

Day 4 - Advanced JMeter Scripting and Real-World Scenarios

Pada sesi materi ini, kita akan mempelajari teknik-teknik scripting yang lebih kompleks dan bagaimana implementasinya dalam situasi nyata yang sering dihadapi dalam pengujian performa.

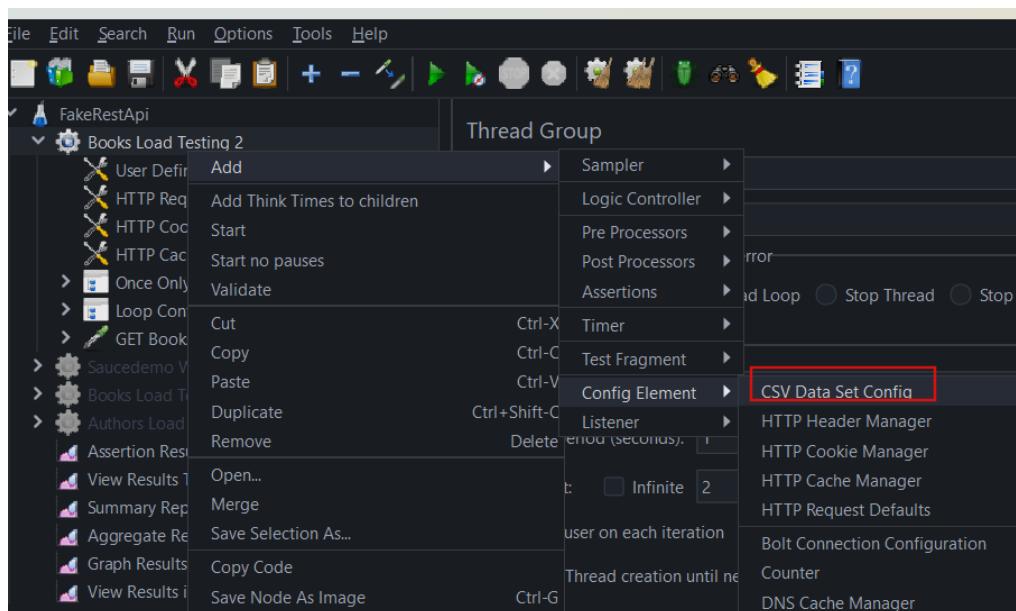
CSV Data Set Config

CSV Data Set Config adalah elemen konfigurasi di JMeter yang memungkinkan kita untuk membaca data dari file CSV (Comma-Separated Values) dan menggunakannya dalam pengujian. Dengan menggunakan elemen ini, kita dapat mengelola data input yang besar dan beragam, seperti username, password, atau parameter lainnya, sehingga pengujian dapat dilakukan secara dinamis dan efisien. Ini sangat berguna untuk melakukan pengujian beban dengan variasi data yang berbeda tanpa perlu mengubah skrip pengujian secara manual.

Berikut adalah langkah-langkah cara menggunakan CSV Data Set Config di JMeter:

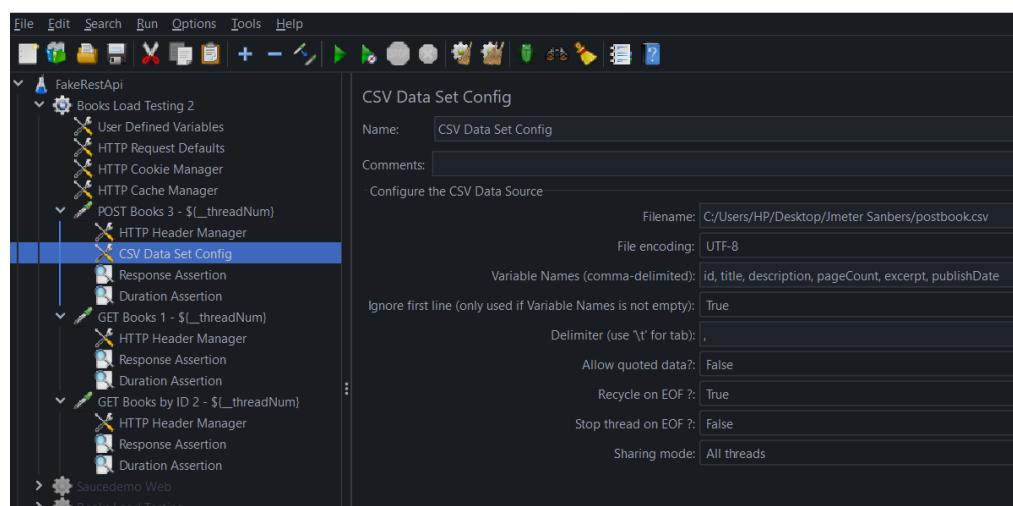
1. Tambahkan Elemen CSV Data Set Config:

- Klik kanan pada thread group atau sampler spesifik di mana kita ingin menggunakan data CSV.
- Dari menu, pilih Add > Config Element > CSV Data Set Config.



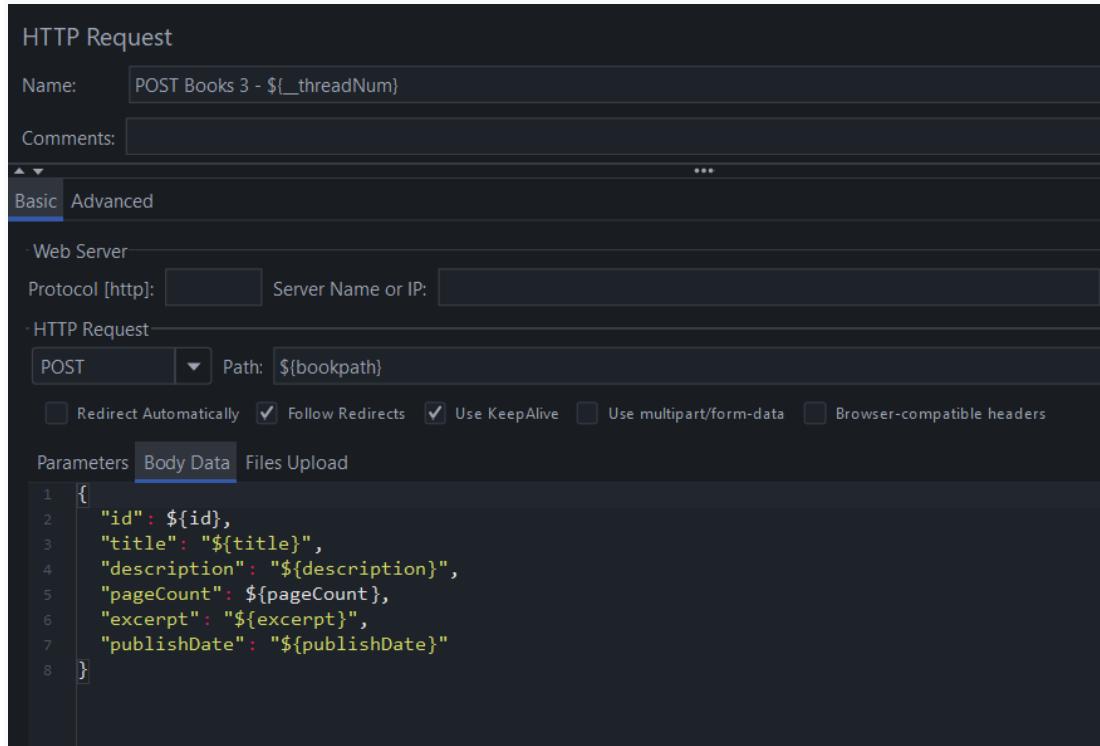
2. Konfigurasi CSV Data Set Config:

- Di elemen CSV Data Set Config, kita akan melihat beberapa kolom yang perlu diisi:
 - **Filename:** Masukkan path ke file CSV kita.
 - **File Encoding:** Pilih encoding untuk file CSV kita.
 - **Variable Names:** Buat nama variabel yang sesuai dengan kolom di file CSV kita, dipisahkan dengan koma.
 - **Delimiter:** Tentukan karakter yang memisahkan nilai dalam file CSV kita, seperti koma (,) titik koma (;) atau tab (/t)
 - **Allow Quoted Data:** Centang ini jika nilai data kita berada dalam tanda kutip.
 - **Recycle on EOF:** Jika dipilih, JMeter akan kembali ke awal file saat mencapai akhir.
 - **Stop Thread on EOF:** Jika dipilih, thread akan berhenti saat mencapai akhir file.



3. Menggunakan Data CSV di Sampler:

- Di sampler kita (seperti permintaan HTTP atau permintaan JDBC), kita dapat menggunakan nama variabel yang telah kita atur di CSV Data Set Config. Cukup gunakan sintaks **`\${variableName}`** untuk merujuk nilai dari file CSV.
- Misalnya, jika kita mendefinisikan variabel bernama **username**, kita dapat menggunakan **`\${username}`** di sampler kita.



Tips: Pastikan untuk menempatkan CSV Data Set Config di bawah thread group atau sampler yang tepat, tergantung di mana kita ingin menggunakan data tersebut.

Logic Controller

Dalam materi ini, kita akan membahas beberapa contoh JMeter Controllers, khususnya:

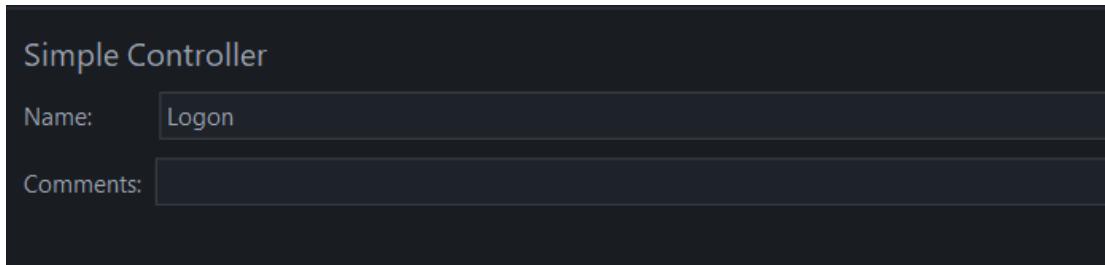
- Simple Controller
- Transaction Controller
- Loop Controller
- While Controller
- ForEach Controller
- If Controller
- Once Only Controller
- Throughput Controller
- Interleave Controller
- Random Controller
- Random Order Controller

Secara teknis, JMeter memiliki dua jenis controller, yaitu **Samplers** dan **Logical Controllers**. Controller yang kita bahas di materi ini adalah **Logical Controllers** yang memungkinkan kita untuk menyesuaikan bagaimana JMeter mengirimkan permintaan sesuai dengan profil beban kita.

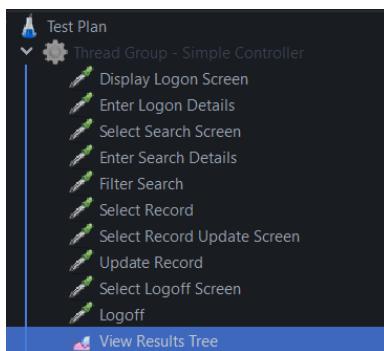
Berikut adalah beberapa contoh penggunaannya menggunakan **Dummy Samplers** dari Jmeter. Teman-teman bisa mengikuti setiap contoh dengan mengunduh JMX [di sini](#).

1. Simple Controller

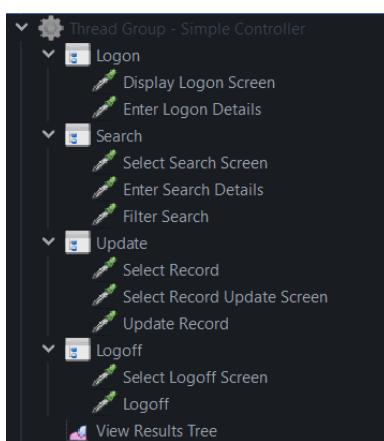
Seperti namanya, ini adalah controller yang paling sederhana dan tidak menawarkan fungsi tambahan untuk cara pengujian kita berjalan, selain memberikan container untuk samplers, pre dan post processors, dan lain-lain.



Ini adalah contoh Test Plan tanpa controller :



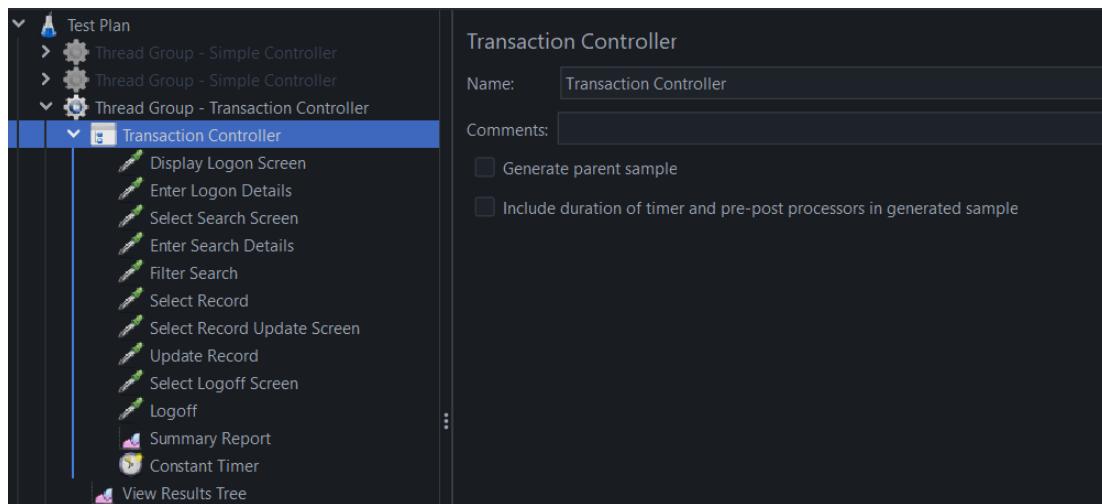
Jika kita menambahkan beberapa Simple Controllers, script menjadi lebih mudah dibaca karena dibagi menjadi area fungsional. Namun, ini tidak mempengaruhi eksekusi pengujian.



2. Transaction Controller

Controller ini juga cukup sederhana dan memungkinkan kita untuk mengelompokkan beberapa sampler dan mengukur total execution time dari setiap 1 transaction controller tersebut. Transaction controller ini juga bisa kita gunakan untuk melakukan full skenario seperti end to end testing.

Sebagai contoh, kita bisa ambil samplers yang sama yang kita gunakan di Simple Controller dan simpan di dalam Transaction Controller. Lalu kita akan menambahkan Summary Report untuk membantu memvisualisasikan data.



Jika kita running, maka kita mendapatkan hasil tambahan yaitu Transaction Controller. Durasi dari Transaction Controller adalah total dari semua child samplers.

Summary Report										
Name:	Summary Report									
Comments:										
Write results to file / Read from file										
Filename	Browse...	Log/Display Only:	<input type="checkbox"/> Errors	<input type="checkbox"/> Successes	<input type="button" value="Configure"/>					
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
Display Logon Screen	1	396	396	396	0,00	0,00%	2,5/sec	0,13	0,00	53,0
Enter Logon Details	1	495	495	495	0,00	0,00%	2,0/sec	0,10	0,00	52,0
Select Search Screen	1	418	418	418	0,00	0,00%	2,4/sec	0,12	0,00	53,0
Enter Search Details	1	390	390	390	0,00	0,00%	2,6/sec	0,13	0,00	53,0
Filter Search	1	262	262	262	0,00	0,00%	3,8/sec	0,17	0,00	46,0
Select Record	1	109	109	109	0,00	0,00%	9,2/sec	0,41	0,00	46,0
Select Record Update ...	1	309	309	309	0,00	0,00%	3,2/sec	0,19	0,00	60,0
Update Record	1	261	261	261	0,00	0,00%	3,8/sec	0,17	0,00	46,0
Select Logoff Screen	1	362	362	362	0,00	0,00%	2,8/sec	0,14	0,00	53,0
Logoff	1	487	487	487	0,00	0,00%	2,1/sec	0,08	0,00	39,0
Transaction Controller	1	3489	3489	3489	0,00	0,00%	7,0/min	0,06	0,00	501,0
TOTAL	11	634	109	3489	908,92	0,00%	1,3/sec	0,11	0,00	91,1

Jika kita mengaktifkan opsi **Generate parent sample** dan menjalankan ulang pengujian, kita dapat melihat bahwa Summary Report hanya berisi Transaction Controller dan tidak ada durasi dari child samplers.



Summary Report										
Name:	Summary Report									
Comments:										
- Write results to file / Read from file										
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
Transaction Controller	1	2484	2484	2484	0,00	0,00%	7,9/min	0,06	0,00	501,0
TOTAL	1	2484	2484	2484	0,00	0,00%	7,9/min	0,06	0,00	501,0

Tapi jika kita melihat **View Result Tree**, kita masih bisa melihat hasil dari individual sampler.

View Results Tree

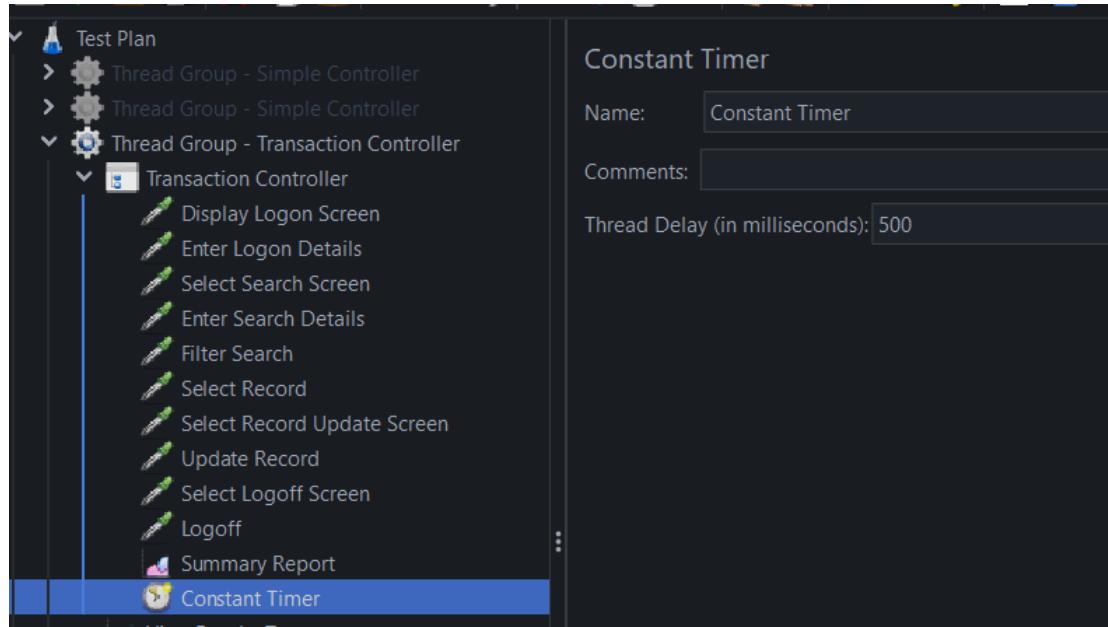
Name:	View Results Tree			
Comments:				
- Write results to file / Read from file				
Filename				
<input type="button" value="Browse..."/>	<input type="button" value="Log/Display Only"/>			
Search: <input type="text"/>	<input type="checkbox"/> Case sensitive	<input type="checkbox"/> Regular exp.	<input type="button" value="Search"/>	<input type="button" value="Reset"/>
<div style="border: 1px solid #ccc; padding: 5px;"> Text Sampler result Request Response data Thread Name: Thread Group - Transaction Controller 1-1 Sample Start: 2025-01-31 11:04:28 WIB Load time: 492 Connect Time: 4 Latency: 37 Size in bytes: 53 Sent bytes: 0 Headers size in bytes: 0 Body size in bytes: 53 Sample Count: 1 Error Count: 0 Data type ("text" "bin" ""): text Response code: 200 Response message: OK </div>				

Opsi lain di Transaction Controller adalah yang diberi label ***Include duration of time and pre-post processor in generated sample***. Jika kita memilih opsi ini.

Transaction Controller

Name:	Transaction Controller
Comments:	
<input checked="" type="checkbox"/> Generate parent sample	
<input checked="" type="checkbox"/> Include duration of timer and pre-post processors in generated sample	

Lalu menambahkan **Constant Timer** dengan penundaan 500 milidetik.



Kita akan menjalankan pengujian **dengan dan tanpa** opsi ini untuk melihat hasilnya.

Summary Report										
Name:	Summary Report									
Comments:										
Write results to file / Read from file										
Filename								Browse...	Log/Display Only:	<input type="checkbox"/>
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/...	Se	Er
Transaction Controller - Include Duration	1	8185	8185	8185	0,00	0,00%	7,3/min	0,06		
Transaction Controller - Do Not Include Duration	1	2573	2573	2573	0,00	0,00%	23,3/min	0,19		
TOTAL	2	5379	2573	8185	2806,00	0,00%	11,2/min	0,09		

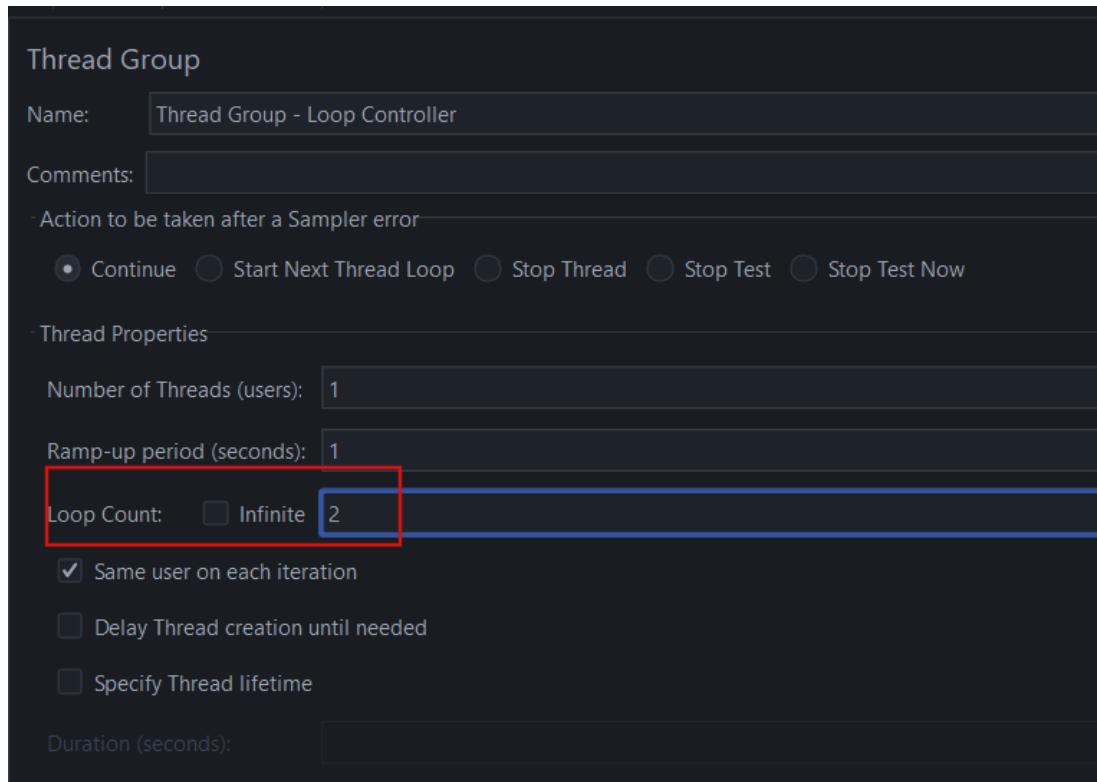
Kita dapat melihat bahwa ketika opsi **include timer** dipilih, waktu respons dari Transaction Controller mencakup delay 500 milidetik antara setiap sampler, dan ketika tidak dipilih, tidak termasuk.

3. Loop Controller

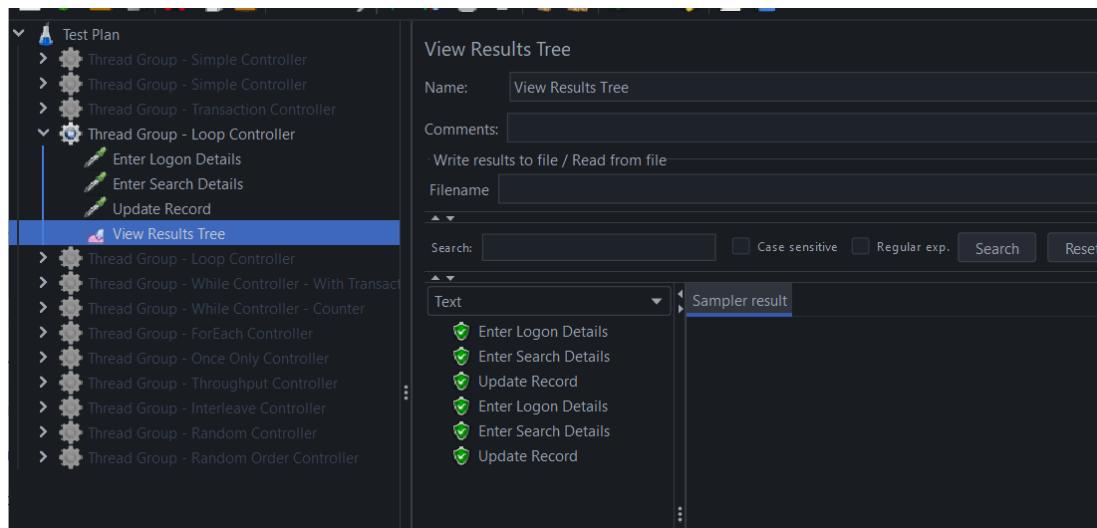
Loop Controller adalah cara untuk mengatur jumlah iterasi yang dijalankan oleh samplers, memberikan lebih banyak kontrol atas scenario kita.

The screenshot shows the configuration panel for a 'Loop Controller'. It has fields for 'Name' (set to 'Loop Controller'), 'Comments', and 'Loop Count' (set to 1). There is also an unchecked checkbox labeled 'Infinite'.

Berikut ini adalah Test Plan yang mengelola jumlah loop menggunakan Thread Group

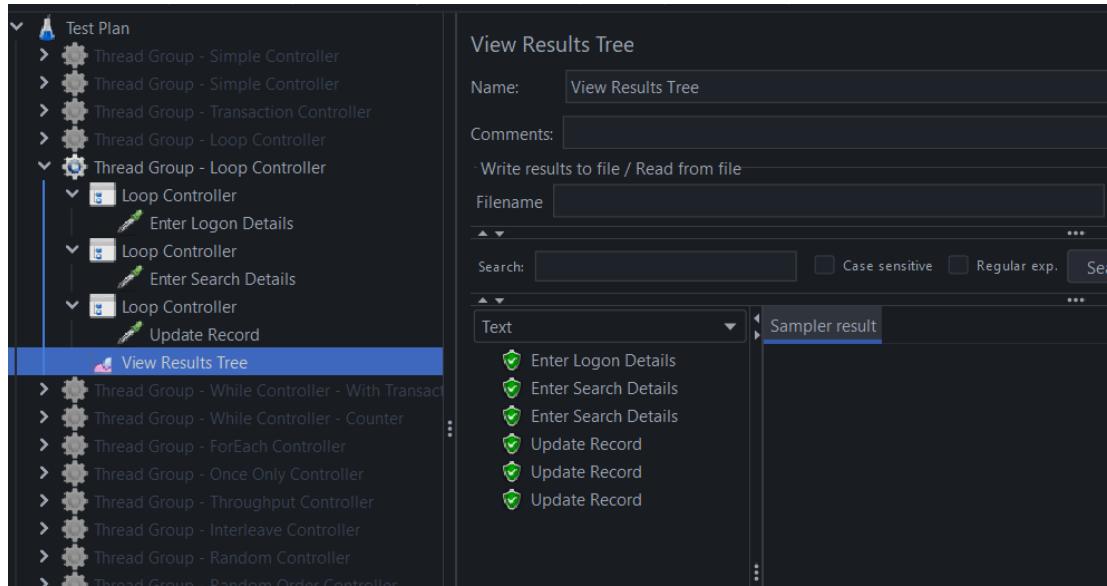


Kita akan melihat bahwa setiap sampler dieksekusi 2 kali sesuai dengan jumlah yang kita atur di Thread Group.



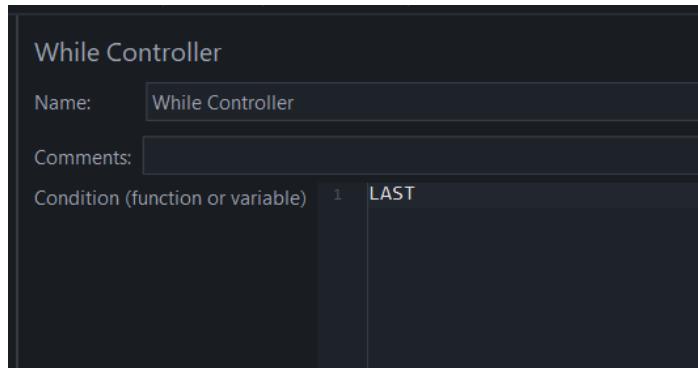
Jika kita ingin setiap sampler berulang dengan jumlah yang berbeda, kita bisa menggunakan Loop Controller. Kita akan mengatur Loop Count di Thread Group kembali ke 1 dan menambahkan beberapa Loop Controllers untuk mengontrol setiap sampler.

Kita akan mengatur Loop Count untuk Loop Controller pertama menjadi 1, yang kedua menjadi 2, dan yang ketiga menjadi 3. Jika kita menjalankan pengujian, kita akan melihat hasilnya.



4. While Controller

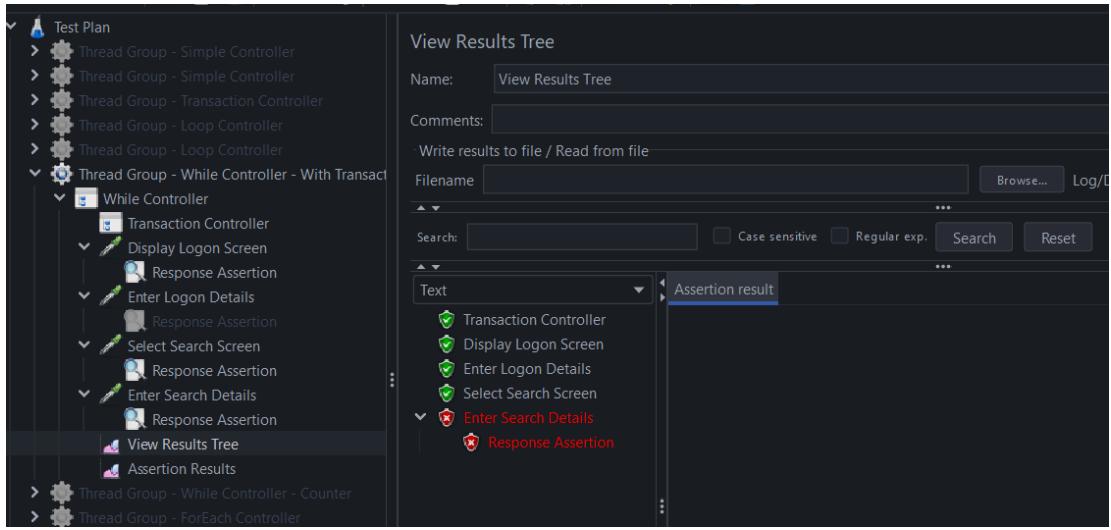
While Controller akan menjalankan elemen-elemen didalamnya selama kondisi yang ditentukan bernilai **true**.



Jadi, While Loop hanya akan keluar setelah kondisi yang kita periksa menjadi **false**. Jika kita membiarkan kondisi kosong, maka loop akan keluar ketika sampel terakhir dalam loop gagal.

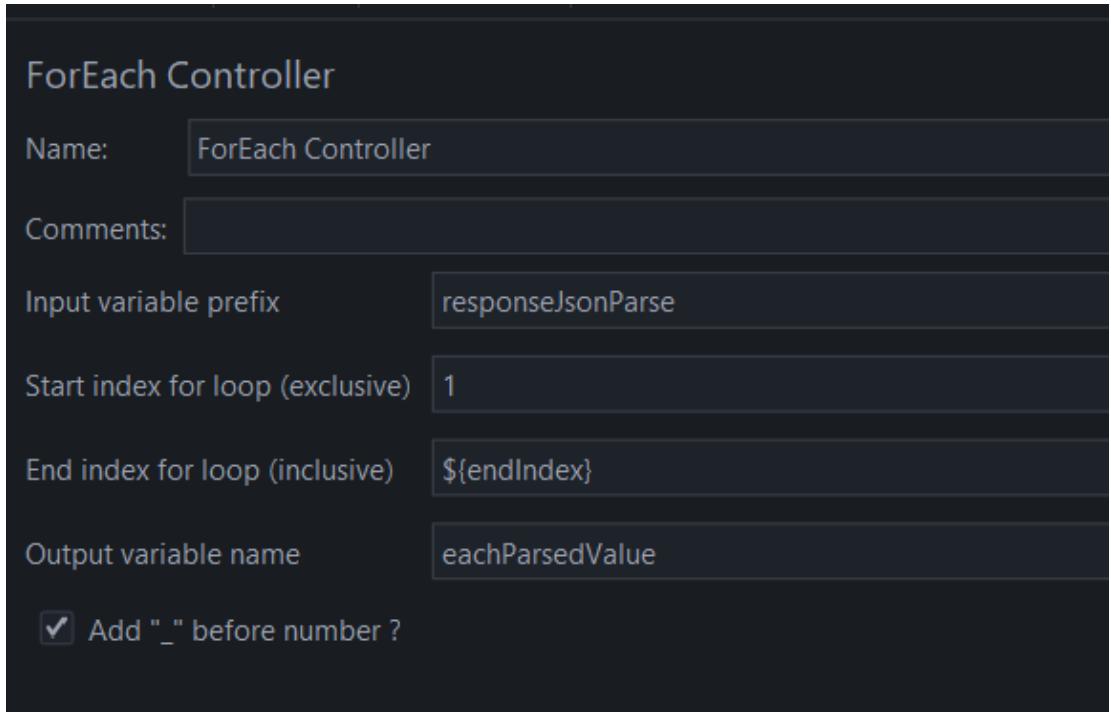
Kita akan menambahkan **Response Assertion** ke setiap Dummy Sampler dalam pengujian kita untuk memeriksa kode respons **200**.

- Jika kita mengatur kondisi kita ke **LAST** dan menjalankan pengujian, itu akan berjalan tanpa henti karena sampel terakhir selalu berhasil.
- Jika kita mengatur kode respons dari Dummy Sampler terakhir ke sesuatu selain **200**, sampel terakhir akan gagal, dan While Loop akan keluar.



5. ForEach Controller

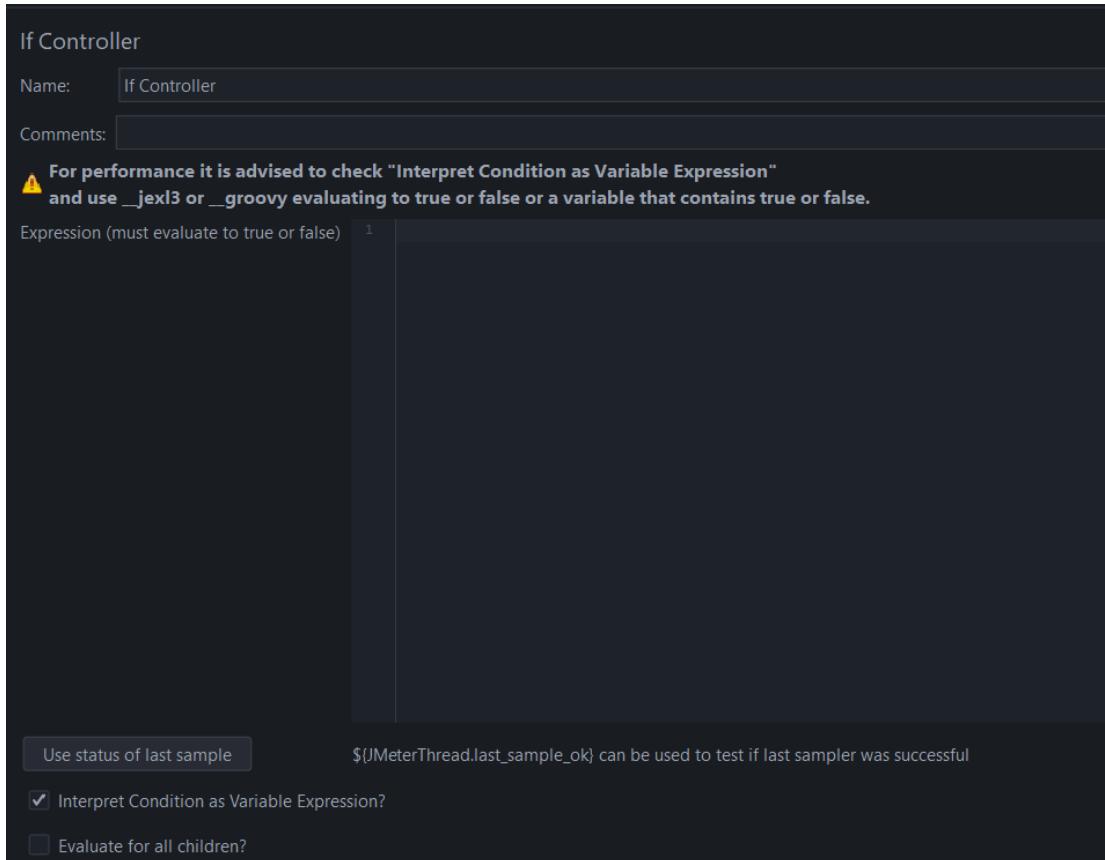
Seperti namanya, ForEach Controller mengulang nilai **array**. Controller ini menerima nilai array kemudian mengiterasi array tersebut sampai selesai.



Ini sangat berguna ketika kita memanggil halaman web atau API yang mengembalikan data yang ingin kita ekstrak dan uji, seperti list URL atau record pelanggan.

6. If Controller

If Controller di JMeter adalah elemen yang memungkinkan kita untuk mengeksekusi sampler atau elemen pengujian lainnya berdasarkan kondisi tertentu. Dengan menggunakan If Controller, kita dapat mengontrol alur eksekusi pengujian dengan cara yang lebih dinamis, sehingga hanya bagian tertentu dari test plan yang akan dijalankan jika kondisi yang ditentukan terpenuhi.



Skenario If controller lebih kompleks, untuk referensi penggunaan scenario if controller, teman-teman bisa lihat referensi [di sini](#).

7. Once Only Controller

Once Only Controller adalah controller yang sangat sederhana, yang hanya menjalankan elemen di dalamnya sekali.



Ini biasanya digunakan untuk case login atau logout dari pengujian kita yang mungkin hanya ingin dilakukan sekali per eksekusi pengujian.

8. Throughput Controller

Controller ini tidak benar-benar mengontrol throughput dari samplers dalam pengujian, tetapi memungkinkan kita untuk mengontrol seberapa sering child samplers nya dieksekusi.

Ada dua mode untuk Throughput Controller ini:

- Percent executions
- Total executions

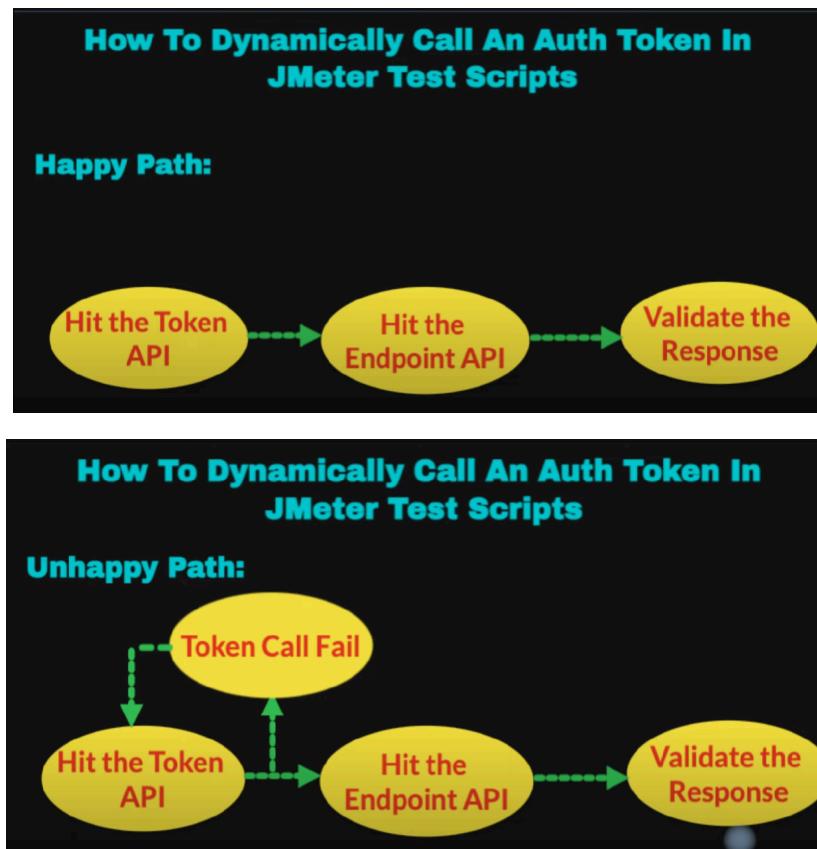
Throughput Controller

Name:	Throughput Controller
Comments:	
Based on:	Total Executions
Throughput:	10
<input checked="" type="checkbox"/> Per User	

Dynamically Call Auth Token in Jmeter (Real Scenario)

Dalam real project, biasanya project kita akan memerlukan real authentication untuk mengakses sebuah endpoint. Pertanyaannya adalah bagaimana memanggil token autentikasi secara dinamis dalam script pengujian JMeter.

Berikut ini adalah gambaran use case untuk memanggil token autentikasi secara dinamis.



Solusinya adalah melibatkan pendekatan langkah demi langkah sebagai berikut:

1. Menggunakan ekspresi reguler untuk mengekstrak token
2. Menyimpannya dalam variabel
3. Dan meneruskannya secara dinamis ke API Endpoint

Dengan menerapkan solusi ini, pengujian dapat mengatasi tantangan token yang kadaluarsa selama eksekusi dan memastikan pengujian berhasil.

Dalam case ini, kita memerlukan endpoint untuk skenario diatas. Untuk endpoint ini kita akan menggunakan endpoint dari Sandbox Paypal

<https://developer.paypal.com/docs/api/orders/v2/>

Prerequisite :

1. Sebelum menggunakan API tersebut, pastikan teman-teman sudah memiliki akun paypal.
2. Jika sudah memiliki akun paypal, silahkan masuk ke **PayPal Developer Dashboard**.
3. Setelah masuk ke dashboard, pilih menu **Apps & Credential**.
4. Kemudian teman-teman akan melihat informasi **Client Secret** dan **Password** untuk digunakan sebagai **basic auth** di postman nanti, untuk meng generate Bearer Token teman-teman.

The screenshot shows the PayPal Developer Dashboard interface. At the top, there's a navigation bar with links for Docs, APIs & SDKs, Tools, Video Library, Help, Business Dashboard, and user profile. Below the navigation, there are tabs for Home, Apps & Credentials (which is selected), Testing Tools, and Event Logs. A toggle switch indicates 'Sandbox' mode is active. A message at the top says 'Accelerate your integration with a new interactive Checkout guide. Explore now'. Below that, a note says 'You're in sandbox mode.' On the right, there's a 'Create App' button. The main content area is titled 'API Credentials' and contains a message: 'Viewing sandbox API credentials. Upgrade your account to PayPal for Business to view live credentials.' A table lists the following data:

App name	Client ID	Secret	Created date
Default Application	Afjio0dfNeNhDoVJysjSz3...	EL6C22oiwFjNRgxTYZK7vEcR4CFvZL2us9IT7Ug Po4tt9rK8vtG6nJdF3niZGC71FYzplPvoAHtYmZW	31/05/24, 13:58

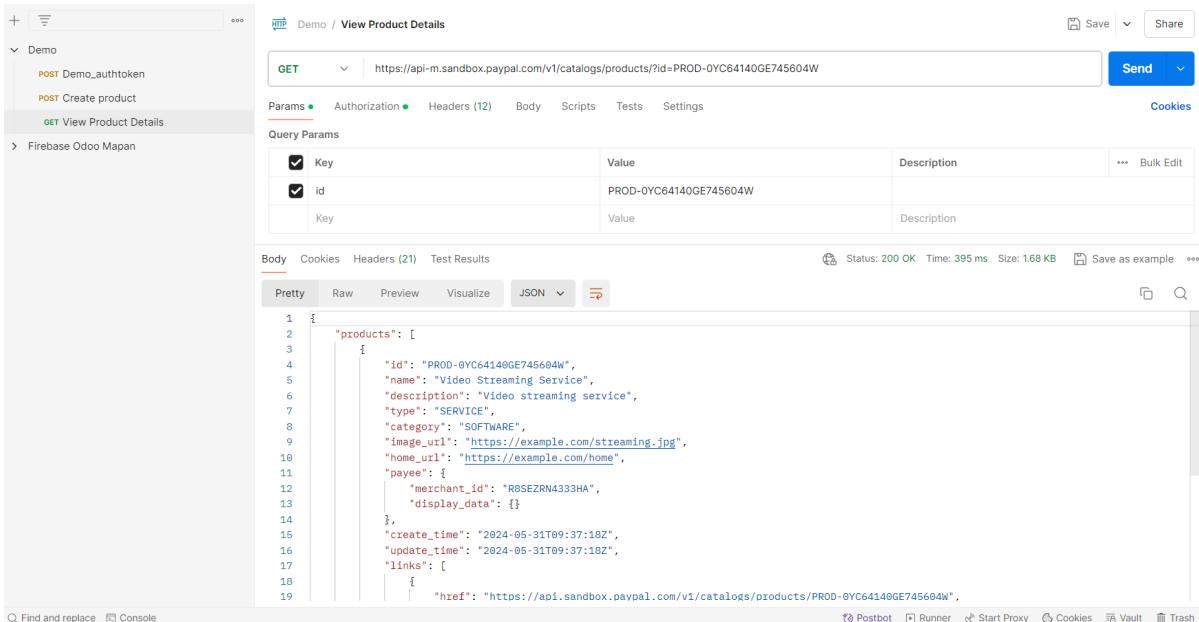
5. Dalam case sekarang, kita akan implementasi 3 endpoint sebagai berikut:

The screenshot shows the PayPal API documentation for the Catalog Products endpoint. The left sidebar has a tree view of resources: API responses, Core Resources, Overview, Add Tracking, Catalog Products (selected), Catalog Products (Create product, List products, Show product details, Update product), Definitions, Disputes, Identity, Invoicing, Orders, Partner Referrals, Payment Experience, and Payment Method Tokens. The main content area is titled 'Show product details' and shows the following details:

- API endpoint:** GET /v1/catalogs/products/{product_id}
- Security:** OAuth2
- Request:**
 - PATH PARAMETERS:** product_id (required, string, The product ID)
 - HEADER PARAMETERS:** Authorization (required, string, To make REST API calls, include the bearer)
- Request samples:** curl -v -X GET https://api-m.sandbox.paypal.com/v1/catalogs/products/{product_id} -H 'X-PAYPAL-SECURITY-CONTEXT: {"consumer":{"accountNumber": "1234567890123456"}, "clientType": "MOBILE_APP", "clientName": "MyApp", "clientVersion": "1.0.0", "language": "ENGLISH", "country": "INDIA", "ipAddress": "192.168.1.100"}' -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'Prefer: return=representation'
- Response samples:** 200 application/json (JSON response object)

Copy paste curlnya ke postman, dan create 3 request di postman :

- Login Auth : <https://developer.paypal.com/api/rest/authentication/>
- Create Product : <https://developer.paypal.com/docs/api/catalog-products/v1/>
- View Product Details :
https://developer.paypal.com/docs/api/catalog-products/v1/#products_get



The screenshot shows the Postman interface with a successful API call. The URL is https://api-m.sandbox.paypal.com/v1/catalogs/products/?id=PROD-0YC64140GE745604W. The response status is 200 OK, time is 395 ms, and size is 1.68 KB. The response body is a JSON object containing product information:

```
1 {
2   "products": [
3     {
4       "id": "PROD-0YC64140GE745604W",
5       "name": "Video Streaming Service",
6       "description": "Video streaming service",
7       "type": "SERVICE",
8       "category": "SOFTWARE",
9       "image_url": "https://example.com/streaming.jpg",
10      "home_url": "https://example.com/home",
11      "payee": {
12        "merchant_id": "R6SEZRN4333HA",
13        "display_data": {}
14      },
15      "create_time": "2024-05-31T09:37:18Z",
16      "update_time": "2024-05-31T09:37:18Z",
17      "links": [
18        {
19          "href": "https://api.sandbox.paypal.com/v1/catalogs/products/PROD-0YC64140GE745604W"
20        }
21      ]
22    }
23  ]
24}
```

Skenario yang akan kita jalankan nanti, kita memerlukan bearer token untuk membuat order dan melihat order details, tapi bearer token disini hanya berlaku selama 30 menit, sementara kita akan membuat automate test dengan durasi 1 hour. Berikut adalah langkah-langkah untuk memulai pengujian tersebut.

Step 1 : Prepare Project Folder

JMeter memiliki banyak elemen konfigurasi untuk mendefinisikan beberapa variabel yang dapat digunakan dalam pengujian JMeter.

Contoh Elemen Konfigurasi yang Sering Digunakan:

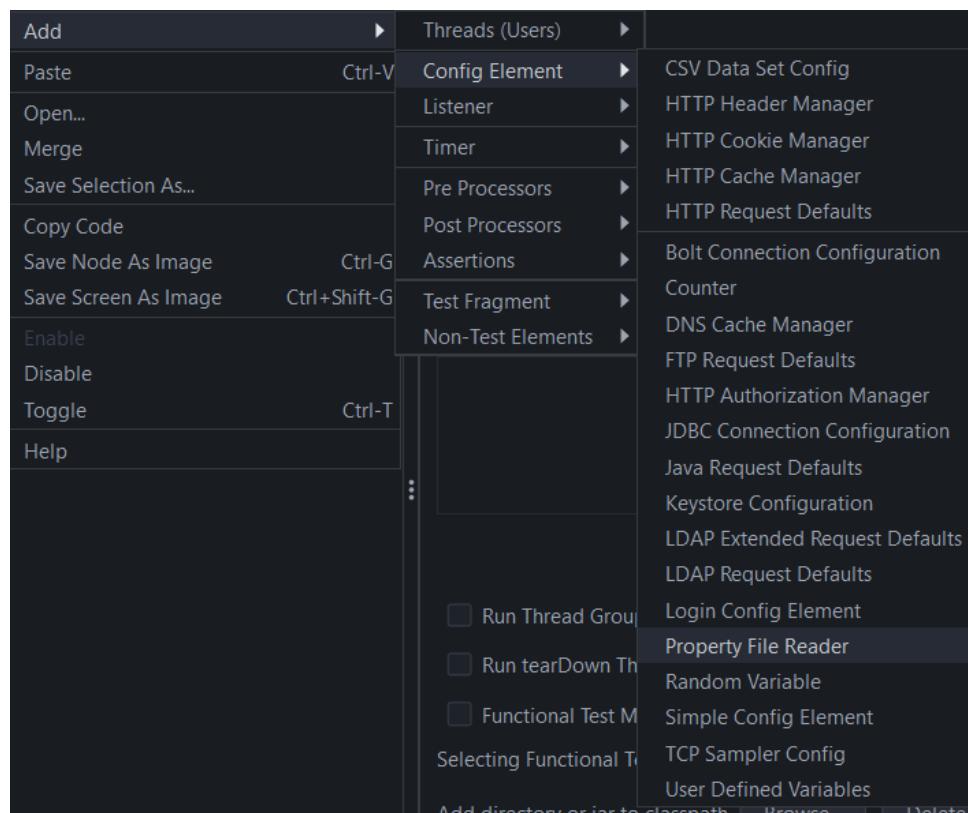
- **User Defined Variables:** Digunakan untuk membuat nama dan nilai variabel.
- **CSV Data Set Config:** Digunakan untuk membaca data pengujian dari file CSV.

Namun, jmeter juga memiliki elemen untuk membaca file properti pengguna! Berikut adalah langkah-langkah menggunakan pembaca File Properti di Jmeter :

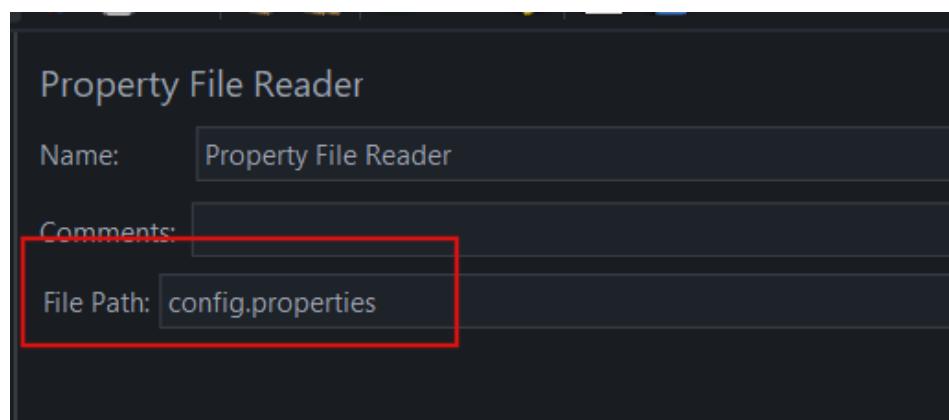
1. Persiapkan File Properti: Buat file **config.properties** yang berisi variabel yang ingin kita gunakan dalam pengujian. Contoh properties dalam file :

```
APP_NAME=Google Project
HOST_URL=www.google.com
APP_PROTOCOL=https
```

2. Tambahkan Elemen Pembaca File Properti:
 - Di JMeter, klik kanan pada Test Plan.
 - Pilih **Add > Config Element > Property File Reader**.
 - Pastikan elemen ini disimpan di bagian atas, tepat di bawah Test Plan.
3. Download dan Ekstrak File:
 - Sebelumnya jika kita tidak melihat Properti File reader under config element, kita perlu mendownload kita perlu mengunduh **tag-jmeter-extn-1.1.jar** dari [sumber yang terpercaya](#).
 - Ekstrak file tersebut dan letakkan di folder JMETER_HOME/lib/ext agar JMeter dapat mengenali elemen baru ini.
 - Kemudian restart Jmeter, dan kita bisa melihat elemen baru ini



4. Pilih elemen Property File Reader untuk menambahkannya di bawah Test Plan, dan masukkan path File Propertinya



5. Simpan Test Plan, di folder yang sama dengan config.properties file kita

6. Under Test Plan, tambahkan variabel dengan nama yang sama di config properties file

User Defined Variables		
	Name:	Value
APP_NAME	Name: APP_NAME	Value: \${__P(APP_NAME)}
APP_PROTOCOL	Name: APP_PROTOCOL	Value: \${__P(APP_PROTOCOL)}
HOST_URL	Name: HOST_URL	Value: \${__P(HOST_URL)}

7. Tambahkan value dari variable dengan format **`__P(APP_NAME)`**
 8. Untuk mengecek properties file ini berhasil, bisa coba add request, tambahkan variabel tersebut dan running apakah requestnya berhasil.

HTTP Request

Name: HTTP Request

Comments:

Web Server

Protocol [http]: ROTOCOL Server Name or IP: \${HOST_URL} Port Number:

HTTP Request

GET Path: Content encoding:

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display

Search: Case sensitive Regular exp. Search Reset

Text

HTTP Request

Sampler result Request Response data

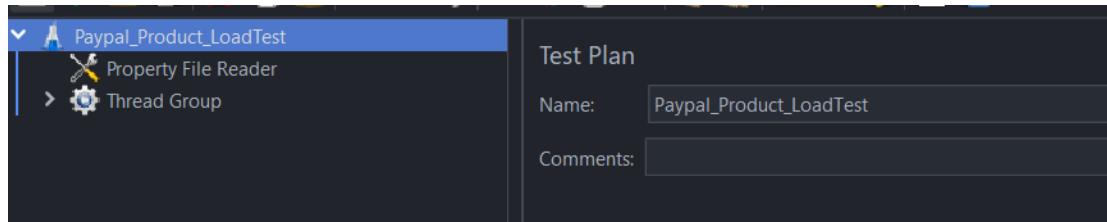
Thread Name: Thread Group 1-1
 Sample Start: 2025-01-31 18:11:38 WIB
 Load time: 324
 Connect Time: 228
 Latency: 302
 Size in bytes: 23095
 Sent bytes: 116
 Headers size in bytes: 1174
 Body size in bytes: 21921
 Sample Count: 1
 Error Count: 0
 Data type ("text"|"bin"|""): text
 Response code: 200
 Response message: OK

Step 2 : Merancang solusi untuk load testing otomatis untuk "show product details' API

- Buat Test Plan baru dengan format penamaan yang tepat

<applicationName>_<typeoftest>

Berikut adalah pengaturan pada naming convention yang digunakan



- Analisis API 'show product details' di Postman

A screenshot of the Postman interface. A GET request is made to 'https://api-m.sandbox.paypal.com/v1/catalogs/products/?id=PROD-0YC64140GE745604W'. The response status is 200 OK, and the JSON body is displayed, showing product details like id, name, description, type, category, and links.

- Tambahkan variabel TEST_DURATION' untuk menentukan durasi load test

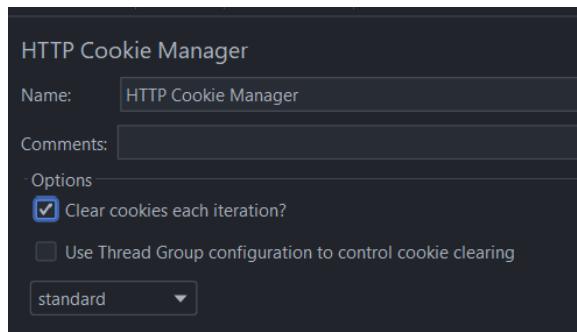
A screenshot of the JMeter 'User Defined Variables' section. It shows variables: APP_NAME, APP_PROTOCOL, HOST_URL, and TEST_DURATION. The TEST_DURATION variable is highlighted with a red box.

Case disini, kita memerlukan bearer token untuk membuat order dan melihat order details, tapi bearer token disini hanya berlaku selama 30 menit.

Casenya, bagaimana kalau kita membuat automate test, dengan durasi 1 hour, jadi akan ada case failed.

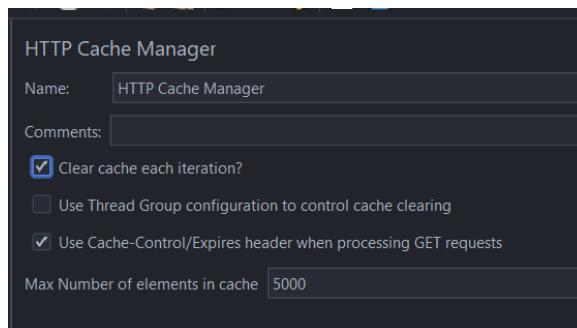
4. Konfigurasikan **HTTP cookie manager** untuk menghapus cookie pada setiap iterasi.

HTTP cookie manager memungkinkan untuk menyimpan informasi sesi dan memungkinkan penambahan cookie secara manual.

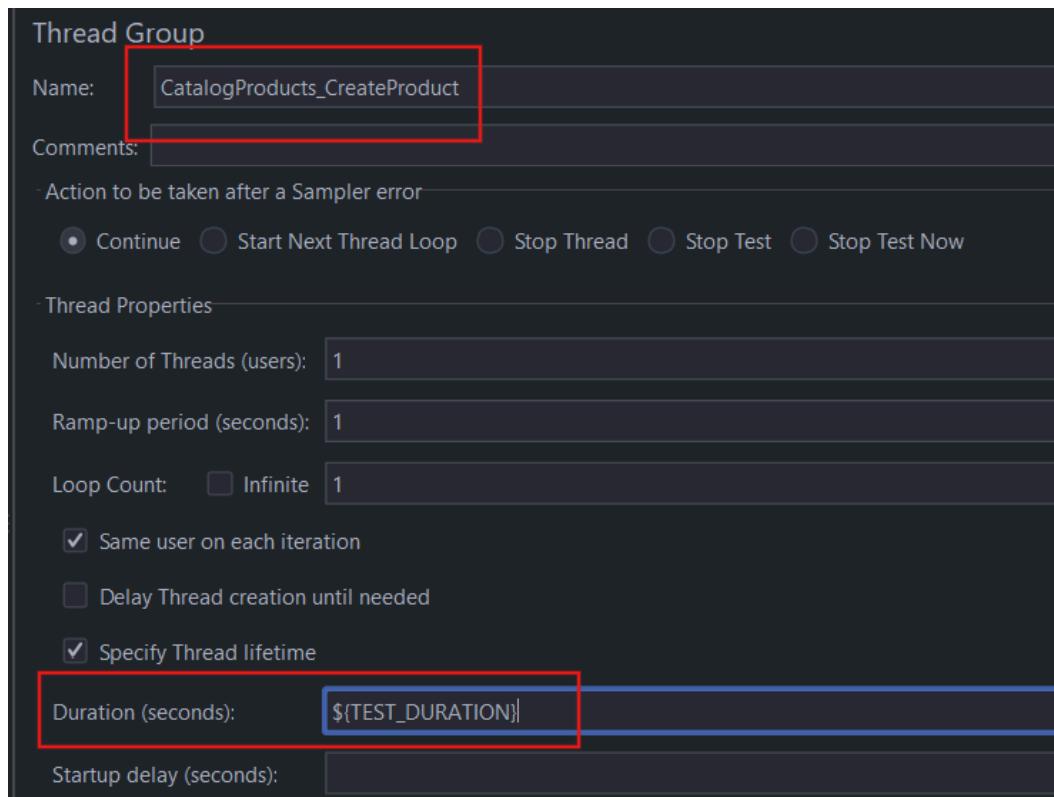


5. Konfigurasikan **HTTP cache manager** untuk menghapus cookie pada setiap iterasi.

HTTP cache manager digunakan untuk caching dan memungkinkan untuk membersihkan cache di setiap iterasi.



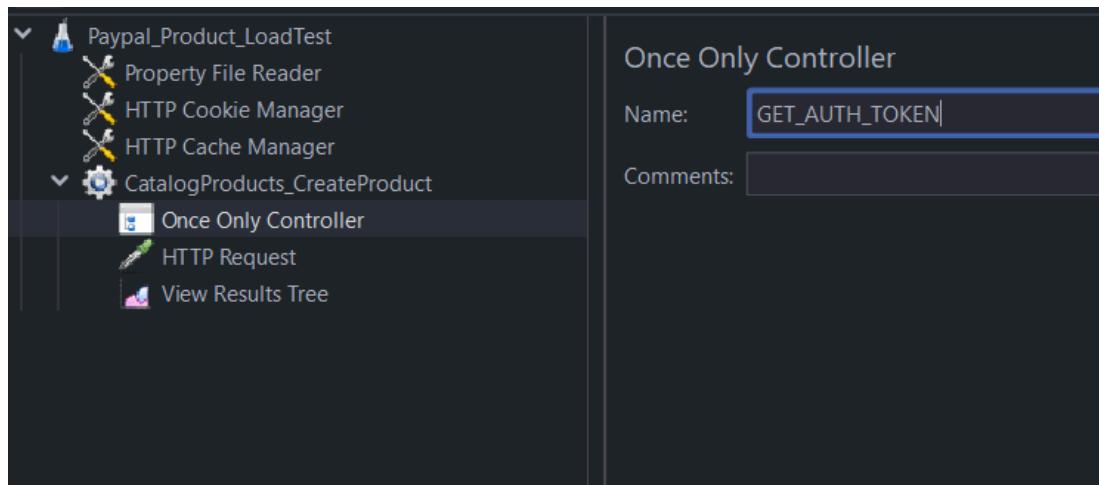
6. Menambahkan **Thread Group** untuk mengontrol jumlah thread dalam test plan dan menetapkan durasi pengujian. Berikut adalah pengaturan pada naming convention yang digunakan untuk Thread Group. <functionalName>_<functionalityName>



Step 3 : Menambahkan “once only controller” untuk menghasilkan token autentikasi

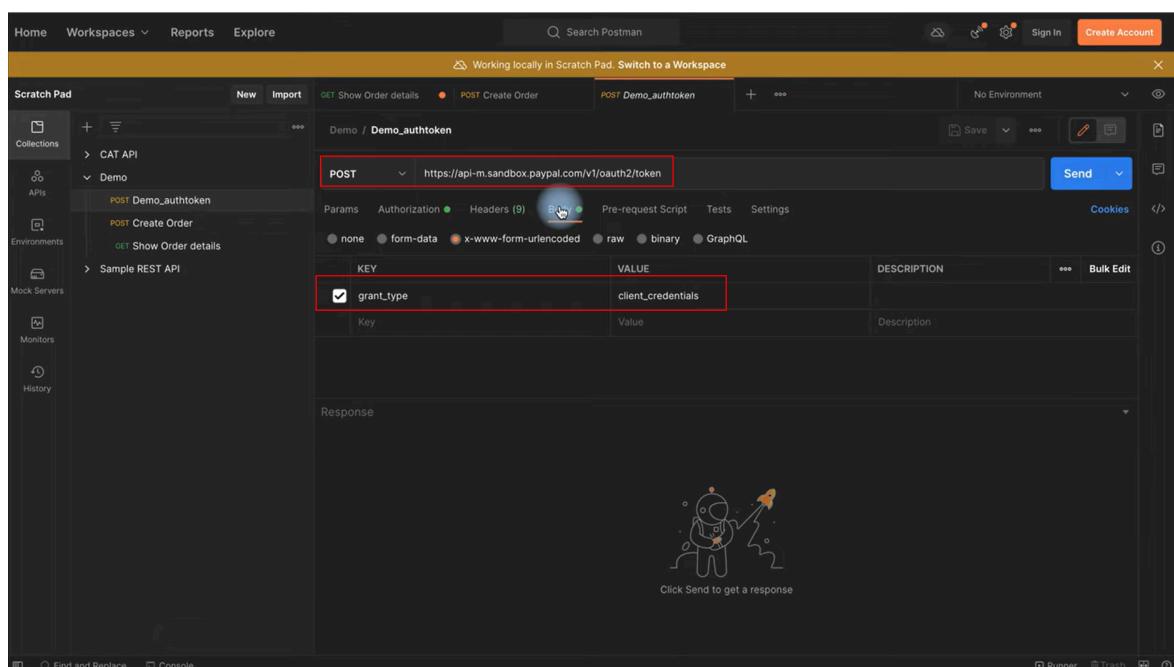
1. **Once only controller** memastikan bahwa pengontrol di dalamnya hanya diproses satu kali per thread.
2. Itu akan melewati permintaan apa pun di bawahnya selama iterasi lebih lanjut.
3. **Once only controller** selalu mengeksekusi selama iterasi pertama dari setiap pengontrol induk perulangan.

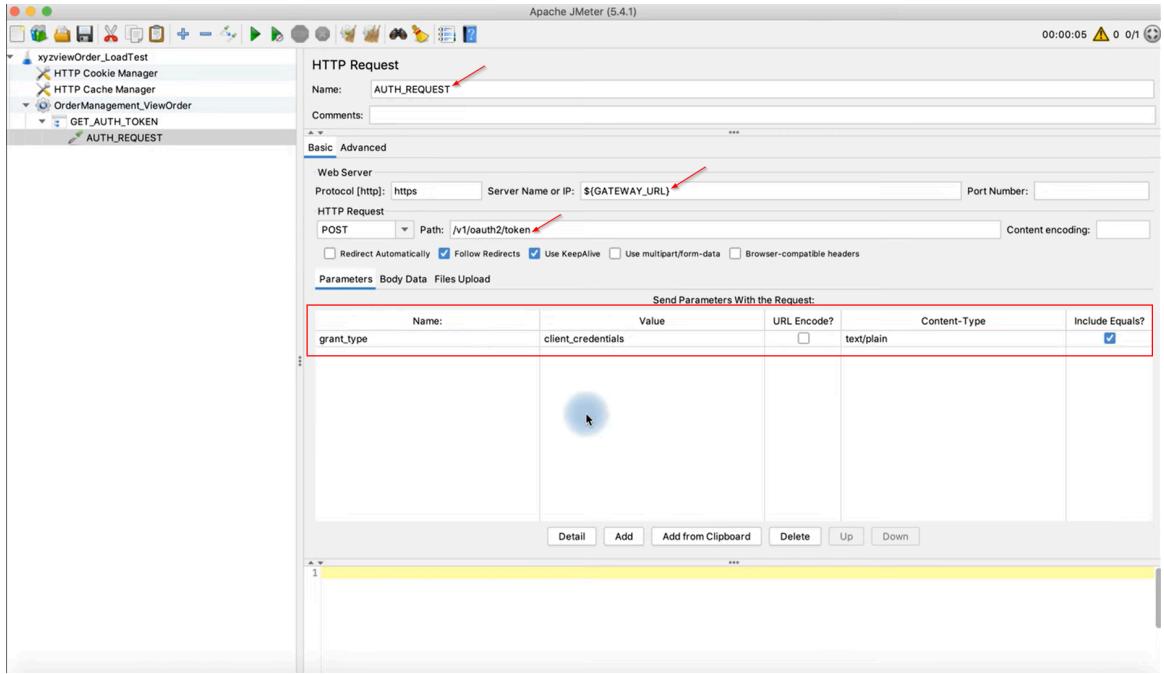
Rename Once Only Controller dengan GET_AUTH_TOKEN



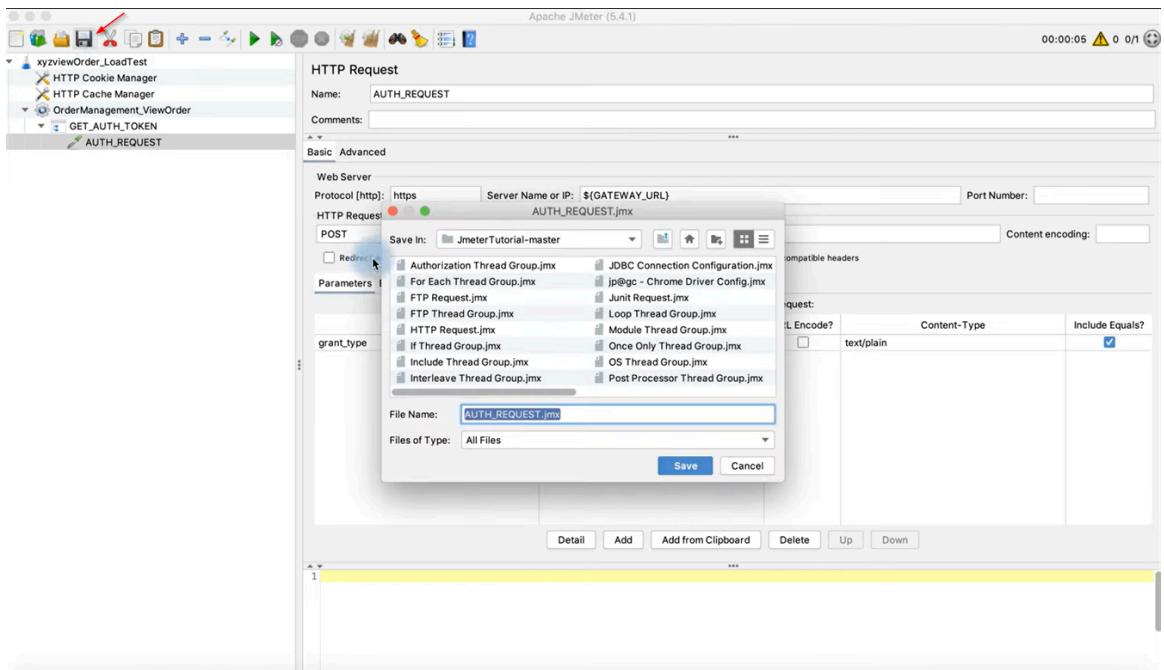
4. Tambahkan HTTP Request untuk **GET auth token**.

Dan konfigurasi untuk HTTP Request bisa dilihat pada settingan postman

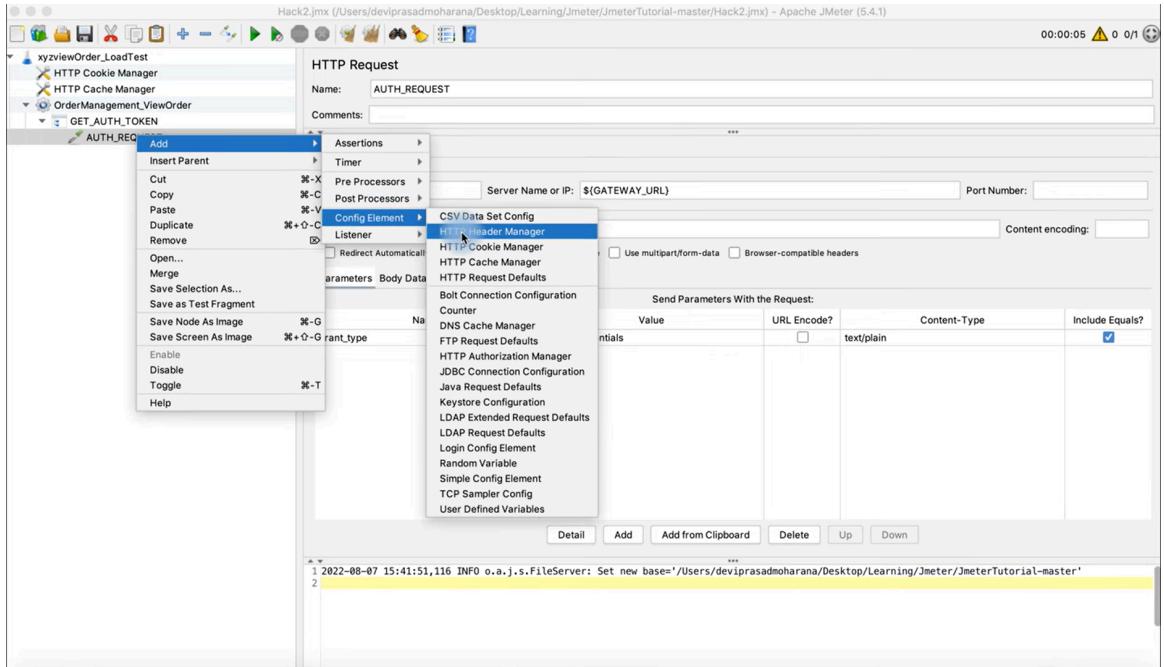




Simpan Test Plan.

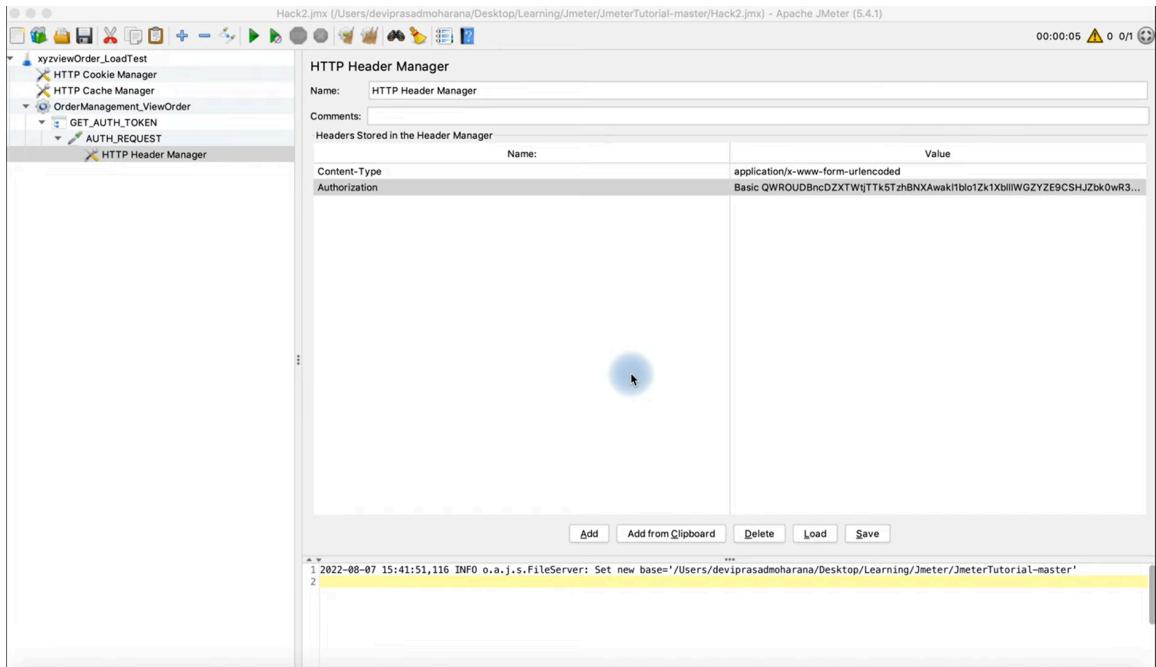


5. Berikan header yang diperlukan di Postman untuk permintaan token. Tambahkan **http header manager** dan sertakan header yang diperlukan.

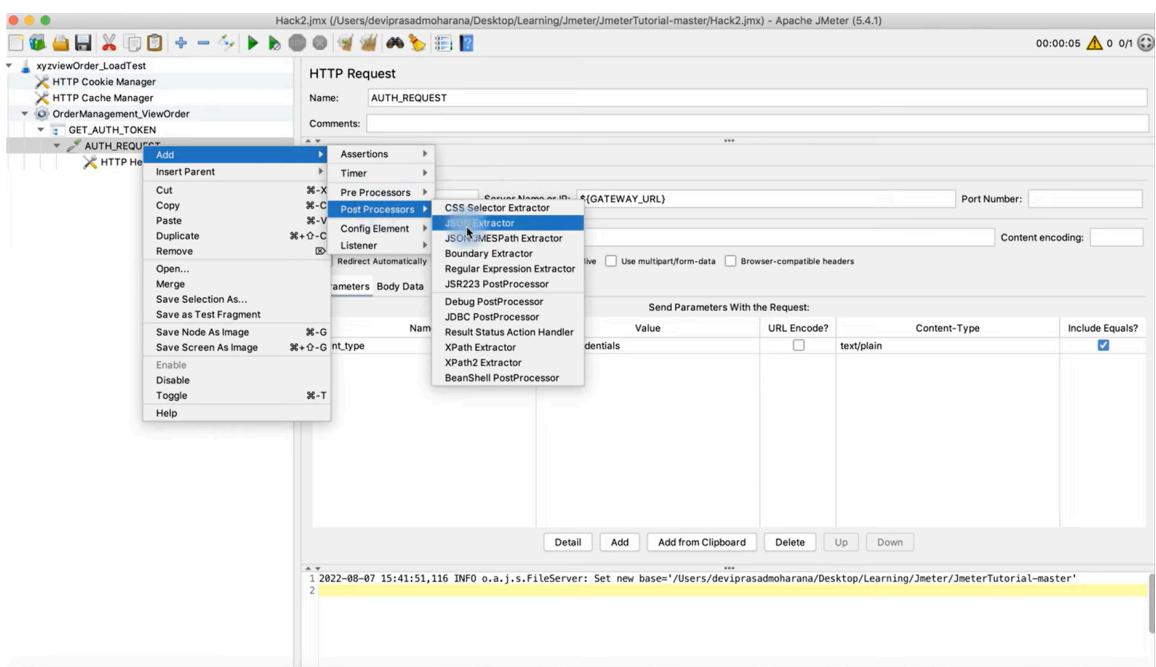


Dan lihat konfigurasi Header pada postman

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
<input checked="" type="checkbox"/> Authorization	Basic QWROUDBncDZXTWtjTTk5TzhBNXAwaklIblo...	<calculated when request is sent>		
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>			
<input checked="" type="checkbox"/> Content-Type	application/x-www-form-urlencoded	<calculated when request is sent>		
<input type="checkbox"/> Content-Length	<calculated when request is sent>			
<input type="checkbox"/> Host	<calculated when request is sent>			



6. Ekstrak dan simpan nilai token menggunakan ekstraktor JSON.



The screenshot shows the Postman interface with a successful API call. The URL is `https://api-m.sandbox.paypal.com/v1/oauth2/token`. The response body is:

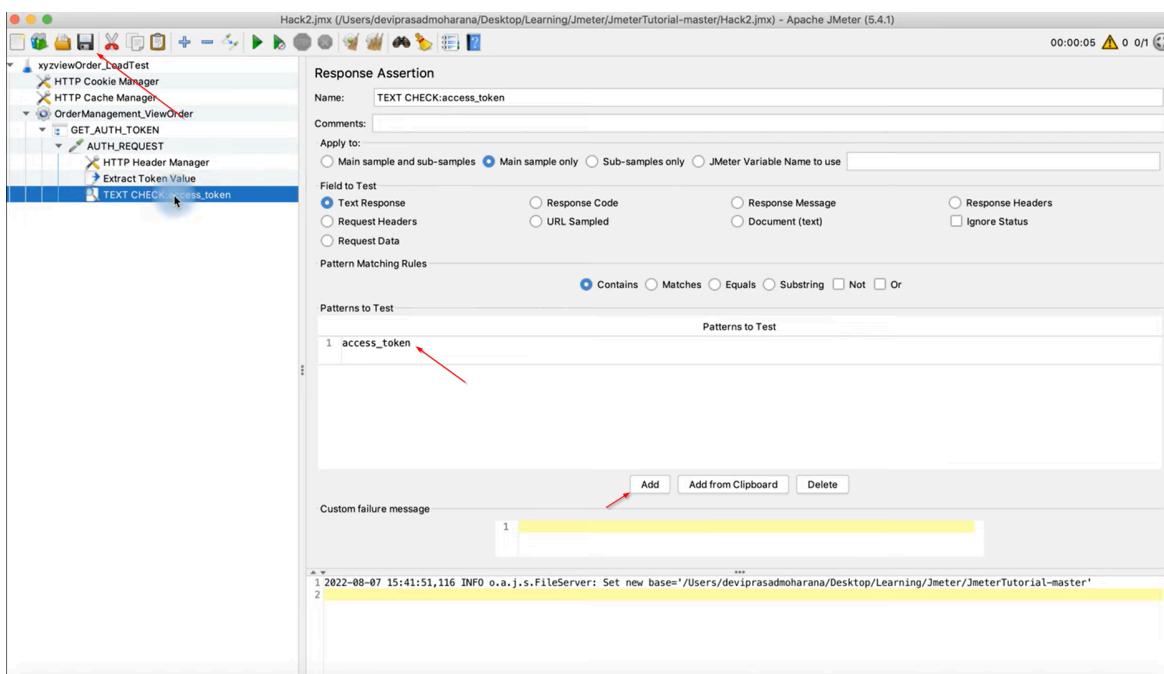
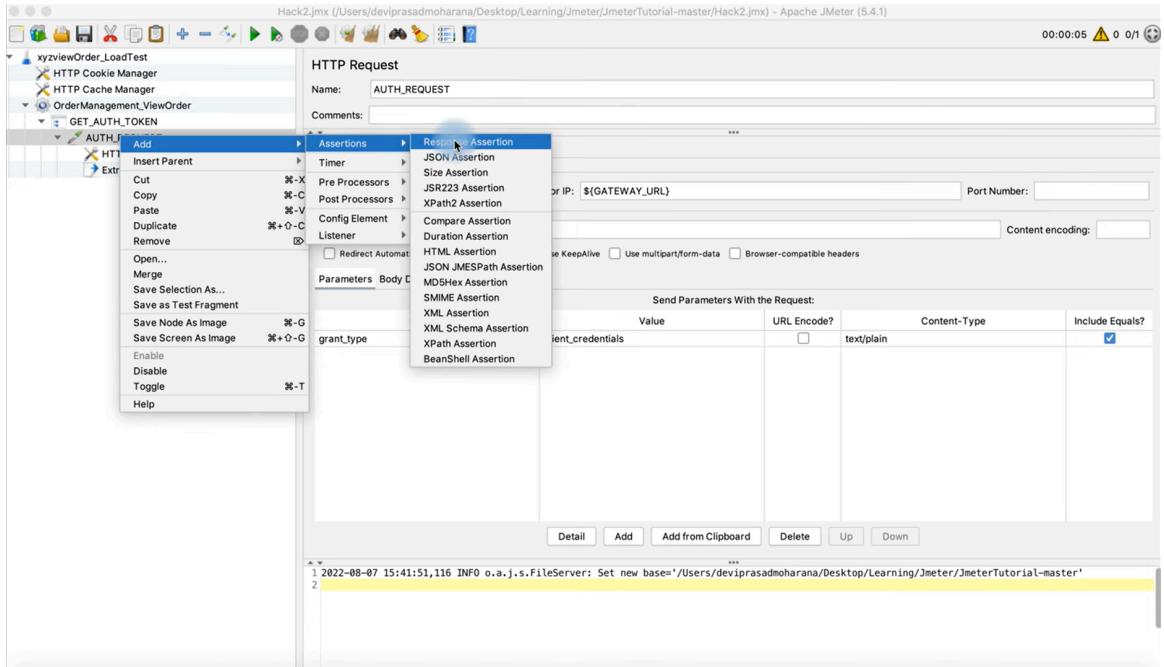
```

2 "scope": "https://uri.paypal.com/services/invoicing https://uri.paypal.com/services/disputes/read-buyer https://uri.paypal.com/services/payments/realmfpayment https://uri.paypal.com/services/disputes/update-seller https://uri.paypal.com/services/payments/payment/authcapture openid https://uri.paypal.com/services/disputes/read-seller https://uri.paypal.com/services/payments/refund https://api.paypal.com/v1/vault/credit-card https://api.paypal.com/v1/payments/* https://uri.paypal.com/payments/payouts https://api.paypal.com/v1/vault/credit-card/* https://uri.paypal.com/services/subscriptions https://uri.paypal.com/services/applications/webhooks"
3 "access_token": "E01AAKThplppGyjGmSEPiTNoH2LCM60eSEikU-LtoCRf34OnyKhes5g29upJ-AcIJUVs-U-Aje48LY210WtjI"
4 "token_type": "Bearer",
5 "app_id": "APP-80W28448SP519543T",
6 "expires_in": 32400,
7 "nonce": "2022-08-07T10:16:10ZPnQjU_13owmGzX34ke00AX21Iauk1zPfv01VNFE92Zg"
8 }

```

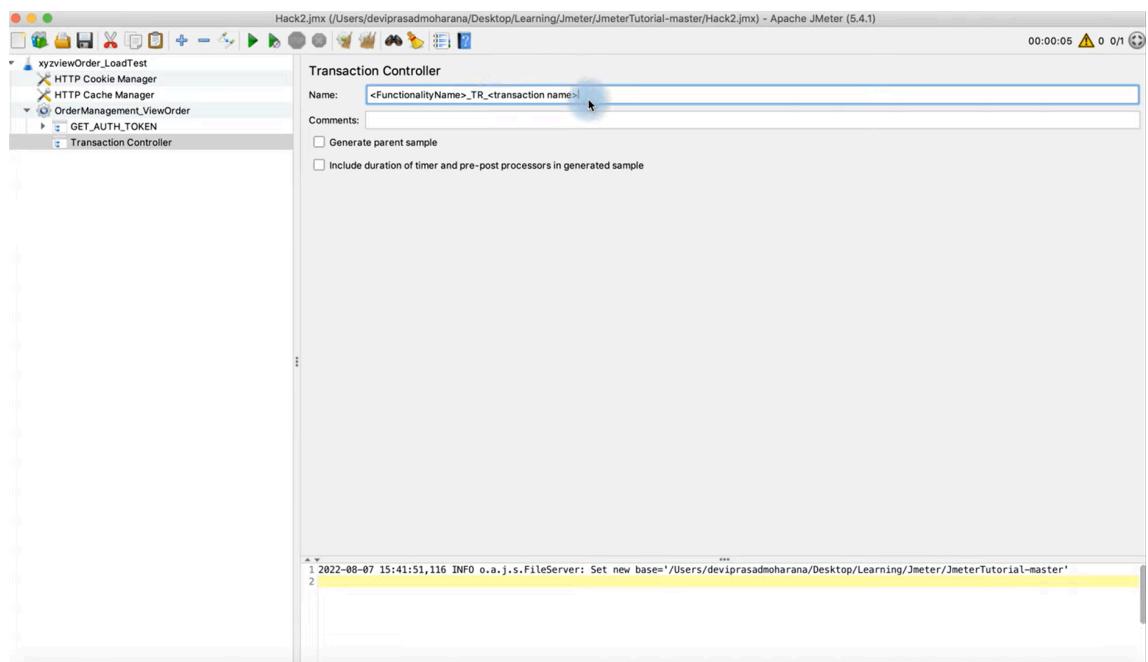
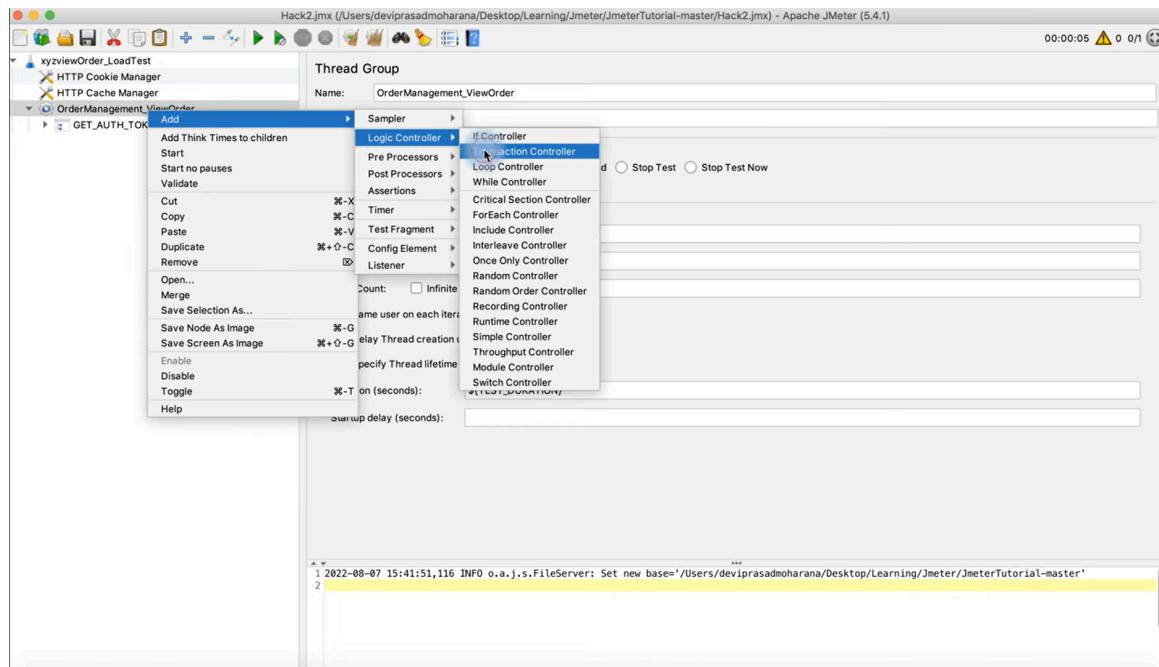
The screenshot shows the Apache JMeter interface with a JSON Extractor configuration. The 'Names of created variables' field contains `token` and the 'JSON Path expressions' field contains `$.access_token`.

7. Lalu verifikasi token tersebut dengan menggunakan Response Assertion

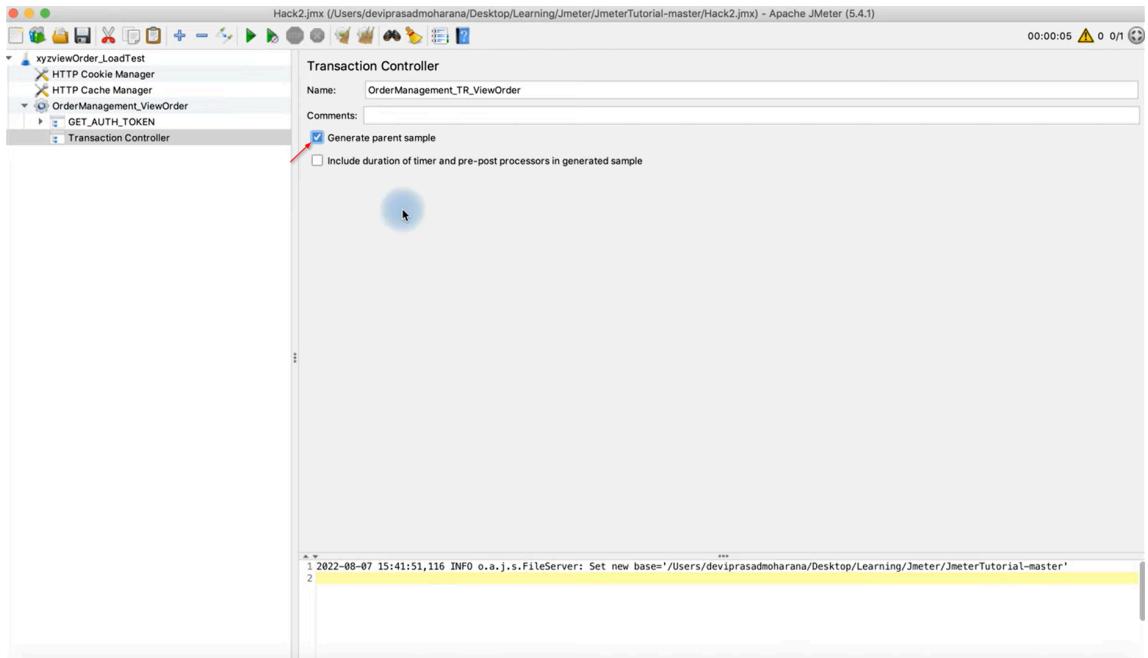


Step 4 : Menerapkan API target (Transaction Controller)

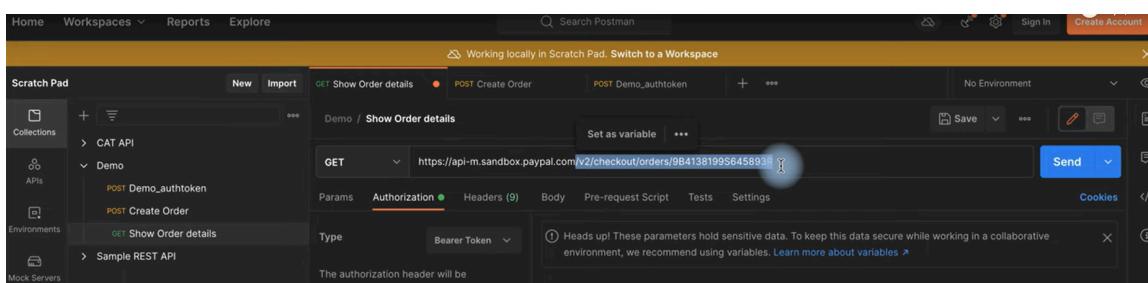
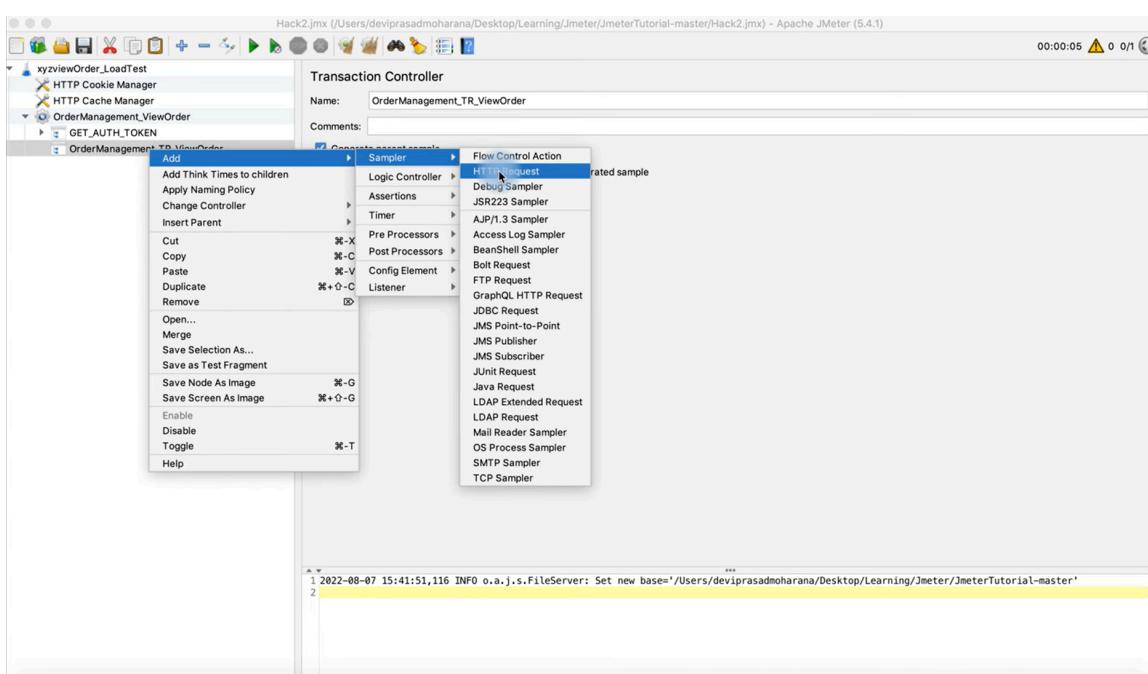
1. Tambahkan Transaction Controller ke thread group
2. Ikuti konvensi penamaan untuk Transaction Controller

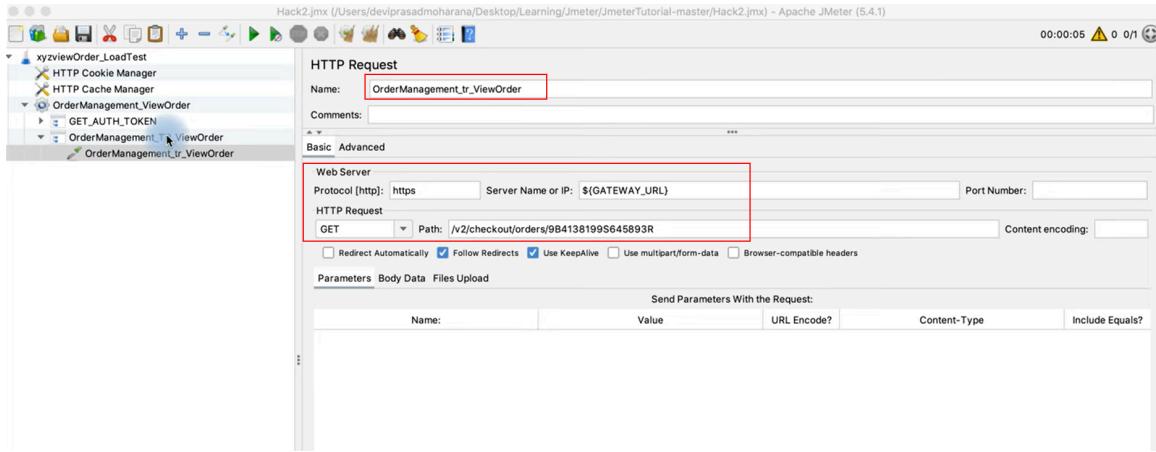


Contoh :

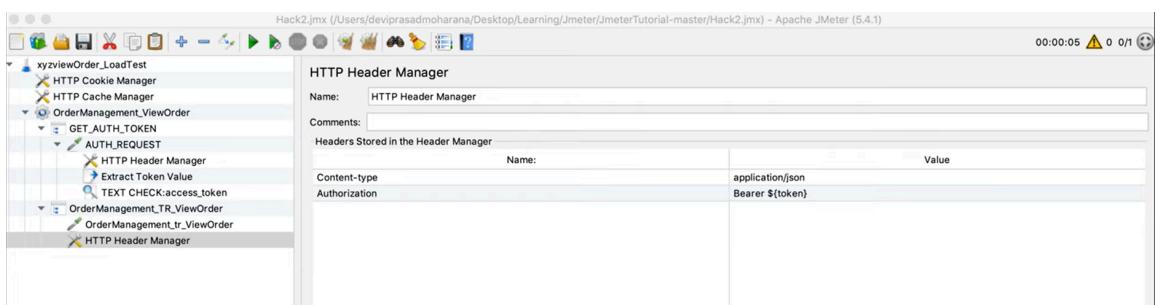
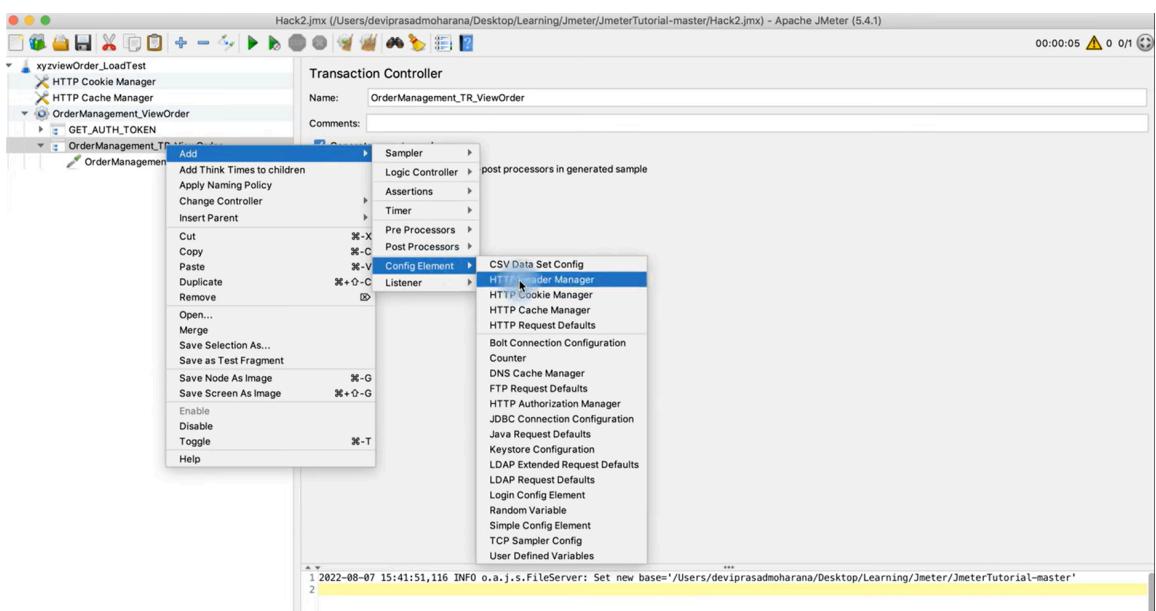


3. Tambahkan HTTP request dan konfigurasikan API target



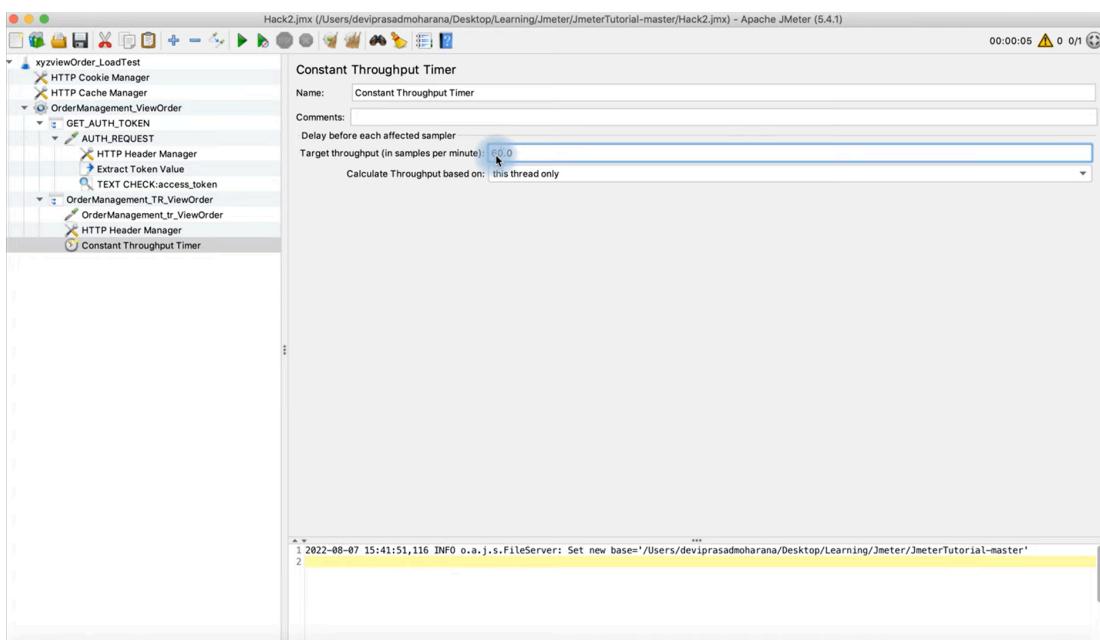
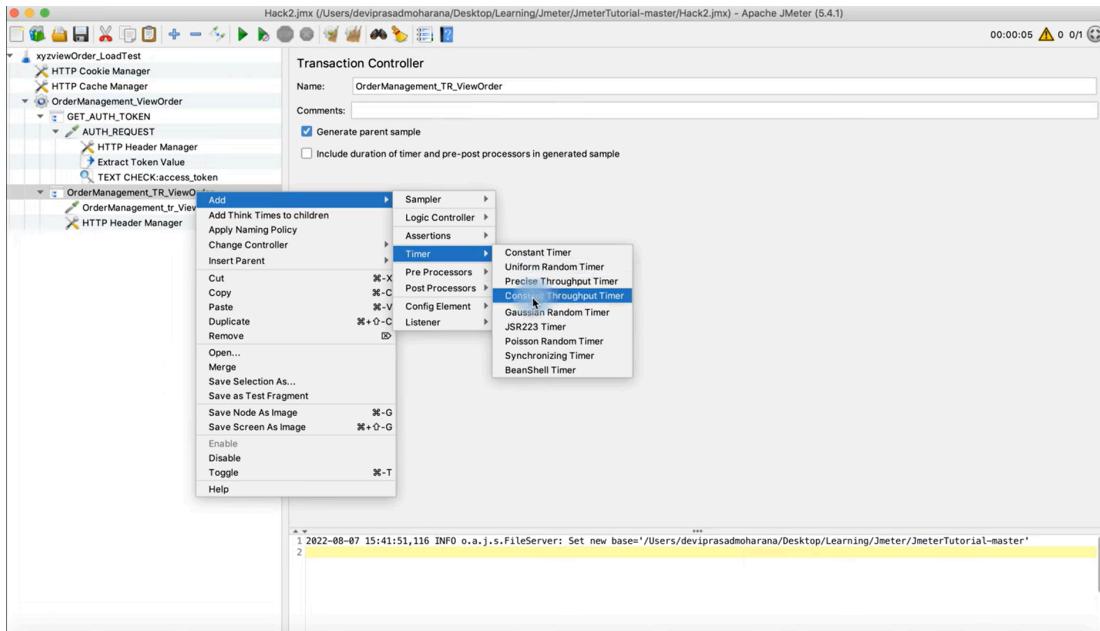


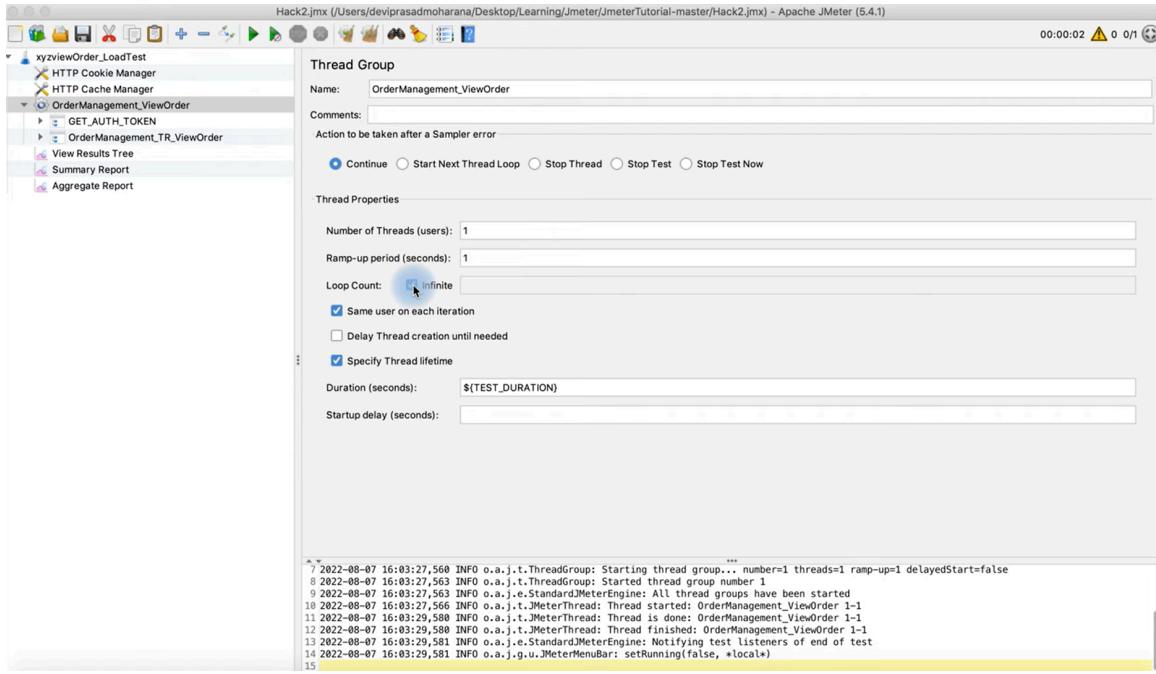
4. Tambahkan HTTP header manager dan atur token otorisasi



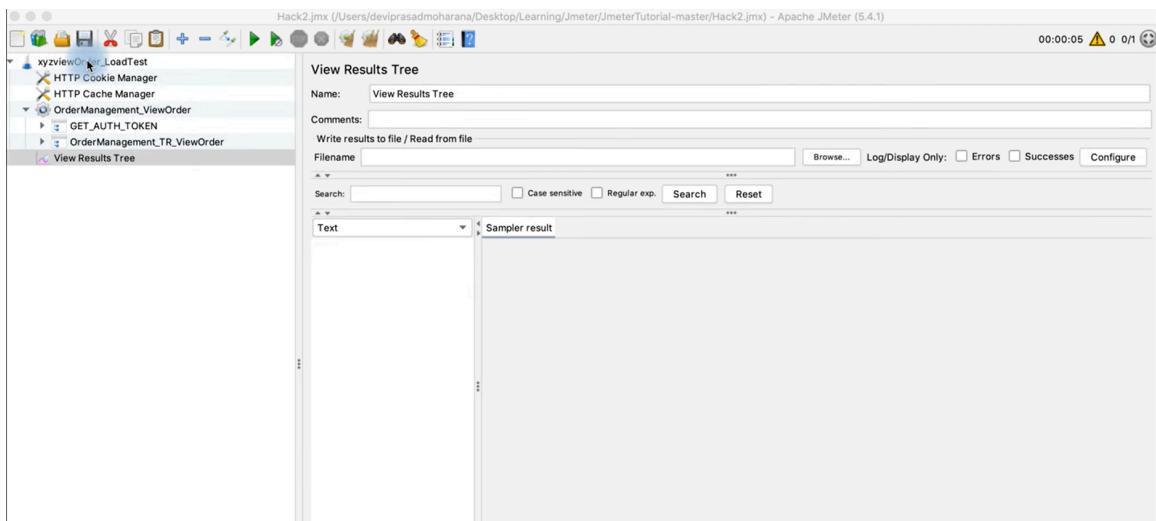
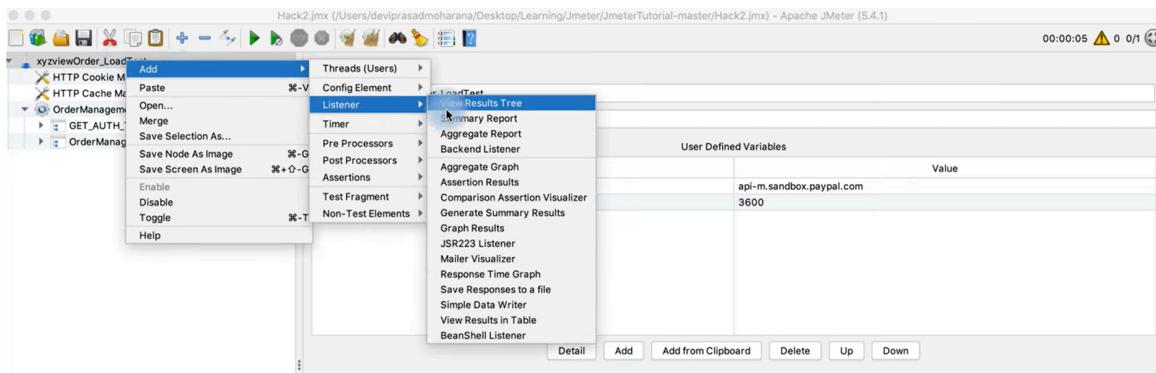
5. Tambahkan constant throughput timer

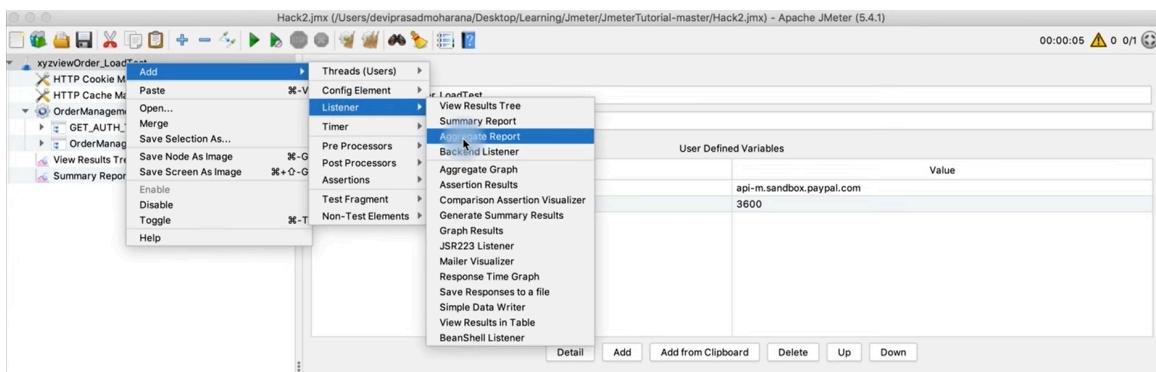
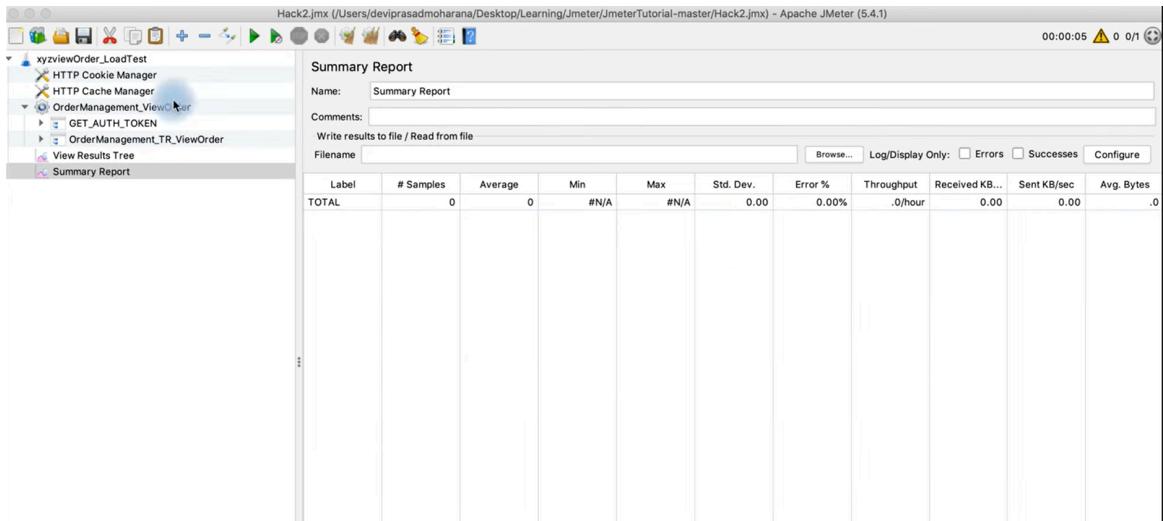
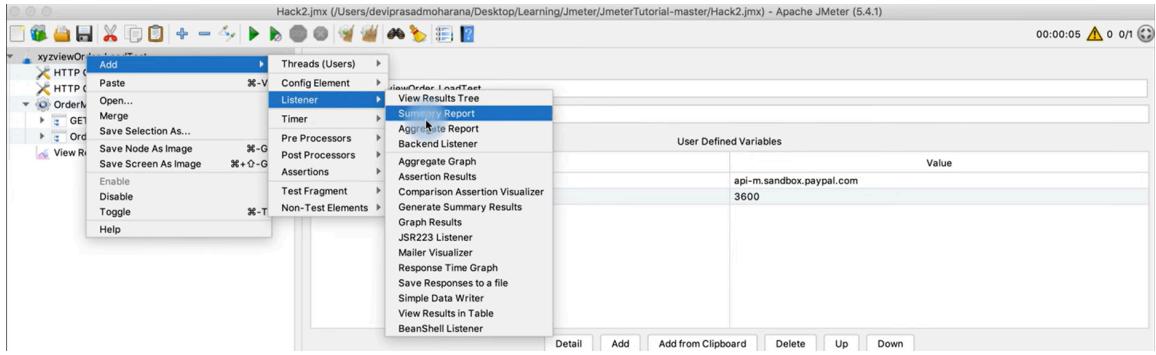
- Untuk mencapai satu TPS (Transcation Per Second), target nilai throughput perlu ditetapkan sebesar 60 sampel per menit.
- Pastikan untuk memilih opsi '**Infinite**' dan bukan jumlah putaran tertentu untuk menjalankan pengujian selama durasi yang diinginkan.
- Test plan mencakup perancangan dan otorisasi API, menyiapkan rencana pengujian, dan menambahkan Listener yang diperlukan untuk memverifikasi hasilnya.

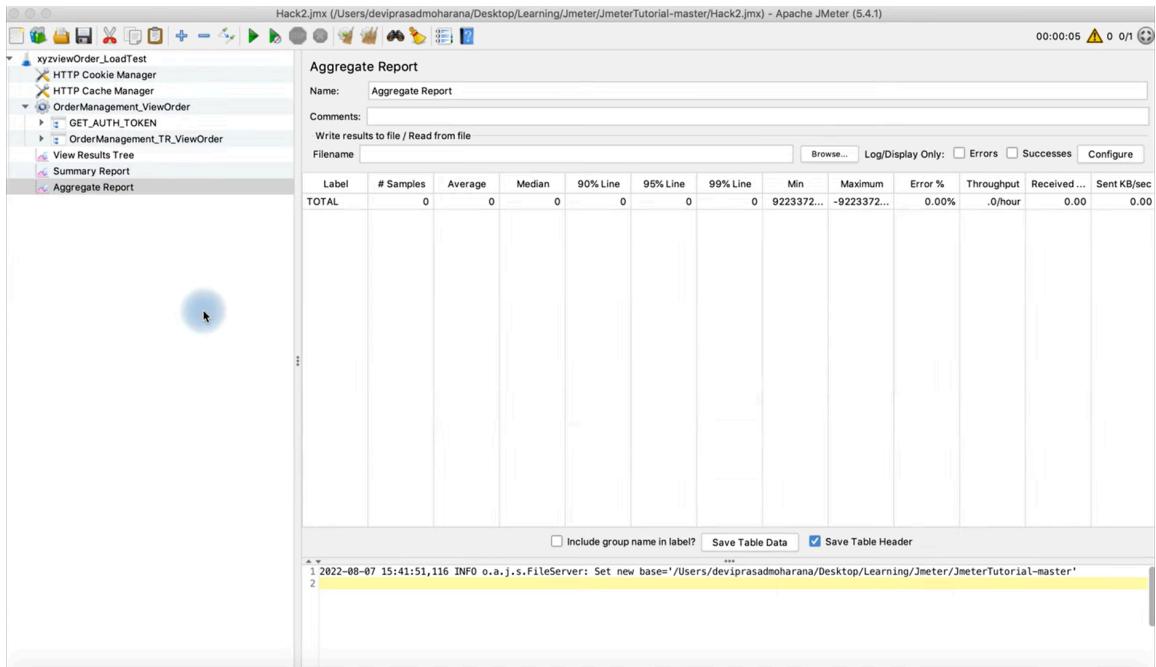




6. Tambahkan beberapa Listener yang diperlukan untuk memverifikasi hasilnya.







Step 5 : Jalankan pengujian untuk load testing

