	898 Logout		200 OK	Login 1-2	text	TRUE			489	2348	200	200
206	1.56E+12	0 -- Executing TC001 -- Keywords: Login_Logout		200 OK	Login 1-2	text	TRUE			2636	0	200	200
207	1.56E+12	3497 Login_Logout		200 Number of Login 1-18			TRUE			6285	2664	200	200
208	1.56E+12	2606 Login		200 OK	Login 1-18	text	TRUE			3158	316	200	200
209	1.56E+12	0 Debug Sampler		200 OK	Login 1-18	text	TRUE			2638	0	200	200
210	1.56E+12	891 Logout		200 OK	Login 1-18	text	TRUE			489	2348	200	200
211	1.56E+12	0 -- Executing TC001 -- Keywords: Login_Logout		200 OK	Login 1-18	text	TRUE			2638	0	200	200
212	1.56E+12	3849 Login_Logout		200 Number of Login 1-20			TRUE			6285	2664	200	200
213	1.56E+12	3552 Login_Logout		200 Number of Login 1-107			TRUE			6286	2664	200	200
214	1.56E+12	2997 Login		200 OK	Login 1-20	text	TRUE			3158	316	200	200
215	1.56E+12	0 Debug Sampler		200 OK	Login 1-20	text	TRUE			2638	0	200	200
216	1.56E+12	2704 Login		200 OK	Login 1-107	text	TRUE			3158	316	200	200
217	1.56E+12	852 Logout		200 OK	Login 1-20	text	TRUE			489	2348	200	200
218	1.56E+12	0 Debug Sampler		200 OK	Login 1-107	text	TRUE			2639	0	200	200
219	1.56E+12	848 Logout		200 OK	Login 1-107	text	TRUE			489	2348	200	200
220	1.56E+12	3708 Login_Logout		200 Number of Login 1-60			TRUE			6285	2664	200	200

Gambar 6.2.1 : View Results tree Keyword

Perlu diketahui bahwa akan ada sebanyak 200 data keyword execution yang berjalan sebanyak 200 user account, tetapi secara result tree kita tidak mempertimbangkan proses ini sebagai bagian dari proses login dan logout karena tidak melakukan sebuah transaction data hanya menghitung tanpa melakukan proses/aktivitas. Tetapi setelah semua account terpenuhi baru proses login berjalan dan melakukan proses/aktivitas login-logout ini ditandai dengan adanya URL yang dikirimkan

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
201	1.548-12	0 - Executing TC001 - Keywords_Login_Logout		200 OK		Login-1>000 test	TRUE		310		200	200 null				
202	1.548-12	3493 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6234	2444		200 null				
203	1.548-12	2595 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		2594		
204	1.548-12	0 Debug Sampler		200 OK		Login-1> test	TRUE		2657	0		200 null				
205	1.548-12	3494 Login_Logout		200 OK		Login-1> test	TRUE		490	2444		200 http://172.18.102.86:8180/mobile-user-auth/api/logout		389		
206	1.548-12	0 - Executing TC001 - Keywords_Login_Logout		200 OK		Login-1> test	TRUE		2656	0		200 null				
207	1.548-12	3497 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6235	2444		200 null				
208	1.548-12	2408 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		2405		
209	1.548-12	0 Debug Sampler		200 OK		Login-1> test	TRUE		499	2444		200 null				
210	1.548-12	3495 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			2638	0		200 null				
211	1.548-12	0 - Executing TC001 - Keywords_Login_Logout		200 OK		Login-1> test	TRUE		6235	2444		200 null				
212	1.548-12	3499 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6236	2444		200 null				1
213	1.548-12	3532 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6236	2444		200 null				1
214	1.548-12	2979 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		2997		
215	1.548-12	0 Debug Sampler		200 OK		Login-1> test	TRUE		2638	0		200 null				
216	1.548-12	2704 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		2704		
217	1.548-12	832 Logout		200 OK		Login-1> test	TRUE		499	2444		200 http://172.18.102.86:8180/mobile-user-auth/api/logout		852		
218	1.548-12	3496 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6235	2444		200 null				
219	1.548-12	943 Logout		200 OK		Login-1> test	TRUE		499	2444		200 http://172.18.102.86:8180/mobile-user-auth/api/logout		943		
220	1.548-12	3708 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6235	2444		200 null				
221	1.548-12	3707 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6235	2444		200 null				
222	1.548-12	3903 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6234	2444		200 null				
223	1.548-12	2881 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		2851		
224	1.548-12	0 Debug Sampler		200 OK		Login-1> test	TRUE		2638	0		200 null				
225	1.548-12	2944 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		2944		
226	1.548-12	2936 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		2936		
227	1.548-12	831 Logout		200 OK		Login-1> test	TRUE		499	2444		200 http://172.18.102.86:8180/mobile-user-auth/api/logout		835		
228	1.548-12	0 Debug Sampler		200 OK		Login-1> test	TRUE		2638	0		200 null				
229	1.548-12	2946 Login		200 OK		Login-1> test	TRUE		6235	2444		200 null				
230	1.548-12	841 Logout		200 OK		Login-1> test	TRUE		2637	0		200 null				
231	1.548-12	2847 Logout		200 OK		Login-1> test	TRUE		499	2444		200 http://172.18.102.86:8180/mobile-user-auth/api/logout		861		
232	1.548-12	3709 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6235	2444		200 null				
233	1.548-12	3741 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6235	2444		200 null				
234	1.548-12	3067 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		3067		
235	1.548-12	2795 Login		200 OK		Login-1> test	TRUE		3158	316		200 http://172.18.102.86:8180/mobile-user-auth/api/login		2794		
236	1.548-12	0 Debug Sampler		200 OK		Login-1> test	TRUE		2638	0		200 null				
237	1.548-12	0 Debug Sampler		200 OK		Login-1> test	TRUE		6235	2444		200 null				
238	1.548-12	833 Logout		200 OK		Login-1> test	TRUE		499	2444		200 http://172.18.102.86:8180/mobile-user-auth/api/logout		853		
239	1.548-12	966 Logout		200 OK		Login-1> test	TRUE		499	2444		200 http://172.18.102.86:8180/mobile-user-auth/api/logout		966		
240	1.548-12	3011 Login_Logout		200 OK		Number of samples in transaction : 3, number of failing samples : 0			6235	2444		200 null				

Gambar 6.2.2 : View Results tree Login-Logout

Yang menjadi catatan penting disini kita bisa melihat bahwa setiap satu proses login selalu dibarengi dengan proses logout, artinya secara teknikal setelah login kemudian melakukan proses extractor JSON data dan selanjutnya digunakan untuk Request Logout maka yang terjadi adalah data bisa digunakan secara automation dan melakukan proses yang lainnya.

6.3 Summary Report

Gambar 6.3 : Summary Report

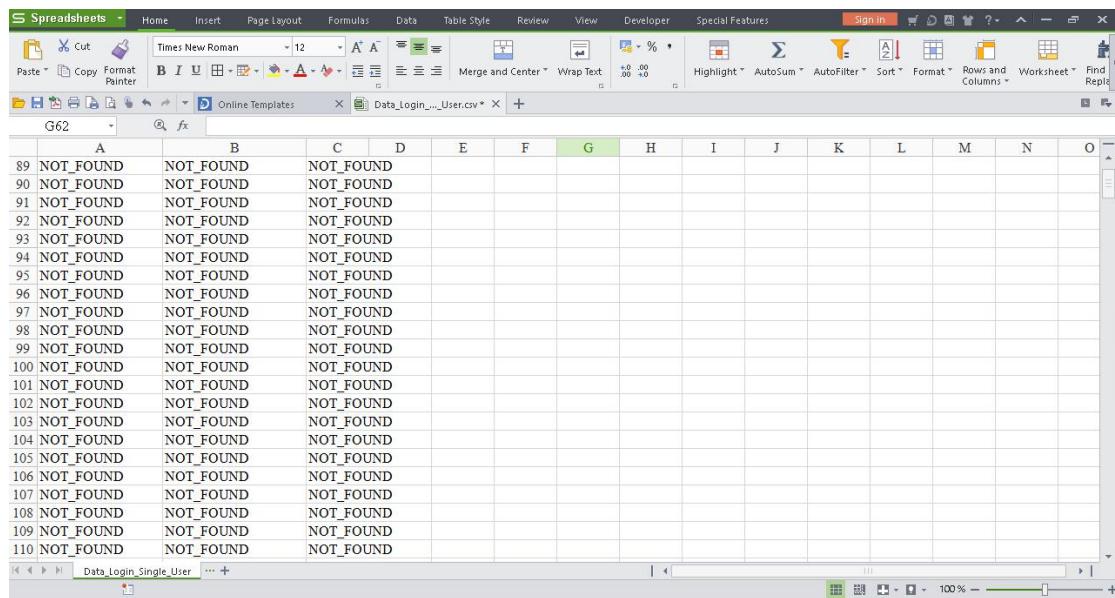
Dari Hasil ujicoba Login-Logout yang telah dilakukan kita bisa melihat bahwa metode keyword data driven bisa digunakan untuk melakukan performance testing.

6.4 Beanshell processor

Untuk melihat apakah kita berhasil menulis semua data JSON resoine ke dalam bentuk csv kita bisa melihat hasil gambar sebagai berikut :

Gambar 6.4.1 : Beansheall processor berhasil menulis data dalam csv File

Jika ada data yang gagal ditulis dalam CSV, artinya tidak ada JSON Respone dari service login, maka beanshell processor akan menulis seperti pada gambar berikut :

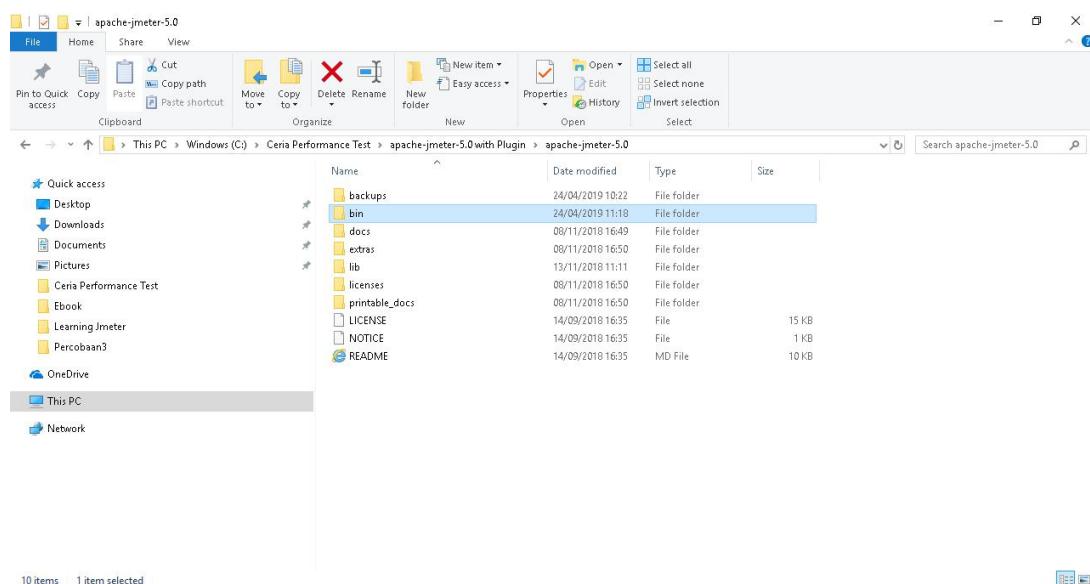


Gambar 6.4.2 : Beansheall processor gagal menulis data dalam csv File

7. Menghasilkan laporan dasboard

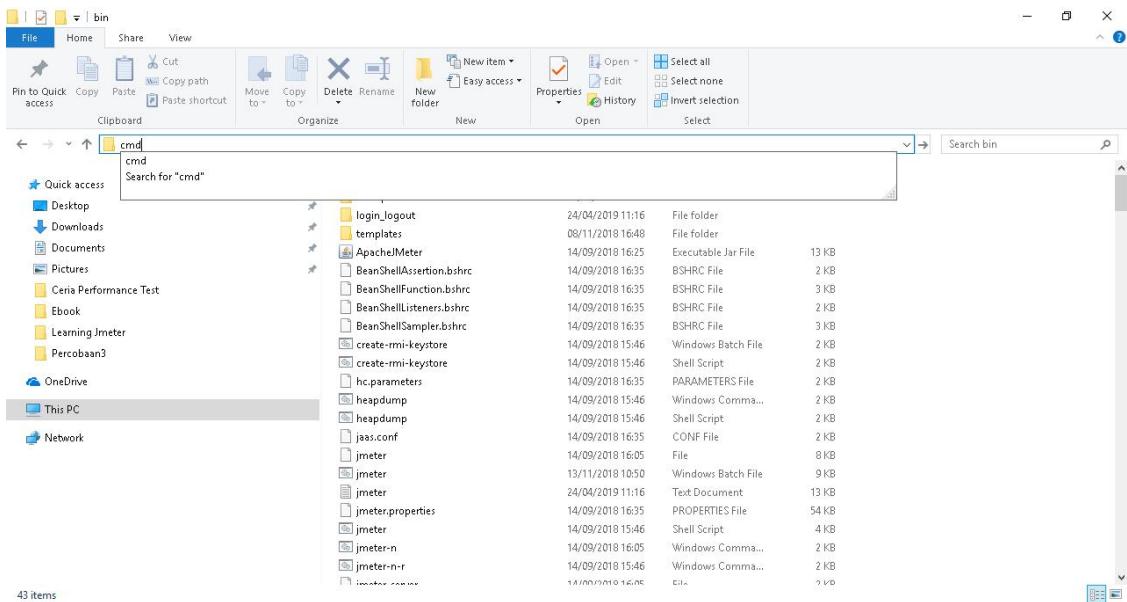
Untuk dapat menghasilkan view result tree menjadi laporan berbentuk dasboard, JMeter menyediakan flur agar dapat dieksekusi melalui CMD Prompt. kita bisa melakukan beberapa langkah sebagai berikut :

- Arahkan folder dimana JMeter disimpan, cari folder bin.



Gambar 7.1 : Folder bin JMeter

- Kemudian kita akan membuka cmd prompt secara langsung pada direktori bin yang ada di JMeter melalui link url folder bin disimpan. Seperti telihat pada gambar dibawah ini



Gambar 7.2 : Open Cmd prom langsung di folder bin JMeter

- Maka akan muncul cmd prompt langsung pada direktori bin yang ada di JMeter. Kemudian kita akan langsung membuat dasboard JMeter seperti telihat pada gambar dibawah ini :

```
PS C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>jmeter -g C:\DATA\Dian\Ebook\Percobaan 2\Percobaan 3\resulttree.csv -o Login_logout
gout
An error occurred: Unknown arg: 2\Percobaan
errorlevel=1
Press any key to continue . . .
c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>jmeter -g C:\DATA\Dian\Ebook\Percobaan 2\Percobaan 3\resulttree.csv -o login_logout
out
An error occurred: Unknown arg: 2\Percobaan
errorlevel=1
Press any key to continue . . .
c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>jmeter -g C:\DATA\Dian\Ebook\Percobaan 3\resulttree.csv -o login_logout
An error occurred: Unknown arg: 3\resulttree.csv
errorlevel=1
Press any key to continue . . .
c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>jmeter -g C:\DATA\Dian\Ebook\Percobaan3\resulttree.csv -o login_logout
An error occurred: Cannot read test results file : C:\DATA\dian\Ebook\Percobaan3\resulttree.csv
errorlevel=1
Press any key to continue . . .
c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>jmeter -g C:\DATA\Dian\Ebook\Percobaan3\resulttree.csv -o login_logout
c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>start index
The system cannot find the file index.

c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>start index.html
The system cannot find the file index.html.

c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>cd login_logout

c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>login_logout>start index.html

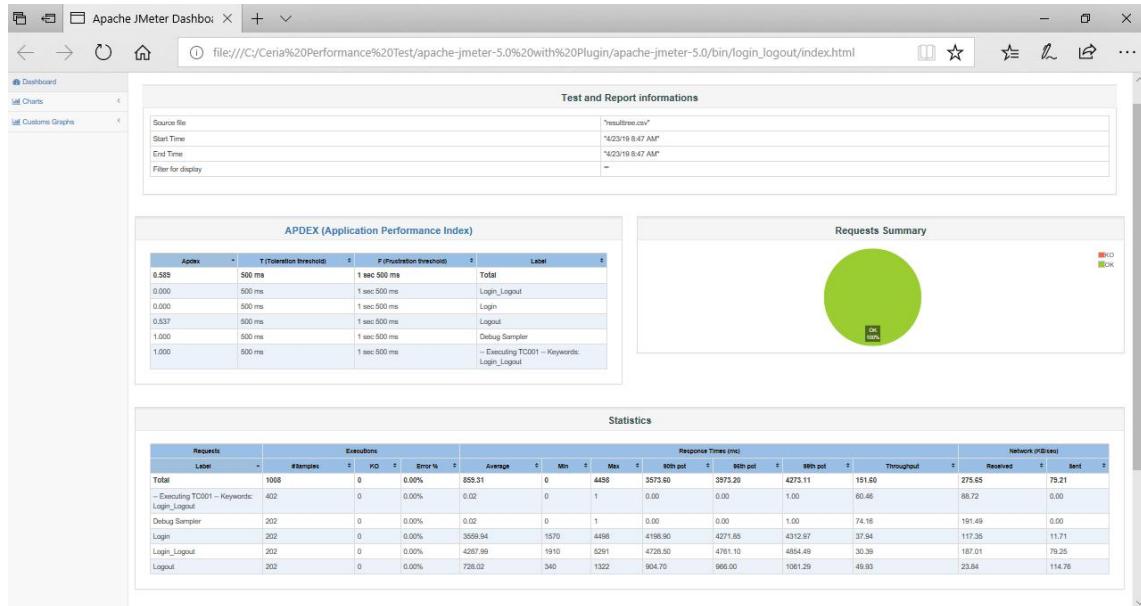
c:\Ceria Performance Test\apache-jmeter-5.0 with Plugin\apache-jmeter-5.0\bin>login_logout
```

Gambar 7.3 : Generate dasboar JMeter via CMD Prompt

Perhatikan beberapa tips berikut karena tidak secara langsung kita bisa menghasilkan laporan dashboard yang kita inginkan ada beberapa hal yang harus di perhatikan :

1. Buat folder tanpa spasi dan jangan terlalu dalam (simple) : karena ketika kita generate dasboard yang ada spasinya, ini akan membuat error level 1 dimana cmd prompt tidak mengenali dimana file .csv yang akan kita generate disimpan.
 2. Gunakan command berikut : JMeter -g [dimanafiledisimpan.csv] -o [namafolderhasilgeneratedashboard] seperti pada perintah berikut : JMeter -g C:\DATA\Diyan\Ebook\Percobaan3\resulttree.csv -o login_logout
 3. Kemudian “cd login_logout” dan “start index.html”

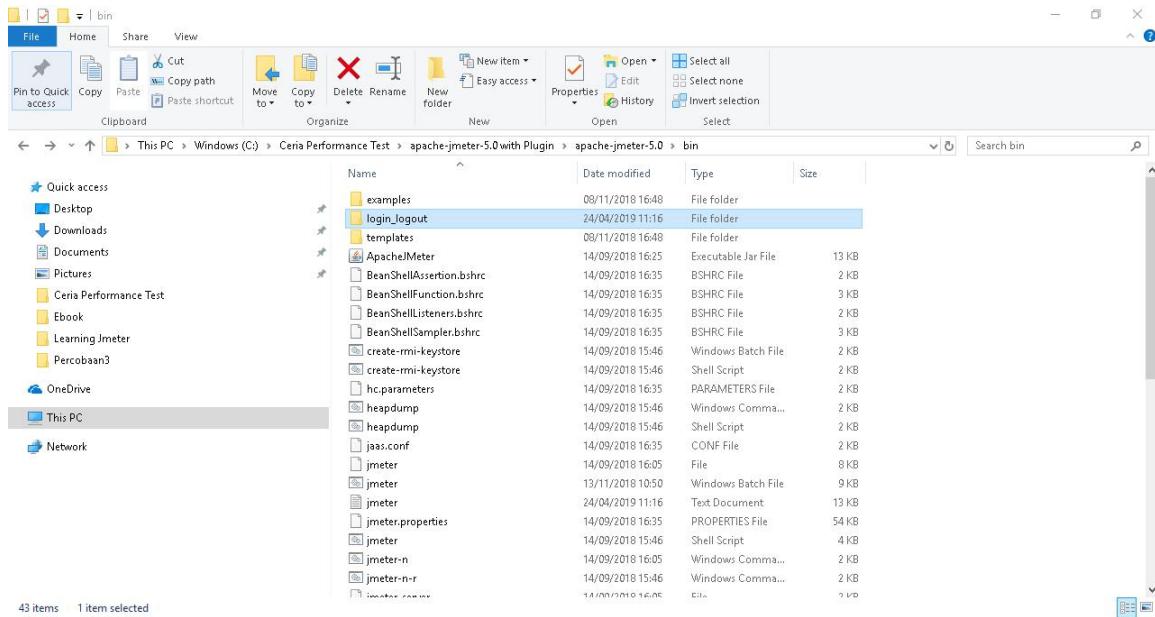
- Untuk melihat hasil laporan yang telah berhasil digenerate, kita bisa membukanya dengan menggunakan google chrome, mozilla, atau browser lainnya. Berikut hasil dasboarnya :



Gambar 7.4 : View Results Generate dasboard JMeter

Kita bisa melihat berbagai informasi melalui dasboard JMeter, detail informasi lebih bagus dan terlihat ilustratif

- Setelah berhasil kita bisa melihat hasil generate dasboar dengan melihatnya dibagian folder bin seperti terlihat pada gambar berikut :



Gambar 7.5 : Generate dasboar JMeter berbentuk folder

Generate dasboard dibentuk dalam sebuah folder utuh, kita bisa membukanya melalui browser dan hasilnya seperti telihat pada Gambar 7.4.

8. Diskusi dan Tanya Jawab

- Saya sebagai pemula di API testing biasnya menggunakan postman atau pun tools lainnya. Apakah saya bisa belajar JMeter ini?

Ya, Penulis pun juga sama belajar dari hal-hal manual dan sederhana, mempelajari flow proses bagaimana software JMeter membaca, menulis, mengolah data sehingga bisa membentuk sebuah skenario testing.

- Apakah JMeter bisa digunakan untuk automation?

Ya, JMeter bisa digunakan untuk automation testing, hanya teknik bagaimana data tersebut bisa digunakan untuk proses selanjutnya perlu disesuaikan dengan skenario testing yang dibuat. Automation disini adalah untuk testing API (Application Programming Interface)

- Kapan waktu terbaik melakukan performance testing ?

Jika kita memiliki jaringan intranet yang traffic nya sangat tinggi, maka akan lebih baik melakukan performance testing pada saat traffic intranet/internet dianggap stabil/rendah. Hal ini karena ketika penulis melakukan ujicoba terhadap aplikasi ini, sering mendapatkan total jumlah sample yang tidak sesuai semestinya, artinya ketika kita menggunakan concurrency thread group sebanyak 200 user account, maka yang berhasil hit tersebut biasanya kurang dari 200 tentunya hal ini kita anggap sebagai data yang tidak ideal sehingga kita perlu terus mengulangi percobaan tersebut sampai mendapatkan data yang ideal.

- Bagaimana kita menjamin performance test yang kita lakukan berhasil ?

Mempertimbangkan banyak aspek Performance test tidak semudah apa yang kita lihat dan sama berdasarkan teori yang ada. Banyak hal-hal yang menjadi pertimbangan penting ketika melakukannya. Dari hasil yang didapat penulis selama ini merupakan hasil ujicoba secara terus menerus dan tentunya dibarengi dengan sabar sampai tahap demi tahap apa yang dilakukan kita pahami. Dan yang terpenting dari proses performance testing menggunakan JMeter adalah untuk mencoba dari hal-sederhana terlebih dahulu agar flow proses yang kita alami di tahap middle ataupun advance bisa memiliki pondasi yang kuat. Sebagai contoh kita mencoba untuk extract JSON cobalah dengan satu variable sederhana, kemudian coba dengan fitur-fitur kecil yang ada mulai pahami tahap demi tahap nya hal ini karena dirasa penulis pun terkadang menyepelekan maksud dan arti fitur-fitur tersebut padahal dilain kesempatan ternyata fitur itu sangat bermanfaat.

- Mana yang lebih baik run via GUI Mode atau Command line mode?

Kedua aspek ini tentunya memiliki keunggulan masing-masing, namun langkah lebih baiknya kita bisa menggunakan GUI untuk membuat skenario testing nya terlebih dahulu, run via GUI Mode. Setelah itu kita bisa menggunakan command line mode untuk running dengan data yang lebih besar ataupun parameter yang lebih banyak. Tentunya, jangan lupa untuk membandingkan data (GUI dan Commandline) apakah sama flow proses yang terjadi. Karena hal itu menjadi pertimbangan yang sangat penting seperti di awal di jelaskan banyak aspek penting yang perlu menjadi pertimbangan ketika kita melakukan performance testing.

9. Kesimpulan

Dari hasil ujicoba kita dapat menyimpulkan bahwa kita telah berhasil membuat skenario automation menggunakan JMeter untuk melakukan performance testing. Secara fungsional skenario yang kita buat telah mampu memproses data yang di dapat dari hasil login, kemudian dilakukan proses selanjutnya yaitu logout. Dari 200 user account yang berbeda yang dilakukan proses login bersama-sama, aplikasi mampu meresponse

dengan baik dimana kita bisa melihat TPS, Throughput, Error rate dan jumlah sample yang kita gunakan sesuai berdasarkan teori perhitungan yang ada. Hasil ini menunjukkan bahwa aplikasi pada kasus login mampu mengeksekusi service yang ada dengan sangat baik.

Refference

Erinle, Bayo. (2017). Performance Testing with JMeter, Birmingham : Packt Publishing Ltd

<https://stackify.com/ultimate-guide-performance-testing-and-software-testing/> diakses pada tanggal (25 April 2019).

<https://www.guru99.com/JMeter-element-reference.html> diakses pada tanggal (25 April 2019).

<https://www.redline13.com/blog/2018/05/guide-jmeter-thread-groups/> diakses pada tanggal (25 April 2019)

<https://medium.com/@DragosCampean/quick-guide-to-jmeters-thread-groups-and-when-to-use-them-6dde95f6c9dd>

Thank You

Tidak ada gading yang tak retak merupakan sebuah pribahasa yang sering kita dengar, setiap orang pasti tidak ada yang sempurna. Begitupun dengan tulisan ini masih banyak kekurangan. Tidak banyak yang diharapkan dari tulisan ini, semua ini berasal dari kata “Lupa”, ya dengan menulis kita akan mengingat setiap case by case yang kita alami dari perjuangan menjadi seorang engineer, akan sangat bangga jika tulisan ini bisa bermanfaat bagi banyak orang. Terima kasih kepada team QA Enginner yang senantiasa sabar membimbing, menerima keluh kesah, diskusi yang tidak pernah berhenti. Kepada @Fakih Basyaruddin @Dwi Yunita @Rahmad M @Ryan @Rudi Butar Butar. Semoga kelak menjadi ladang amal ibadah yang senantiasa terus mengalir sampai akhirat amiin ya robbal'alamin.