Plugins for analyzing results

Contents

Installing Plugins

1. Download plugins-manager.jar: https://jmeter-plugins.org/install/Install/
2. Added the downloaded jar file to the folder lib/ext JMeter
3. Ran the JMeter (restarted if it was opened while adding the jar file to the folder lib/ext), opened Options, and select Plugins manager

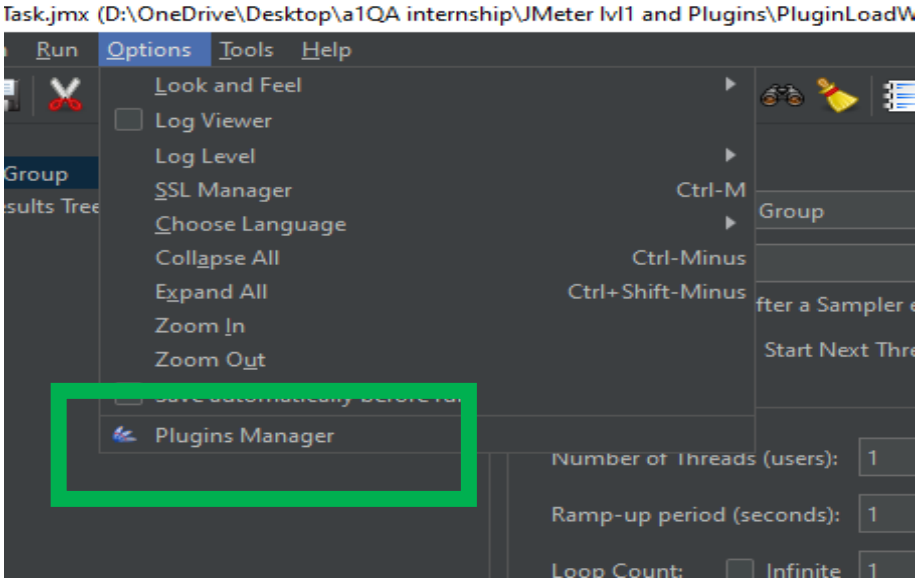
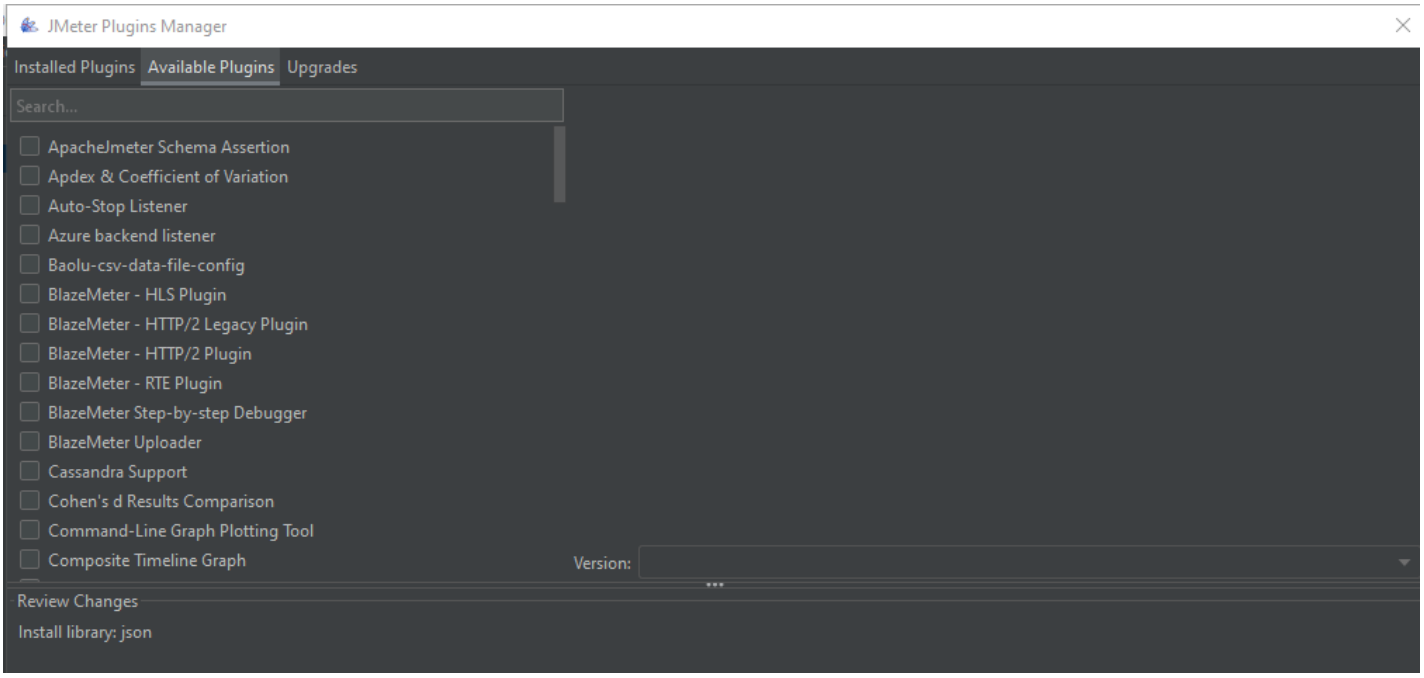
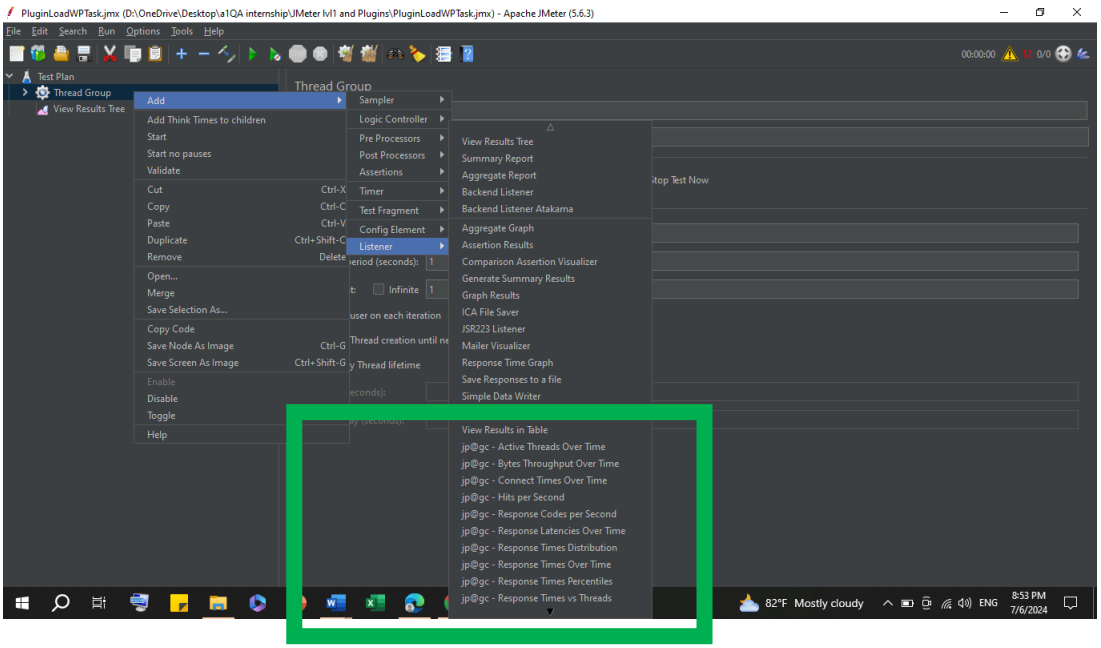


Figure 1

4. Went to the Available Plugins tab and select the plugins you need to install



5. I found the installed plugins inside the test plan elements, and the name indicates jp@gc - …


PluginLoadWPTask.jmx (D:\OneDrive\Desktop\a1QA internship\JMeter lvl1 and Plugins\PluginLoadWPTask.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
Test Plan
Thread Group
View Results Tree
Add
Add Think Times to children
Start
Start no pauses
Validate
Cut
Copy
Paste
Duplicate
Remove
Open...
Merge
Save Selection As...
Copy Code
Save Node As Image
Save Screen As Image
Enable
Disable
Toggle
Help
Sampler
Logic Controller
Pre Processors
Post Processors
Assertions
Timer
Test Fragment
Config Element
Listener
View Results Tree
Summary Report
Aggregate Report
Backend Listener
Backend Listener Atakama
Aggregate Graph
Assertion Results
Comparison Assertion Visualizer
Generate Summary Results
Graph Results
ICA File Saver
JSR223 Listener
Mailer Visualizer
Response Time Graph
Save Responses to a file
Simple Data Writer
View Results in Table
jp@gc - Active Threads Over Time
jp@gc - Bytes Throughput Over Time
jp@gc - Connect Times Over Time
jp@gc - Hits per Second
jp@gc - Response Codes per Second
jp@gc - Response Latencies Over Time
jp@gc - Response Times Distribution
jp@gc - Response Times Over Time
jp@gc - Response Times Percentiles
jp@gc - Response Times vs Threads
82°F Mostly cloudy
ENG
8:53 PM
7/6/2024

JMeter Plugin - 3 Basic Graphs

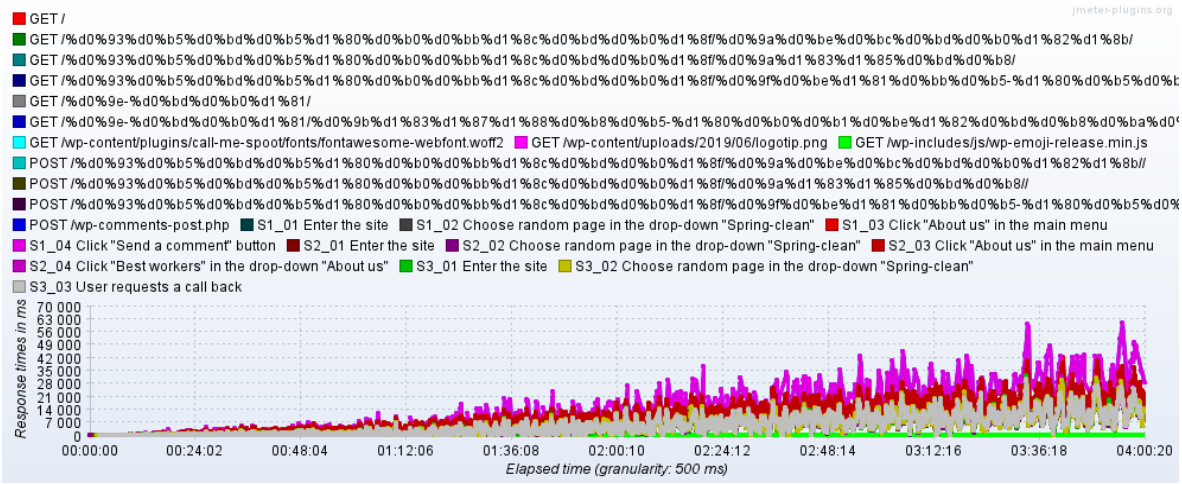
1. Average Response Time

This graph shows the average response time in milliseconds for each sampler. The average response time is measured in milliseconds (ms) for each sampler throughout the test. The samplers are listed along the x-axis of the graph, and the average response time is displayed on the y-axis. The response time for each sampler is represented by a vertical bar on the graph.

For instance, the sampler named "GET /" has an average response time of around 7,000 milliseconds. This means it took an average of 7 seconds to complete this sampler request.

It's seen that the average response time varies depending on the sampler. Some samplers, such as "GET /wp-content/plugins/call-me-spoot/fonts/fontawesome-webfont.woff2" have a much lower average response time than others. This could be because the sampler is requesting a smaller file or because the server can respond to this type of request more quickly.

Overall, the graph can be used to identify which samplers are taking the longest to respond. This information can help troubleshoot performance issues with the application.

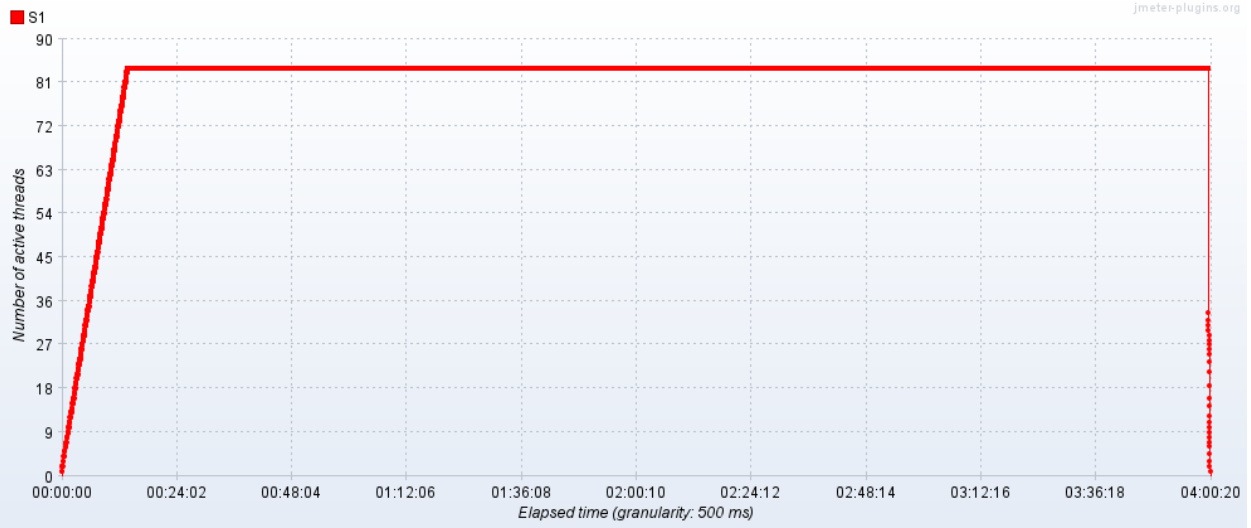


Path: *Test Plan -> Add -> Listener -> jp@gc - Response Times Over Time*

2. Active Threads

Active Threads Over Time is a graph that shows how many concurrent users are active in each thread group during the test.

It shows how many concurrent users were active in each thread group during a test. The X-axis represents the elapsed time of the test, divided into intervals of 500 milliseconds. The Y-axis represents the number of active threads.
In the graph, there is a single thread group, as evidenced by the single line. The number of active threads fluctuates throughout the test, with a peak of 90 users at around the 24-minute mark. It then trails off to zero by the end of the test at the 4-minute mark.
This type of graph can be useful for understanding how many users your application can handle concurrently. It can also help you identify any bottlenecks in the application that may be caused by too many users.



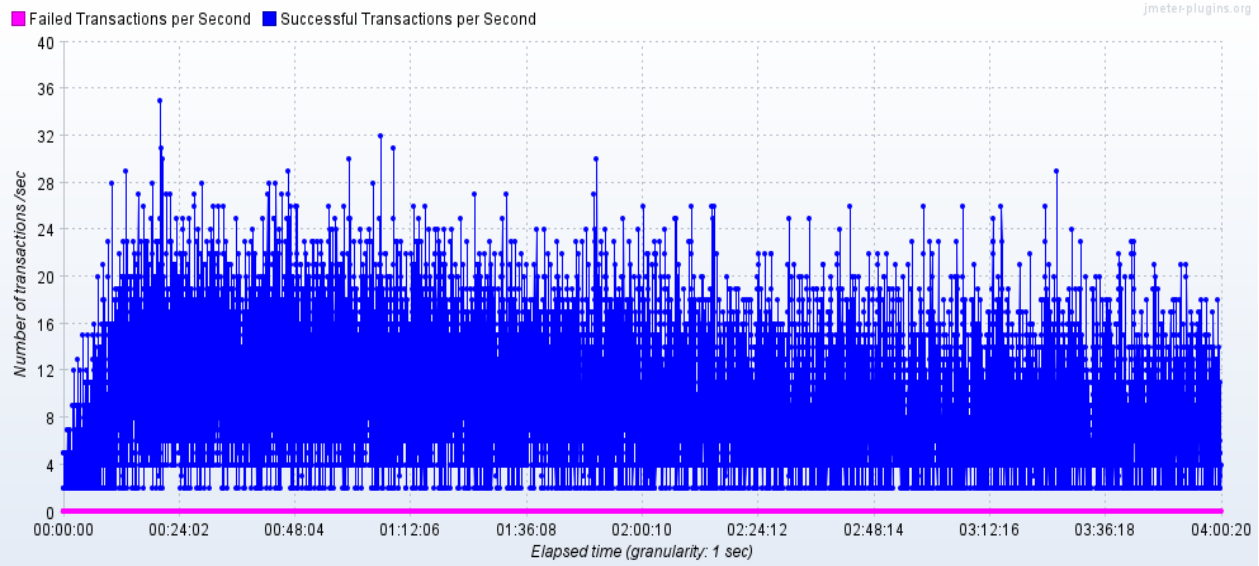
Path: *Test Plan -> Add -> Listener -> jp@gc - Active Threads Over Time*

3. Successful/Failed Transactions (*Transactions per Second*)

This graph shows the number of successful and unsuccessful transactions per second for each sampler during the test.

Each data point on the graph represents the number of successful or failed transactions that occurred during a one-second interval. For example, at the 24-second mark, there were roughly 32 successful transactions and 8 failed transactions per second.

The text at the bottom of the graph indicates timestamps throughout the test.



Path: *Test Plan -> Add -> Listener -> jp@gc – Transactions per Second*

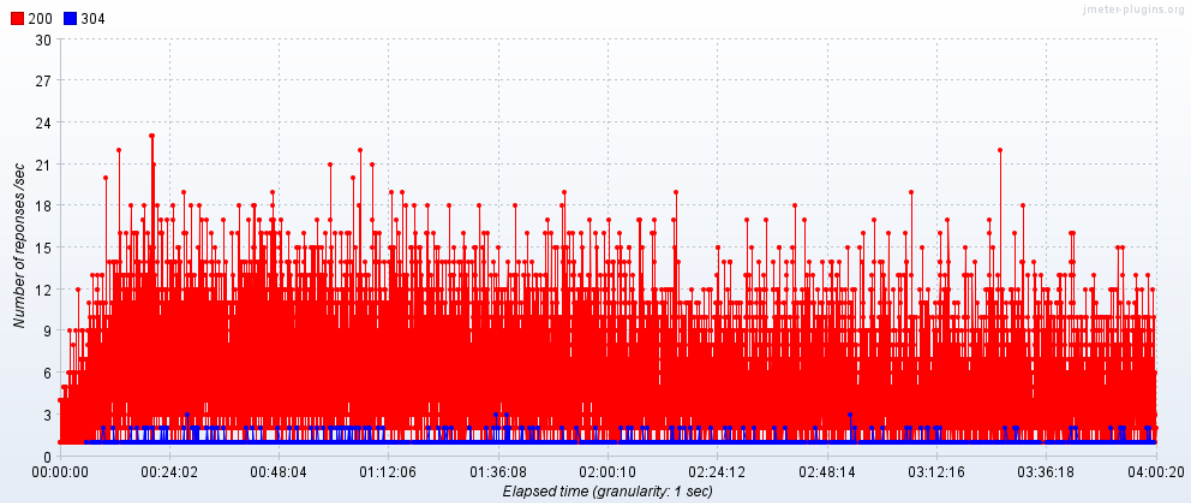
JMeter Plugin - 5 Additional Graphs

1. Response Codes

Plugin Response Codes per Second shows a graph that contains the response code per second during the test.

The graph shows that the test generated mostly successful responses (green line) throughout its duration. There were also instances of 304 (blue line) and 301 (purple line) response codes throughout the test. There were very few errors (red line) detected during the test.

By looking at the graph, can identify which response codes occurred most frequently during the test. This information can help troubleshoot any issues with your application. For instance, if you see a large number of red lines in the graph, it may indicate that there are errors with your application that need to be addressed.



Path: *Test Plan -> Add -> Listener -> jp@gc – Response Codes per Second*

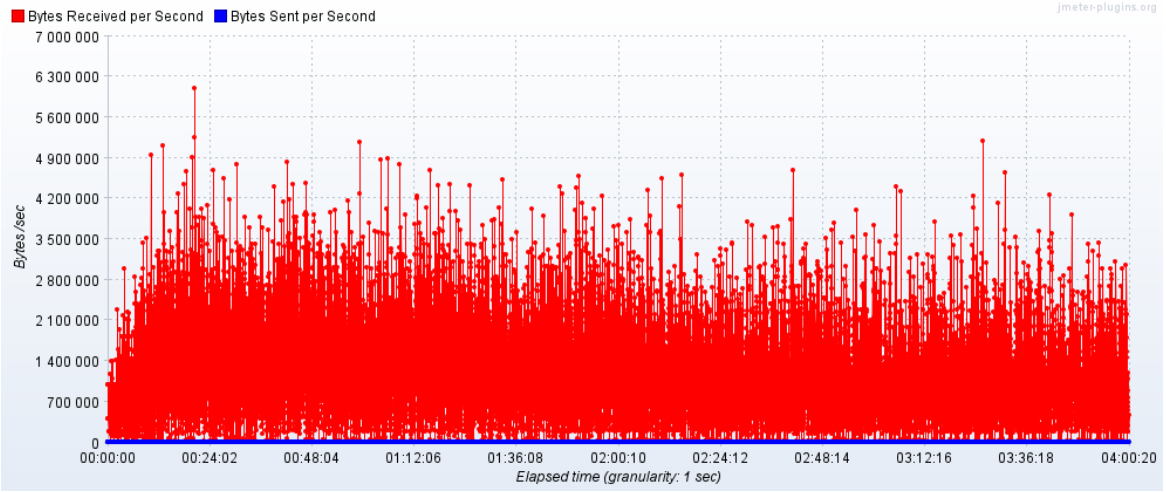
2. Bytes Throughput

It shows the average bytes throughput received and sent per second throughout a load test. The X-axis represents the elapsed time of the test, divided into intervals of one second. The Y-axis represents the number of bytes per second.

Each data point on the graph represents the average number of bytes received or sent during a one-second interval. For example, at the 24-second mark, the test received an average of roughly 5,800,000 bytes per second and sent an average of 2,100,000 bytes per second.

The text at the bottom of the graph indicates timestamps throughout the test.

This type of graph can be useful for understanding how much data your application is transferring during a load test. It can help you identify any bottlenecks in your network or application that may be caused by high data transfer volumes.



Path: *Test Plan -> Add -> Listener -> jp@gc - Bytes Throughput Over Time*

3. Connect Times

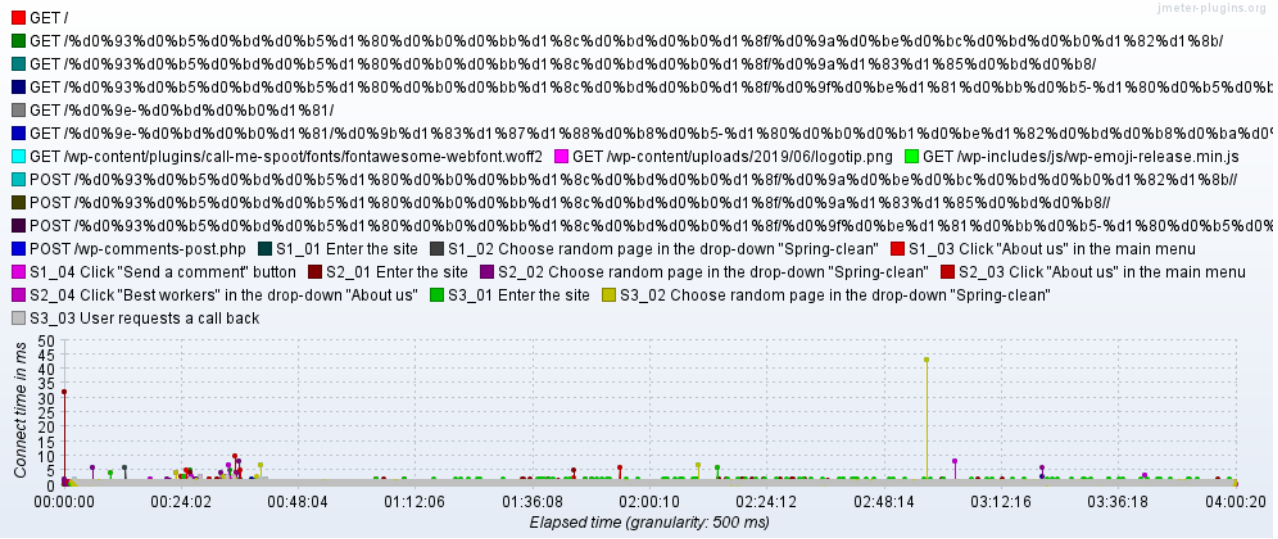
The graph shows the average connection time during the load test.

It shows the connection time in milliseconds (ms) for each sampler throughout the test. The X-axis lists the samplers used in the test, and the Y-axis shows the connect time in milliseconds. The connect time for each sampler is represented by a vertical bar on the graph.

For instance, the sampler named "GET /" has a connect time of around 2 milliseconds. This means it took an average of 2 milliseconds to establish a connection for this sampler request.

The connect time varies depending on the sampler. Some samplers, such as "GET /wp-content/plugins/call-me-spoot/fonts/fontawesome-webfont.woff2" have a much lower connect time than others. This could be because the server where the resource is located is geographically closer or because the connection is already established and cached by the browser.

Overall, the graph can be used to identify which samplers are taking the longest to connect. This information can help troubleshoot performance issues with your application, such as slow network connections or overloaded servers.



Path: *Test Plan -> Add -> Listener -> jp@gc – Connect Times Over Time*

4. Latency

The graph shows the delays during the load test.

It shows the response time in milliseconds (ms) for each sampler throughout a load test. The X-axis lists the samplers used in the test, and the Y-axis shows the response time in milliseconds. The response time for each sampler is represented by a vertical bar on the graph.

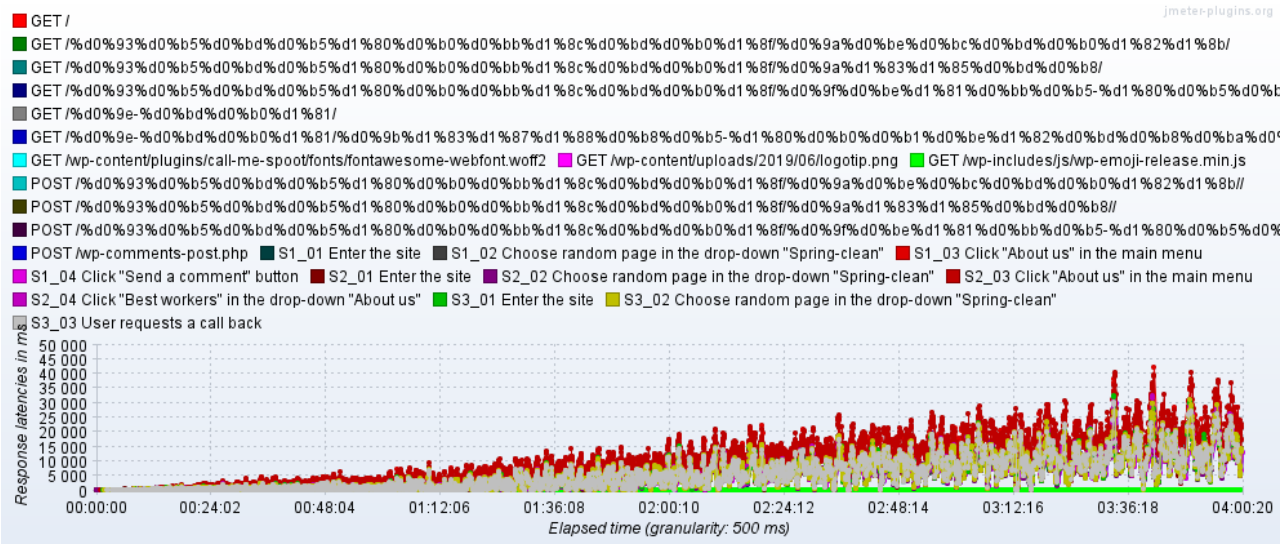
Here are some observations about the response latencies in the graph:

- The response time varies significantly between different samplers.
    - Some samplers, like "GET /wp-content/plugins/call-me-spoot/fonts/fontawesome-webfont.woff2" have a low response time (around 20 ms), while others, like "POST /wp-comments-post.php" have a much higher response time (around 400 ms).
- There seems to be a pattern where requests to external resources (like fonts or images) have a lower response time than requests that require processing on the server (like posting a comment).

This kind of graph can be useful for identifying performance bottlenecks in web applications. By looking at the samplers with the highest response times, can focus efforts on optimizing those specific parts of your application.

Here are some additional things to keep in mind when interpreting this graph:

- The response time can be affected by a number of factors, including the network, the server, and the application itself.
- The scale of the Y-axis may vary depending on the specific test results.
- It is important to compare the response times of similar samplers to identify outliers.

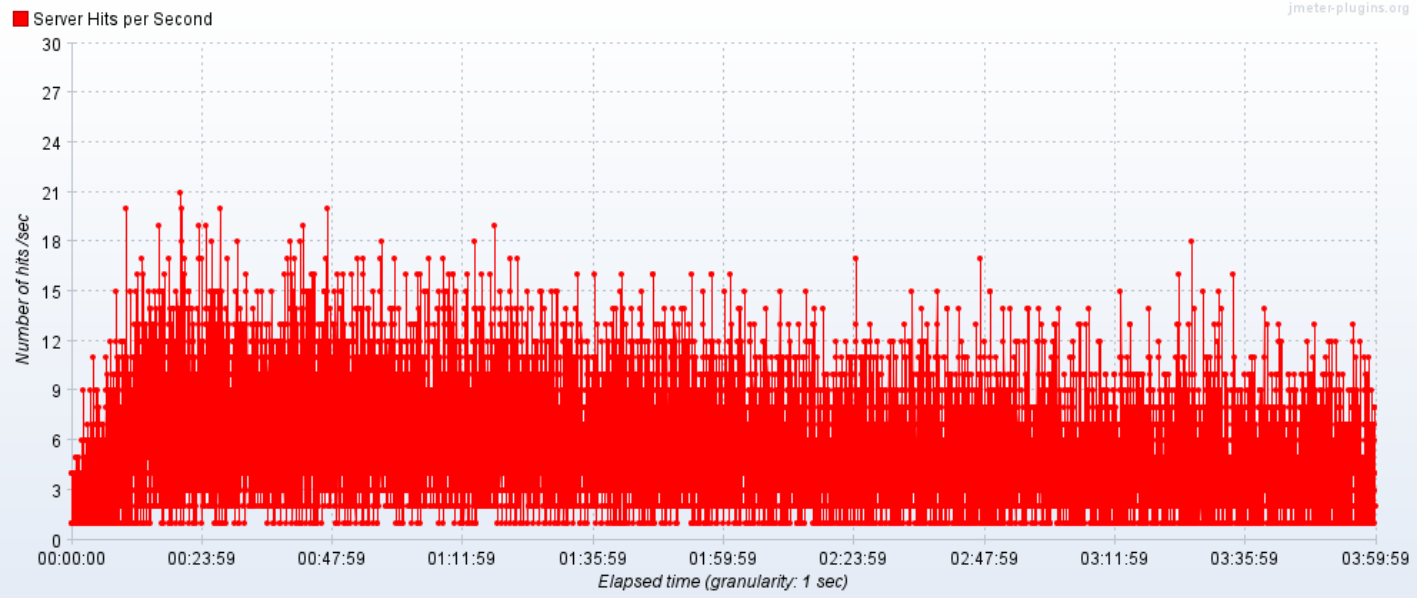


Path: *Test Plan -> Add -> Listener -> jp@gc – Response Latenncies Over Time*

5. Hits/s

The graph shows how many of the requests are sent to the server during the test.

- The number of hits per second can be affected by a number of factors, including the number of virtual users, the complexity of the requests, and the network conditions.
- The scale of the Y-axis may vary depending on the specific test results.
- It is important to consider the hits per second in conjunction with other metrics, such as response time and throughput, to get a complete picture of the application's performance.



Path: *Test Plan -> Add -> Listener -> jp@gc – Hits per Second*

Additional Graphs

1. Response Time vs Threads

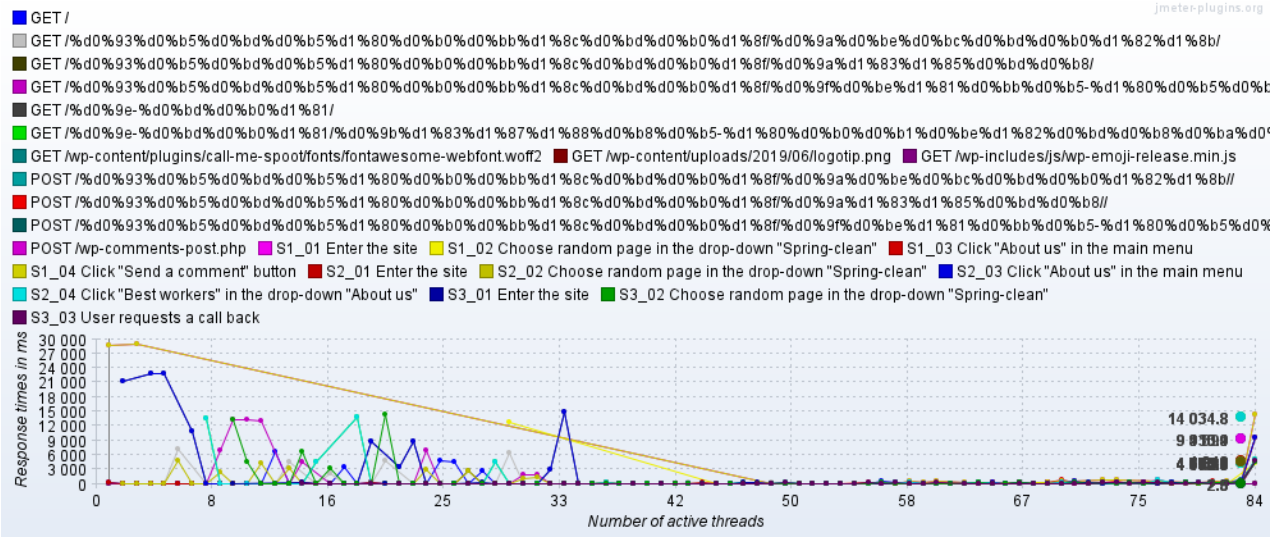
The graph shows how the response time varies depending on the number of parallel threads. The server needs more time to respond when many users request it at the same time. This graph visualizes such dependencies. which is commonly used to analyze performance under load. Here's a breakdown of the graph:

- **X-axis (Number of Active Threads):** This represents the number of concurrent virtual users simulating requests on the server during the test. The number increases gradually from 0 to around 75.
- **Y-axis (Response Time in ms):** This represents the average time it takes for the server to respond to a request, measured in milliseconds (ms).

The vertical bars on the graph represent the average response time for each set of virtual users (threads). As the number of threads increases (more concurrent users), the response time also increases. This is because the server has more requests to handle, and it takes longer to respond to each request.

Here are some observations from the graph:

- There seems to be a relatively stable increase in response time as the number of threads goes up. This could indicate that the server is able to handle the increased load without significant performance degradation.
- It's difficult to say definitively from this graph where the performance bottleneck might be. Ideally, you would want to see a flat line where the response time remains steady as you add more users.

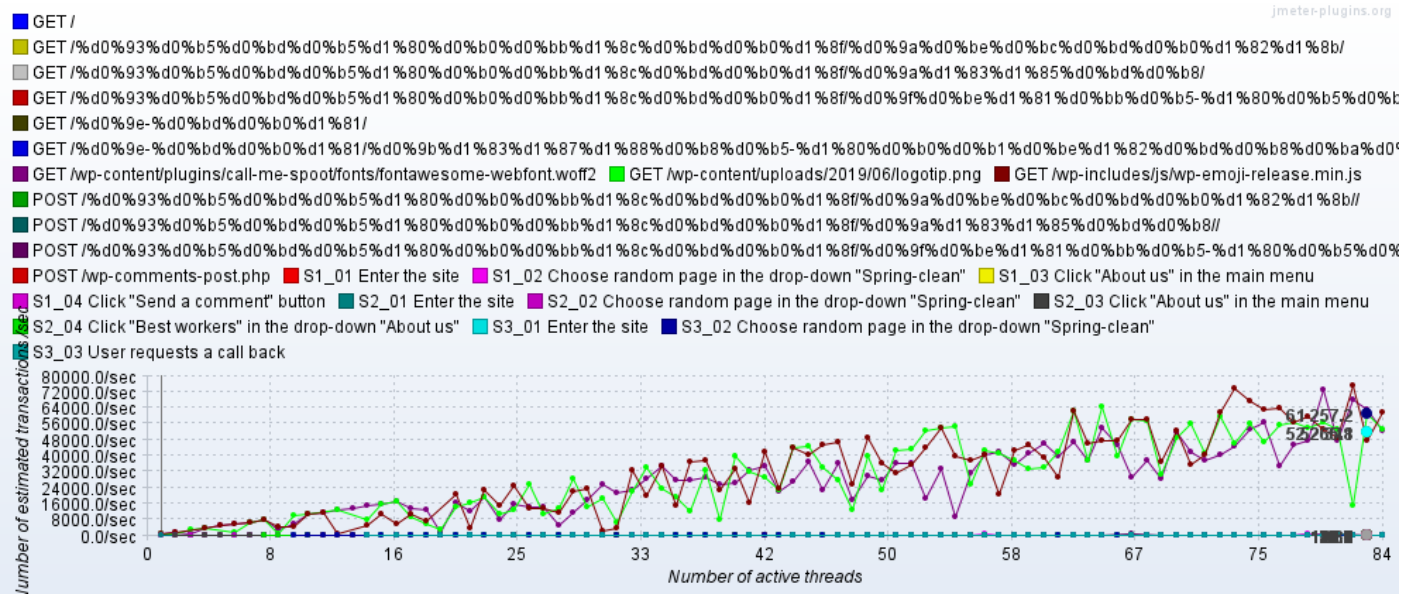


Path: *Test Plan -> Add -> Listener -> jp@gc – Response Time vs Threads*

2. Transaction Throughput vs Threads

Listener Response Times vs Threads. However, it also shows the total transaction throughput. There is the formula for the total throughput: <active virtual users> * 1

second / <1 thread response time>. The graph helps visualize how the application's transaction throughput scales as more concurrent users are introduced. In this instance, it seems the server can efficiently handle up to 50 concurrent users before showing signs of performance limitations.



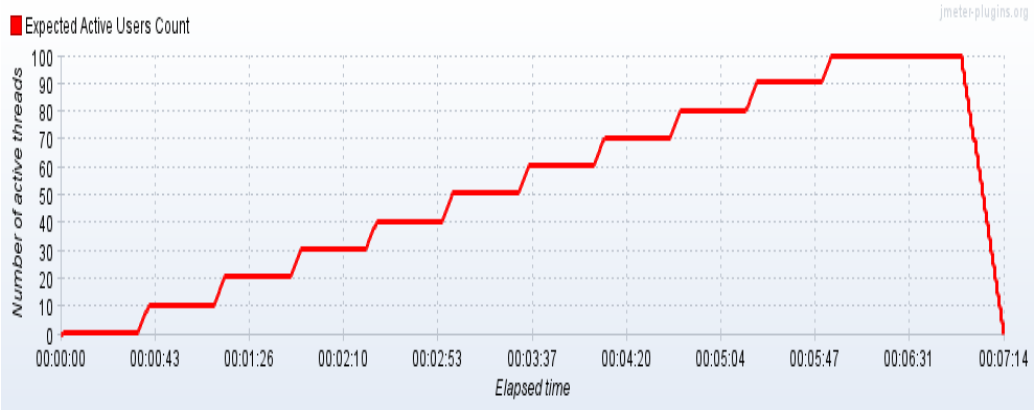
Path: *Test Plan -> Add -> Listener -> jp@gc* – Transaction Throughput vs Threads

3. Stepping Thread Group

Installation: Tools→ Option→Plugin manager→Available Plugins→ Custom Thread Groups

Path: *Test Plan -> Add -> Thread (Users)-> jp@gc – Stepping Thread Group*

The graph in the window provides a visual representation of the planned thread schedule. The X-axis represents time, and the Y-axis represents the number of active threads. In the example you sent, the graph shows a scheduled thread ramp-up over 60 seconds, with 10 users added every 10 seconds, until reaching a total of 100 users. The test will then hold for 60 seconds at 100 users before the thread ramp-down begins.

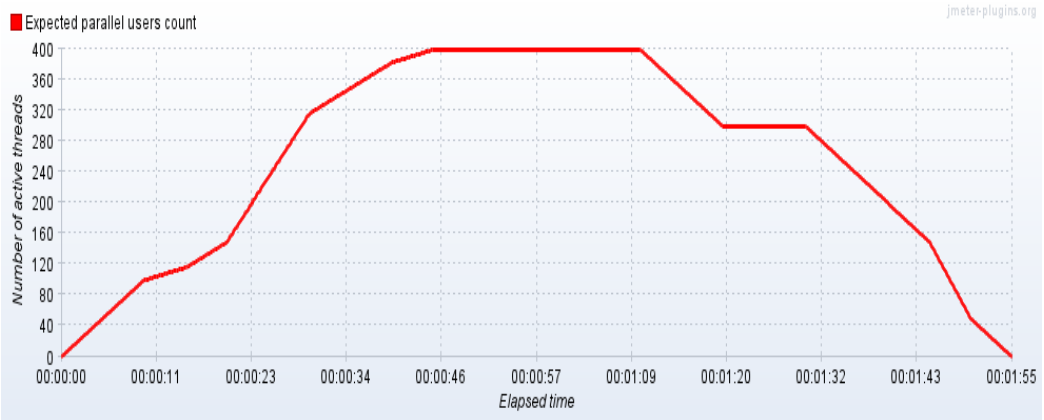


4. Ultimate Thread Group

Installation: Tools→ Option→Plugin manager→Available Plugins→ Custom Thread Groups

Path: *Test Plan -> Add -> Thread (Users)-> jp@gc – Ultimate Thread Group*

The Ultimate Thread Group can be useful for simulating more complex load scenarios on an application. By defining multiple steps with different thread counts and timings, you can create a test that more closely reflects real-world usage patterns. This can help you to identify bottlenecks and performance issues that might not be apparent with a simpler thread ramp-up configuration.



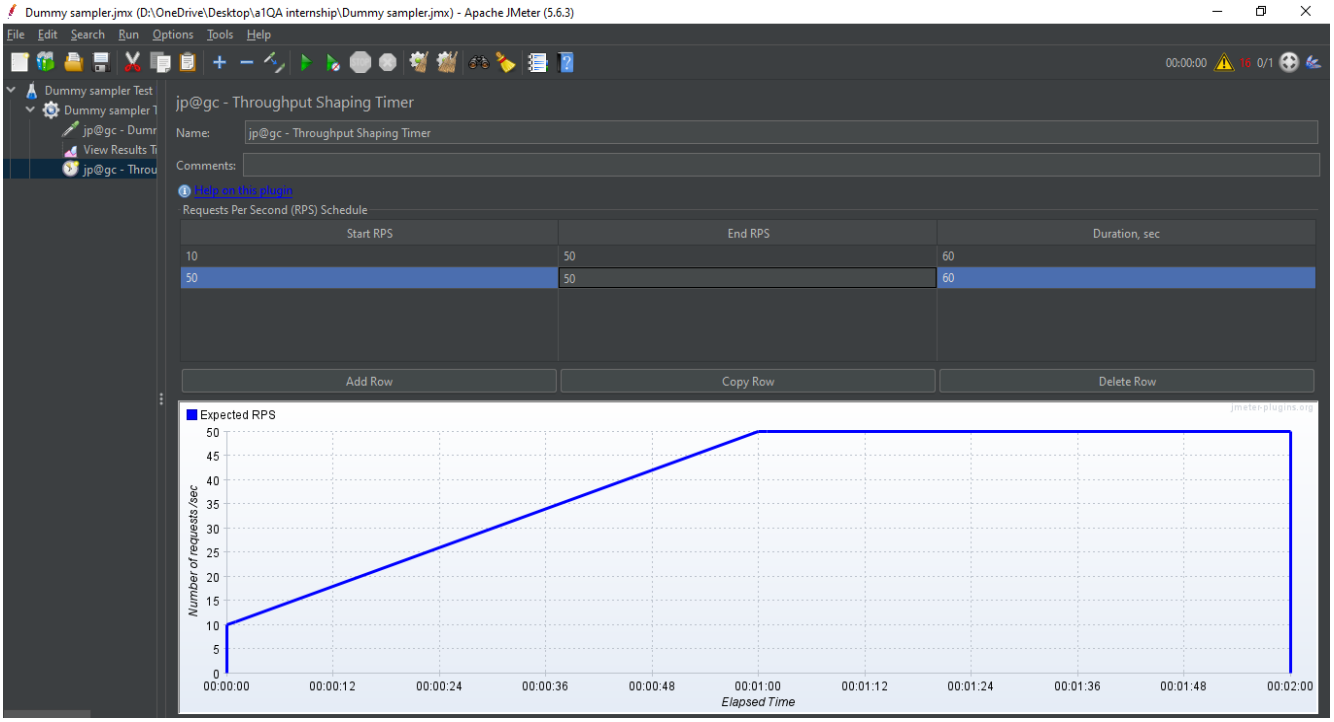
5. Throughput Shaping Timer

Installation: Tools→ Option→Plugin manager→Available Plugins→ Throughput Shaping Timer

Path: *Test Plan -> Add -> Timer-> jp@gc – Throughput Shaping Timer*

The plugin delays requests automatically to reach the target RPS load level. RPS (number of requests per second) = total amount of requests/duration in seconds.
The graph in the window provides a visual representation of the planned RPS schedule. The X-axis represents time (seconds), and the Y-axis represents the desired

RPS. The different horizontal lines in the graph represent the different steps defined in the RPS schedule table.



6. Dummy Sampler

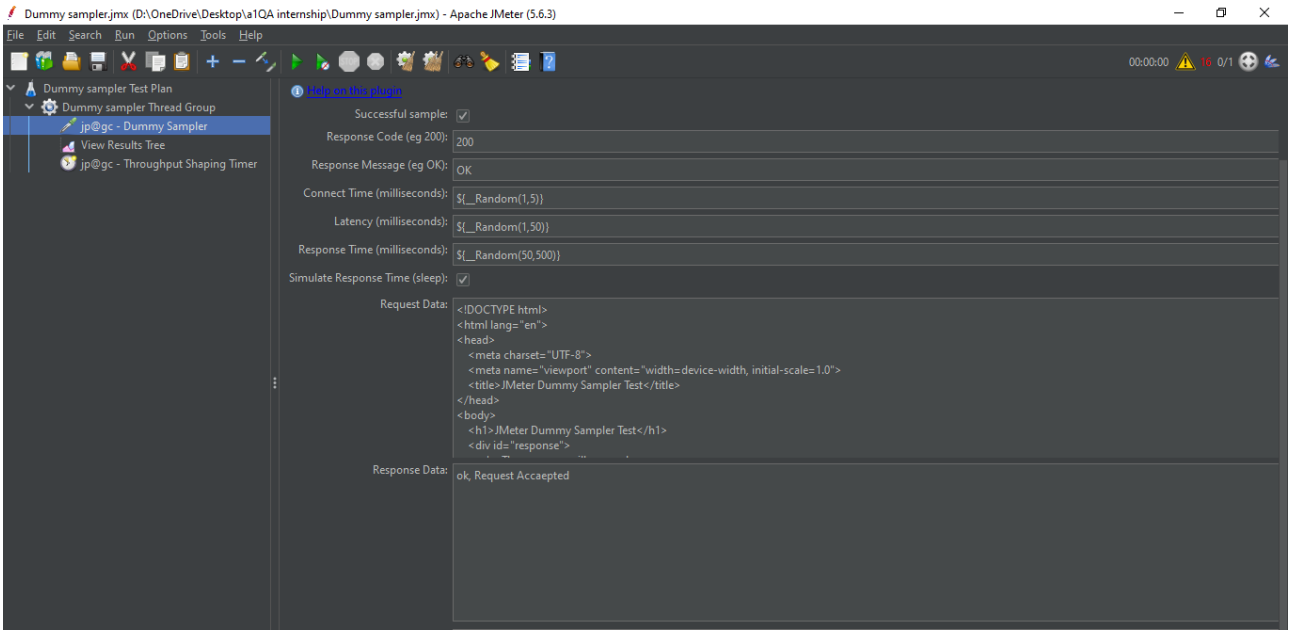
Installation: Tools→ Option→Plugin manager→Available Plugins→ Dummy Sampler

Path: Test Plan ->Thread group-> Add -> Sampler-> jp@gc– Dummy Sampler

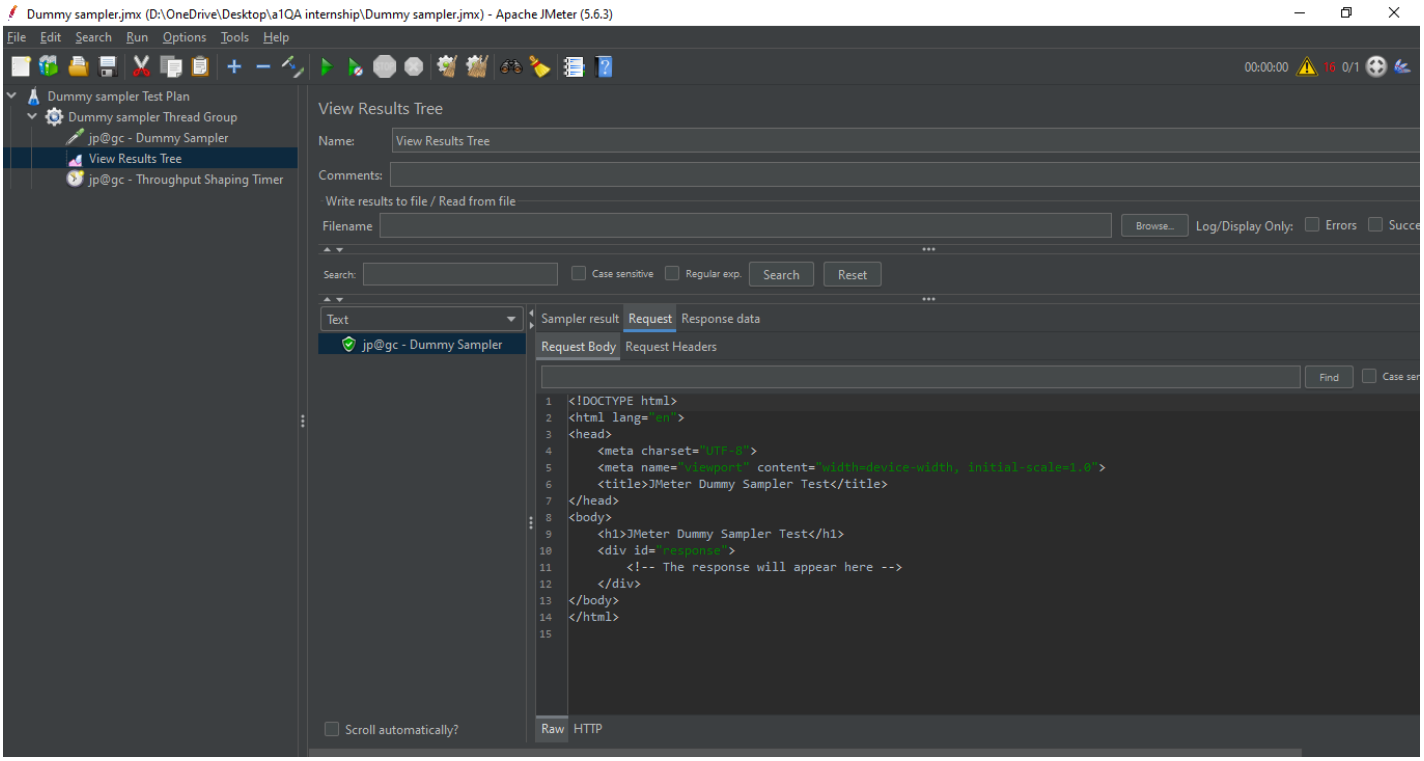
Allows to debug a test without running it. Allows to set the necessary data without creating a real load. Easy to use when debugging RegExp Extractor, BeanShell Post-Processor.

Testing purpose:

1. **Added a Dummy Sampler:** I included a Dummy Sampler in my test plan.
2. **Configured the Request (HTML):** In the Dummy Sampler settings, I pasted the HTML code I wanted to use as the simulated request data.
3. **Set the Response:** I defined the response data as "accepted" to indicate successful processing.
4. **Added a Listener:** I attached the Listener→ View result tree, I wanted to test the Dummy Sampler.
5. **Ran the Test:** When I executed the JMeter test, the Dummy Sampler provided the simulated response (HTML request and "accepted" response) to my listener.

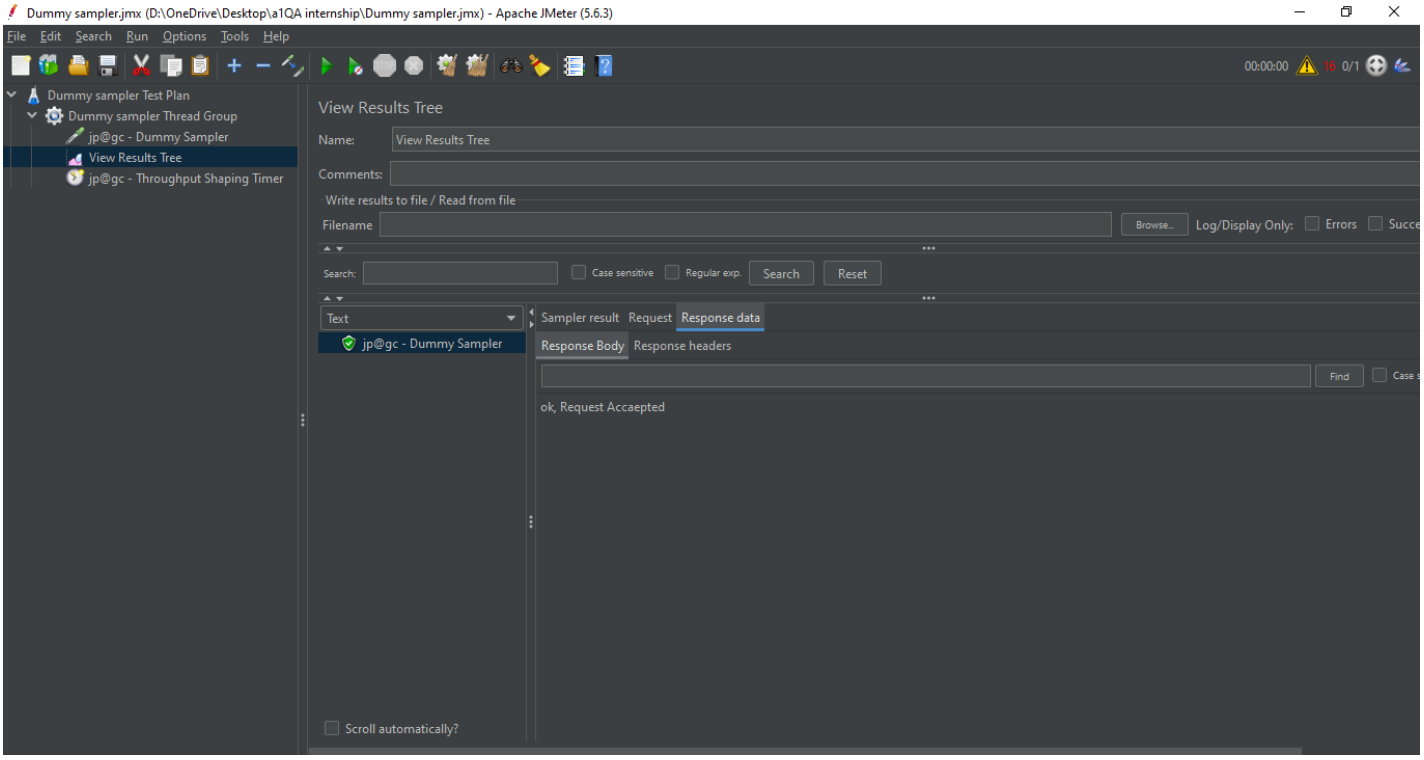

Dummy sampler.jmx (D:\OneDrive\Desktop\a1QA internship\Dummy sampler.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:00:00 16 0/1
Dummy sampler Test Plan
Dummy sampler Thread Group
jp@gc - Dummy Sampler
View Results Tree
jp@gc - Throughput Shaping Timer
Help on this plugin
Successful sample:
Response Code (eg 200):
200
Response Message (eg OK):
OK
Connect Time (milliseconds):
${__Random(1,5)}
Latency (milliseconds):
${__Random(1,50)}
Response Time (milliseconds):
${__Random(50,500)}
Simulate Response Time (sleep):
Request Data:
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JMeter Dummy Sampler Test</title>
</head>
<body>
<h1>JMeter Dummy Sampler Test</h1>

Response Data:
ok, Request Accaepted


Dummy sampler.jmx (D:\OneDrive\Desktop\a1QA internship\Dummy sampler.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
Dummy sampler Test Plan
Dummy sampler Thread Group
jp@gc - Dummy Sampler
View Results Tree
jp@gc - Throughput Shaping Timer
View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Search:
Case sensitive
Regular exp.
Search
Reset
Text
jp@gc - Dummy Sampler
Sampler result
Request
Response data
Request Body
Request Headers
Find
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JMeter Dummy Sampler Test</title>
</head>
<body>
<h1>JMeter Dummy Sampler Test</h1>

<!-- The response will appear here -->

</body>
</html>
Scroll automatically?
Raw
HTTP


Dummy sampler.jmx (D:\OneDrive\Desktop\a1QA internship\Dummy sampler.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
Dummy sampler Test Plan
Dummy sampler Thread Group
jp@gc - Dummy Sampler
View Results Tree
jp@gc - Throughput Shaping Timer
View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Search:
Case sensitive
Regular exp.
Search
Reset
Text
jp@gc - Dummy Sampler
Sampler result
Request
Response data
Response Body
Response headers
Find
ok, Request Accaepted
Scroll automatically?