

|  |
| --- |
|  |

**Toolname Guidelines and Best Practices**

# Table of Contents

1. INTRODUCTION…………………………………………………………………………………………………………………………….

2. BEST PRACTICES…………………………………………………………………………………………………………………………….

# 

# Introduction

This document show best practices of JMETER. JMETER has some limitations especially when it is run in a distributed environment. Following BEST PRACTICES guidelines will assist in creating a real and continuous load

# BEST PRACTICES

Following are the guidelines for developers:

1. **Do not use the Console to create your load :**

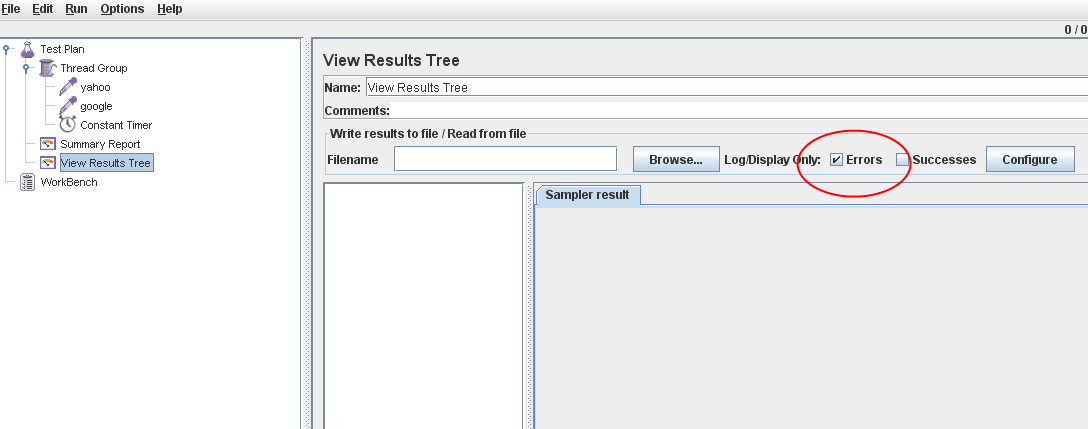
You can use the console for debugging purposes or to run a small load from it to make sure the script is running correctly. The GUI consumes a lot of memory under heavy load , therefore the console server by itself cannot sustain a heavy load

1. **Limit the number of threads per engine to 300 :**

This means the total number of threads generated by your test plan should be less than 300. The overall number of threads running from Blaze Meter will be the total number of threads multiplied by the number of JMeter Engines. For example, a test plan with 200 threads and 4 JMeter Engines, will generate a load of 800 threads.

1. **Disable the “View Result Tree” listener :**

as it consumes a lot of memory and can result in the console freezing or JMeter running out of memory. It is, however, safe to use the “View Result Tree” listener with only “Errors” checked.



1. **Disable all JMeter graphs :**

as they consume a lot of memory. You can view all of the real time graphs using the **JTLs** tab in your web interface.

1. **Monitor the logs :**

Any errors in the test plan or the test itself will appear in the log files that are available online under the **Logs** tab. For example: Out of Memory, disconnections etc

1. **Reducing resource requirement :**

Some suggestions on reducing resource usage.

* Use non-GUI mode: jmeter -n -t test.jmx -l test.jtl
* Use as few Listeners as possible; if using the -l flag as above they can all be deleted or disabled.
* Don't use "View Results Tree" or "View Results in Table" listeners during the load test, use them only during scripting phase to debug your scripts.
* Rather than using lots of similar samplers, use the same sampler in a loop, and use variables (CSV Data Set) to vary the sample. [The Include Controller does not help here, as it adds all the test elements in the file to the test plan.]
* Don't use functional mode
* Use CSV output rather than XML
* Only save the data that you need
* Use as few Assertions as possible
* Use the most performing scripting language (example : JSR223 we recommended using  JSR 223 + Groovy for Scripting. It’s the best option as Groovy scripts can be compiled into native Java code (assuming some requirements are met) so Groovy script execution performance can be almost as fast as Java code. )
* If your test needs large amounts of data - particularly if it needs to be randomised - create the test data in a file that can be read with CSV Dataset. This avoids wasting resources at run-time.

1. **Managing Properties**

When you need to modify jmeter properties, ensure you don't modify jmeter.properties file, **instead copy the property from jmeter.properties and modify its value in user.properties file**.  
Doing so will ease you migration to the next version of JMeter.

1. **Use a proxy server**

The Proxy server benefits you to take out definite common elements from the recorded samples. Furthermore, it is beneficial features to record your testing

1. File names should include **only alphanumeric** characters, underscores or hyphens:

[0-9],[aA-zZ],[\_-]. File names should always **include an extension.**

1. **Clean** the Files tab prior to every test run.

**Best Practices for JTL file**

* 1. **Assign a meaningful JTL file name**

For example: 150210-4000t-1.jtl. Do not use spaces in the JTL file name.

* 1. Make sure that the JTL**file is saved in**XML**format** and not CSV.
  2. **Visit the JTLs tab at least once** to generate the real-time reports.
  3. **Use short and meaningful names** for labels identifying controllers and samplers (for example HTTP requests). If you use long names, the names themselves will take the better part of the graphs generated using the **JTLs** tab.