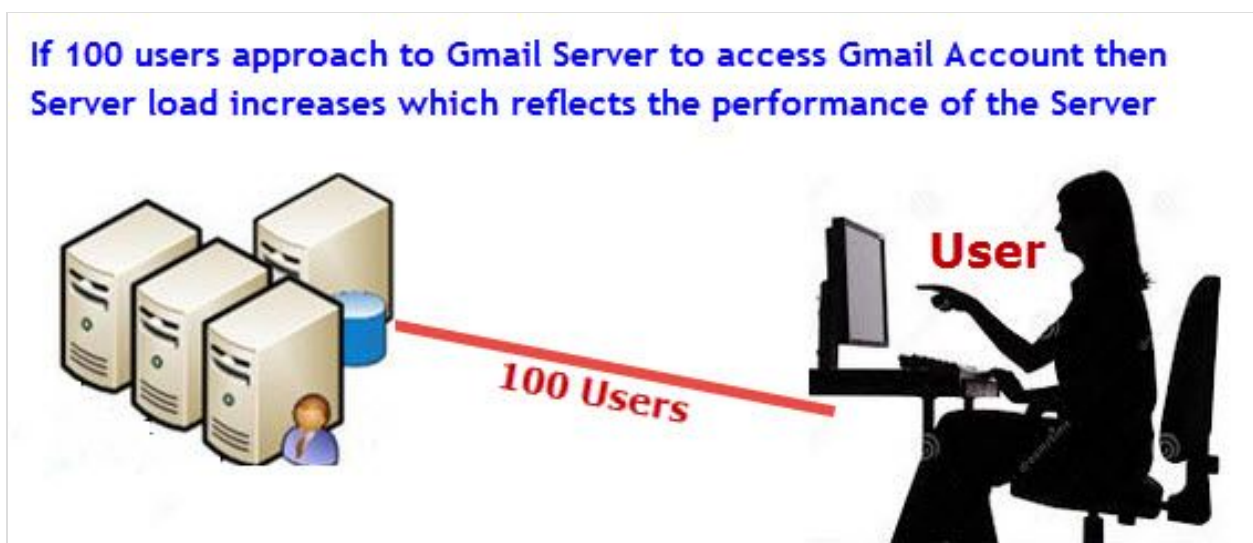


Introduction to Apache JMeter

Apache JMeter is a great open source application with awesome testing abilities. Web Server is a platform which carries loads of numbers of applications and users, so that it is necessary to know that how does it works or performs means; how effective it is to handle simultaneous users or applications.

For example; how the "Gmail" supporting server will perform when numbers of users simultaneous access the Gmail account – basically have to do performance testing using performance testing tools like JMeter, Loadrunner etc.

To check the high performance of the application or server, do high performance testing using JMeter for exceptional results.



Before understanding [Overview of JMeter](#), let us have a look on three testing approach,

Performance Test: This test provides the best possible performance of the system or application under a given configuration of infrastructure. Very fast, it also highlights the change need to be made before application goes into production.

Load Test: This test is useful for determining and preceding the system under the top load it was aimed to work under.

Stress Test: This test tries to break the system by crushing its resources.

Introduction of JMeter:



Apache JMeter; this open source application is a Java based application with a graphical interface, designed to analyse and measure the performance and load functional behaviour of web application or variety of services. Firstly, it was developed by Stefano Mazzocchi of the Apache Software Foundation.

Performance testing means testing a web application\server or any other services against heavy load, several and simultaneous user traffic. JMeter is basically used for testing Web Application or FTP application but currently, it is applicable in functional testing, JDBC database connections, FTP, LDAP, Webservices, JMS, HTTP, generic TCP connections and OS Native processes. JMeter can also be organized as a monitor, although this is naturally measured an ad-hoc solution in lieu of advanced monitoring solutions. One of the good things about this is - it runs on any environment / workstation accepting a Java virtual machine, for example: Windows, Linux, Mac, etc

JMeter design is based on plugins. Most of its "out of the box" features are applied with plugins. Off-site developers can simply spread JMeter with custom plugins.

JMeter works on these many protocols: Web Services – SOAP / XML-RPC, Web – HTTP, HTTPS sites 'web 1.0' web 2.0 (ajax, flex and flex-ws-amf), Database – JDBC drivers, Directory – LDAP, Messaging Oriented service – JMS, Service – POP3, IMAP, SMTP, FTP.

Features Supported by JMeter:

People really think of that why they should go for JMeter? So, to clear their drought, let us get to know JMeter awesome features,

Open source application: JMeter is a free open source application which facilitates users or developers to use source code for other development or modification purpose.

User – friendly GUI: JMeter comes with natural GUI. It is very simple and easy to use and users get familiar very soon with it.

Platform independent: Although, it is totally a Java based desktop application. That's why; it can run on any platform. It is highly extensible and capable to load the performance test in many different server types: Web – HTTP, HTTPS, SOAP, Database via JDBC, LDAP, JMS, Mail – POP3.

Write your own test: Using plugins, write your own test case to extend the testing process. JMeter uses text editor to create a test plan and supplies in XML format.

Support various testing approach: JMeter supports various testing approach such as Load Testing, Distributed Testing, and Functional Testing, etc.

Simulation: JMeter can simulate various users with simultaneous threads, generate heavy load against web application under test.

Support multi-protocol: JMeter works on web application testing and database server testing, and also supports protocols such as HTTP, JDBC, LDAP, SOAP, JMS, and FTP.

Script Test: JMeter is capable to perform automation testing using Bean Shell & Selenium.

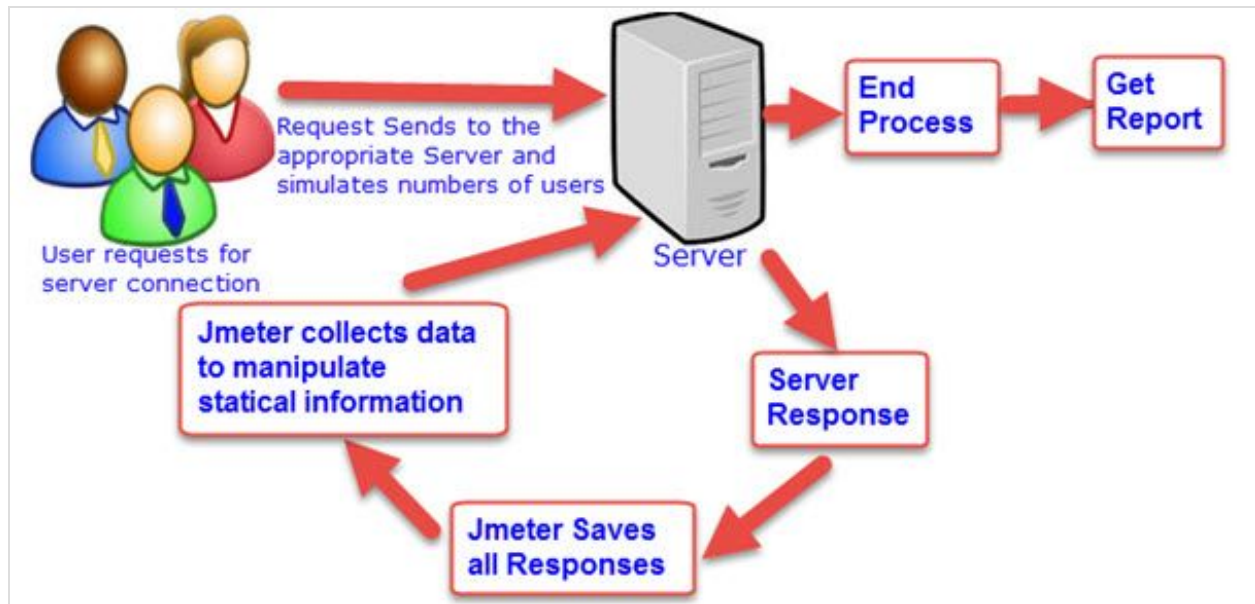
Totally multi-threading framework: It's a full multi-threading framework permits simultaneous sampling by many threads and simultaneous sampling of different functions by distinct thread groups.

Easily understandable test result: It visualizes the test result in a format such as chart, table, tree and log file are very simple to understand.

Easy installation step: Just run the "*.bat" file to use JMeter. In Linux/Unix – JMeter can be approached by clicking on JMeter shell script. In Windows – JMeter can be approached by starting the JMeter.bat file.

JMeter architecture working process:

JMeter simulates number of users send request to an appropriate server which shows the performance/functionality of an appropriate server / application via tables, graphs etc. The figure below depicts this process:



Conclusion:

In JMeter training tutorial first article we have seen about *introduction of JMeter* & what all features supported by JMeter.

JMeter Installation Guide

In previous article we have learned about basic "[Introduction to Apache JMeter](#)" & what all features to be supported in Apache JMeter. In today's article, we will get to know about how to download and install JMeter on local Windows Machine.

JMeter is a pure Java desktop application, that's why it runs on any system which supports Java. So before installing JMeter, first get to know the basic requirement of JMeter to run on local windows machine.

JMeter requirements according to the Operating System

Before installing JMeter, your system should fulfill these requirements,

Operating system	Window XP	Window 7	Free BSD 9.0	Linux 2.4, 2.6, 3.1	Mac OS
Java Virtual Machine	Sun JDK6, Sun JDK 7	Sun JDK 5, 6, 7.	Open JDK 6	Sun JDK 5, 6, 7 and Open JDK 6	JDK6
Architecture	32/64 bits	32 bits	amd64	i386, amd64	

Check for Java on Windows

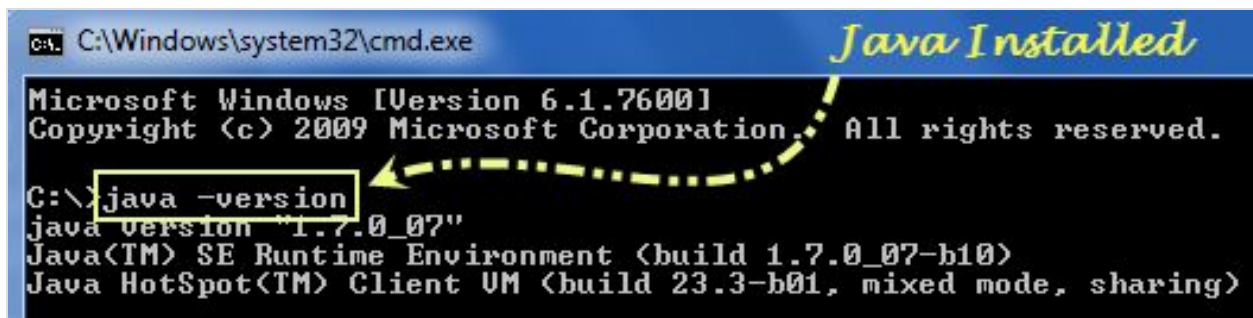
In this tutorial you will get to know how to download and install JMeter on Window operating. These are the steps involved in downloading and installing JMeter,

Before installing JMeter, just check your system is supporting Java or not, if not then install it because JMeter is a pure Java desktop application. You can verify that java is already installed in your windows machine or not using bellow given command in your command prompt. You can use the following procedure to check whether Java JDK is installed successfully in your system.

In Window/Linux – Just go to Terminal or click on start and type “command prompt”

Open the “command prompt” and type command “**java -version**”.

If Java run-time environment is installed successfully, you will see the output as figure below,

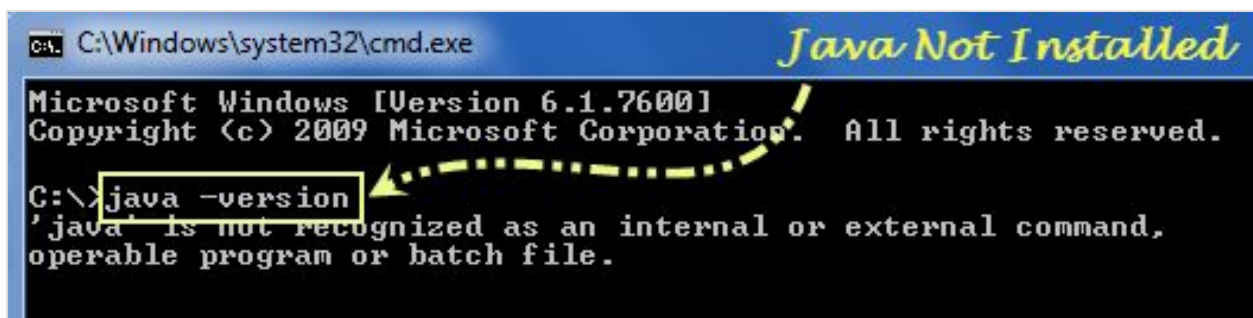


The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a blue header bar with the text "Java Installed" in yellow. The main area is black with white text. It displays the Microsoft Windows version (6.1.7600) and copyright information. The command "java -version" has been entered and executed, resulting in the output: "java version '1.7.0_07'", "Java(TM) SE Runtime Environment (build 1.7.0_07-b10)", and "Java HotSpot(TM) Client VM (build 23.3-b01, mixed mode, sharing)". A yellow dashed arrow points from the "Java Installed" text to the command prompt window.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\>java -version
java version "1.7.0_07"
Java(TM) SE Runtime Environment (build 1.7.0_07-b10)
Java HotSpot(TM) Client VM (build 23.3-b01, mixed mode, sharing)
```

If Java run-time environment is not installed successfully, you will see the output as figure below,



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a blue header bar with the text "Java Not Installed" in yellow. The main area is black with white text. It displays the Microsoft Windows version (6.1.7600) and copyright information. The command "java -version" has been entered and executed, resulting in the error message: "'java' is not recognized as an internal or external command, operable program or batch file." A yellow dashed arrow points from the "Java Not Installed" text to the command prompt window.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\>java -version
'java' is not recognized as an internal or external command,
operable program or batch file.
```

You can get the latest version of Java SE Development Kit in the site:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>, using this site download and install the latest version of Java SE Development Kit. After installation, just

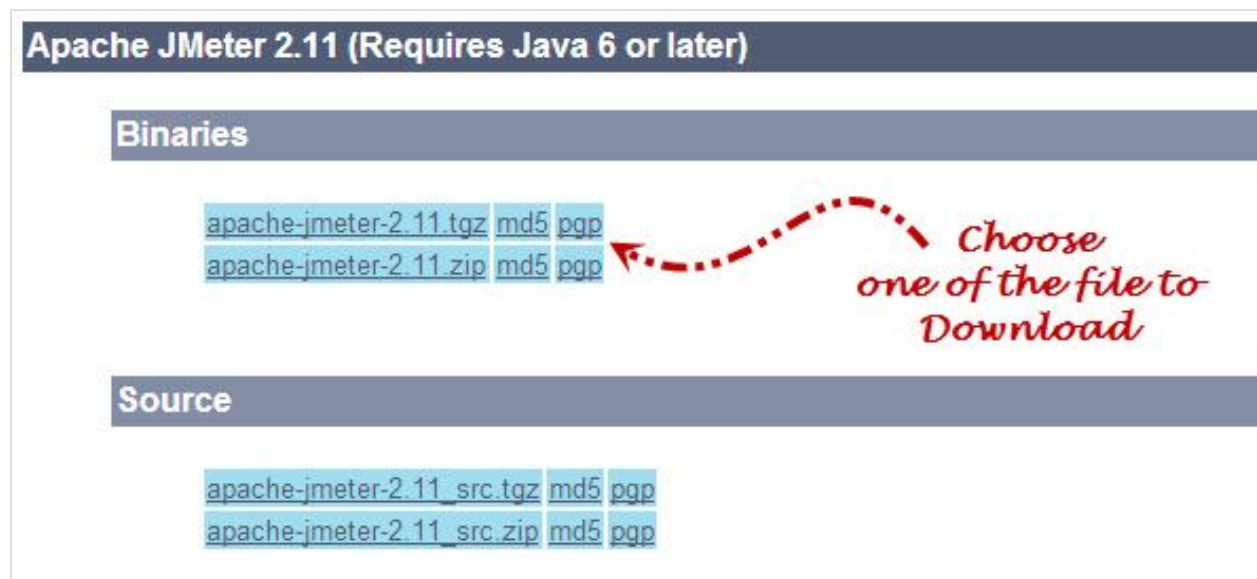
check java is installed successfully or not using command "**Java – Version**" in the command prompt.

Download and installation process of JMeter

Step 1: Download process of v2.11 JMeter

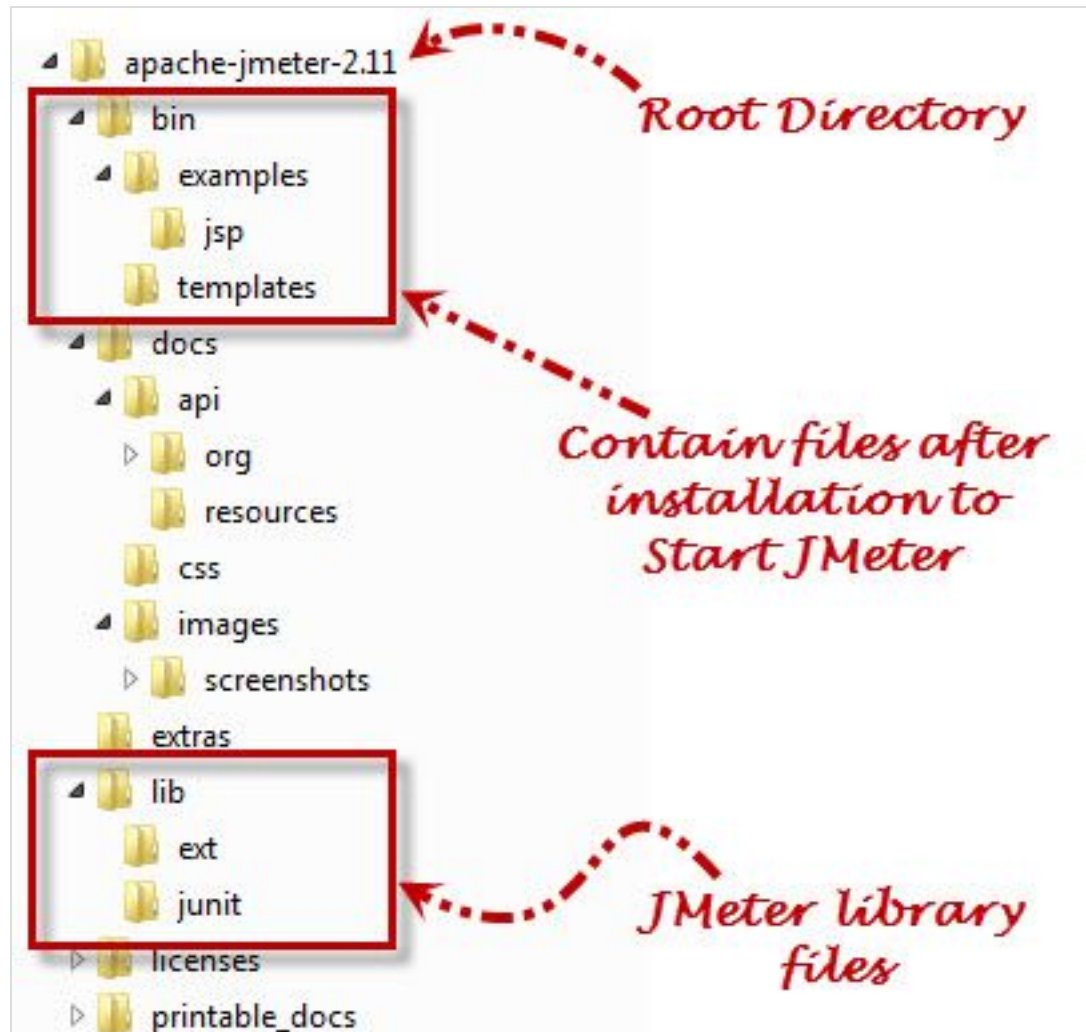
After checking Java, just download JMeter through the site: [Download Latest JMeter](#)

The latest version of JMeter is v2.11, go to the site and choose the Binaries file (either zip or tgz) to download as shown in figure below,



Step 2: Installation process of v2.11 JMeter

It is very simple and easy to install JMeter, just download the ".zip" or ".tgz" file and unzip JMeter file into the directory where you want to install JMeter. A very simple and easy installation process and it's done. After unzip process is done, the directory will look like the given below figure,



Files and directory which generate after installation of JMeter are given below,

/bin: Contains script file which comes after installation of JMeter to start JMeter

/docs: Contains JMeter documentation files

/extras: ant connected additional files

/lib/: Contains Java library files of JMeter

/lib/ext: Contains protocols and core jar files for JMeter

/lib/junit: Contains JUnit library files of JMeter

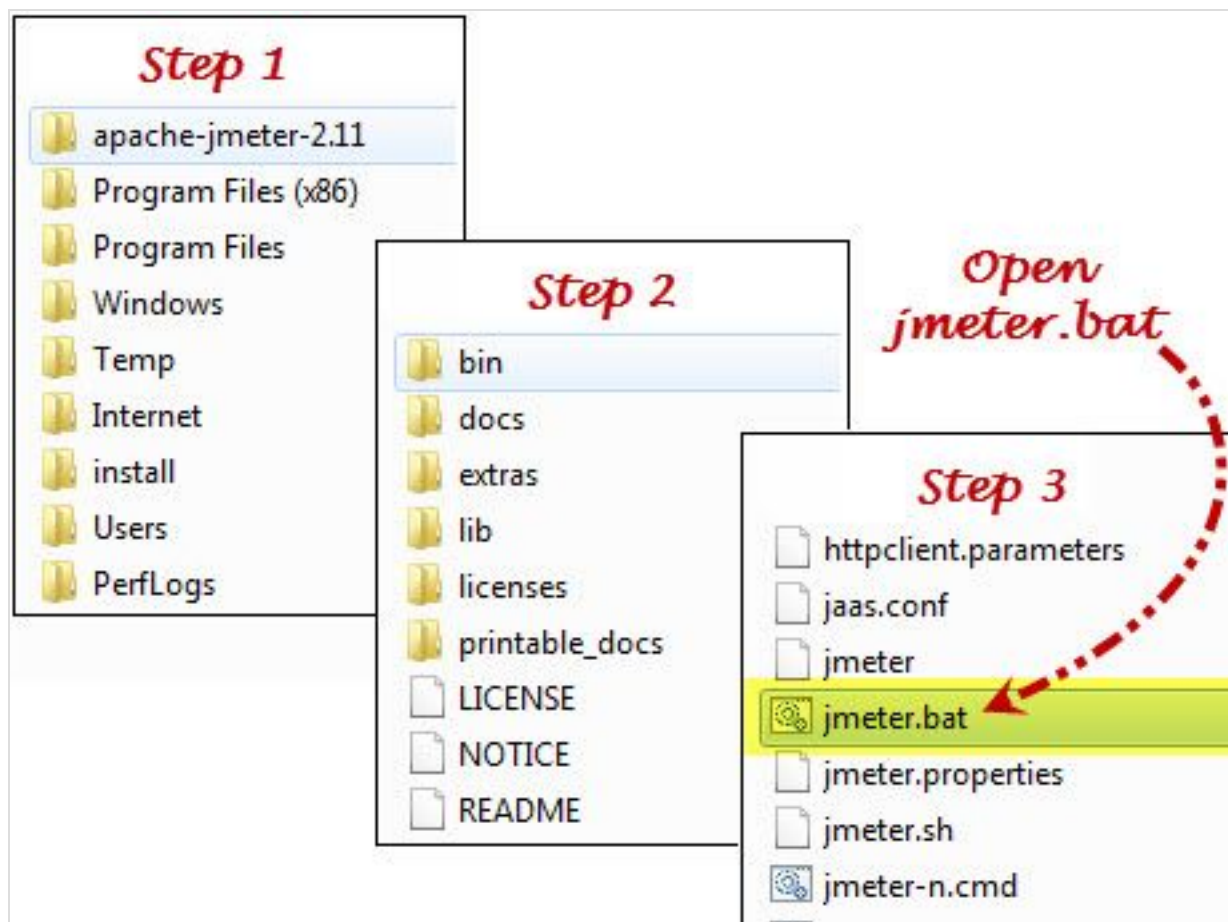
To check, it got installed or not on Windows, go to the directory where did you unzip the .zip or .tgz file, open Apache-JMeter-2.11 folder, click on bin folder, and then click on JMeter.bat file will open v2.11 JMeter application. In UNIX machine type ./bin/JMeter on command prompt to open JMeter.

Start JMeter in three modes

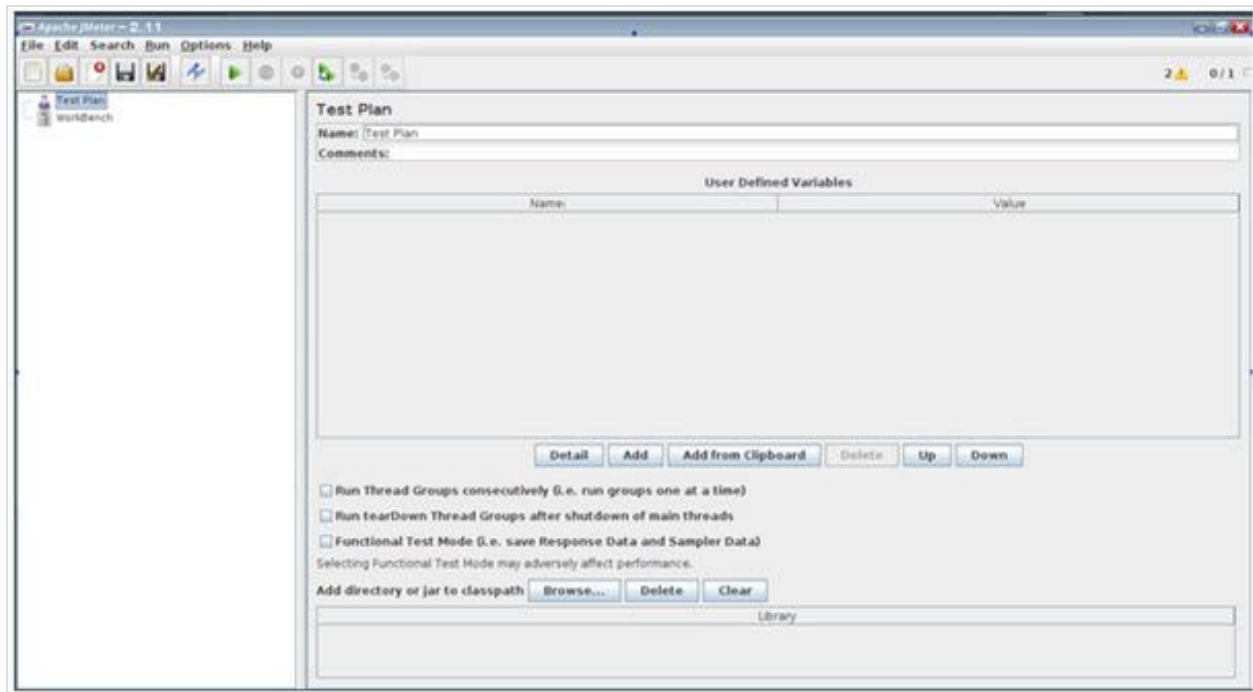
You can open JMeter using 3 modes – GUI Mode, Server Mode and Command Line Mode

1) In GUI Mode: JMeter runs on client computer

In Windows – First, open Apache-JMeter-2.11 folder, click on bin folder, and then click on JMeter.bat file will open v2.11 JMeter application. Shown in figure below,



Once you click on JMeter.bat, it opens JMeter – 2.11, shown in figure below,



2) In Server Mode: JMeter runs on server computer:

It is basically a **client-server** model distributed testing mode. In this process, JMeter runs on server computer as a server component. To start JMeter in server mode, run the bat file: **bin\JMeter-server.bat**, shown in the figure below,



3) JMeter in Command line mode:

If we open JMeter in GUI mode, it takes more computer memory to open it. If you don't want your computer should consume more resources to run JMeter, open JMeter in command line mode.

Type this command in command line: "**JMeter -n -t testPlan.jmx -l log.jtl -H 127.0.0.1 -P 8000**" to open JMeter, shown in the figure below,

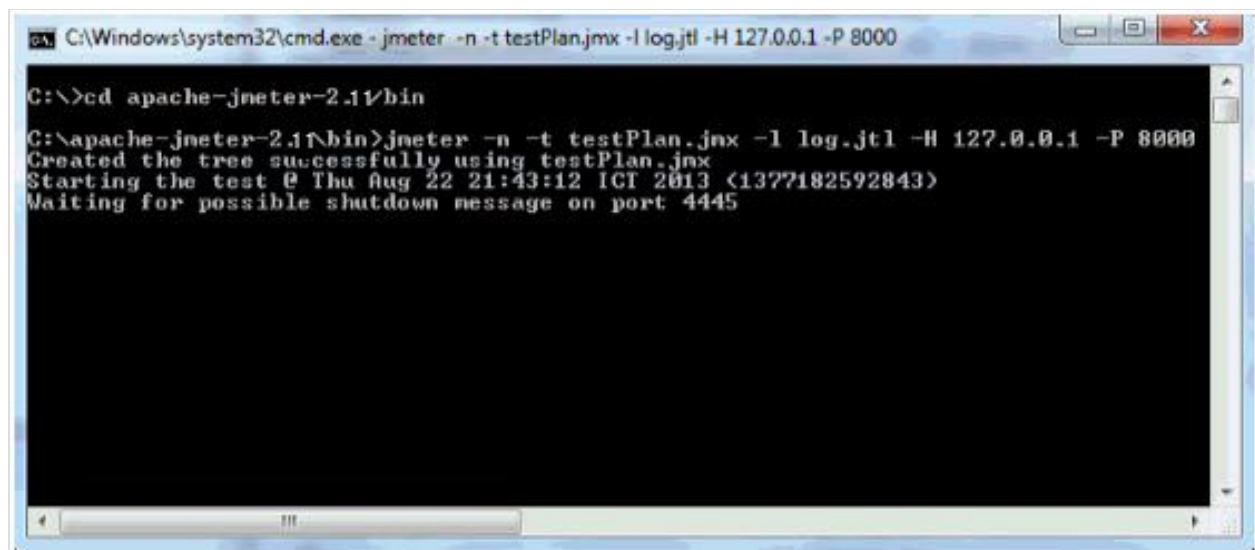
Where,

JMeter-n: Shows that JMeter will run on command line mode

-t testPlan.jmx: Name of file contains the Test plan

-l log.jtl: Shows the log files which contains the test result

-H 127.0.0.1 -P 8000: Proxy server host name and port



```
C:\Windows\system32\cmd.exe - jmeter -n -t testPlan.jmx -l log.jtl -H 127.0.0.1 -P 8000

C:\>cd apache-jmeter-2.11\bin

C:\apache-jmeter-2.11\bin>jmeter -n -t testPlan.jmx -l log.jtl -H 127.0.0.1 -P 8000
Created the tree successfully using testPlan.jmx
Starting the test @ Thu Aug 22 21:43:12 ICT 2013 (1377182592843)
Waiting for possible shutdown message on port 4445
```

Conclusion

In this article we have learn about how to download & install JMeter. Also we have learn about three different mode to run JMeter.

Elements of JMeter

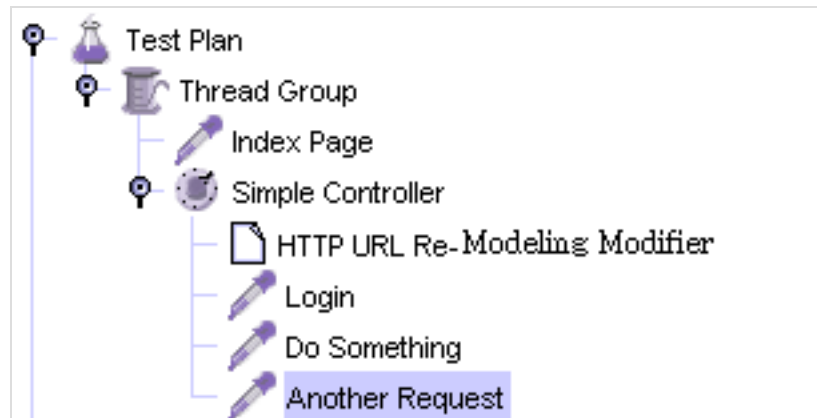
In previous article we have cover [How to install JMeter on your machine](#) to start running your first JMeter script. The different part or component of JMeter is called Element which co-relates with each other but designed for different-different purpose. Before start working on Jmeter, it is best to know all components or Elements of Jmeter with full detail description. From this article you will get to know the Elements of JMeter and why they are useful in JMeter.

Given below figure presents you some common elements in JMeter,

- Test Plan
- Thread Group
- Controllers
- Listeners
- Timers
- Configuration Elements
- Pre-Processor Elements
- Post-Processor Elements
- Execution order of Test Elements

Test Plan

A test plan is the top level body of JMeter, explains sequence of steps execute at run time. A final test plan made up of one or more Thread Groups, Sampler, logic controllers, listeners, timers, assertions, and configuration elements. Each Sampler can be preceded by one or more Pre-processor element, followed by Post-processor element, and/or Assertion element. It saves in Java Management Extensions (JMX) format.

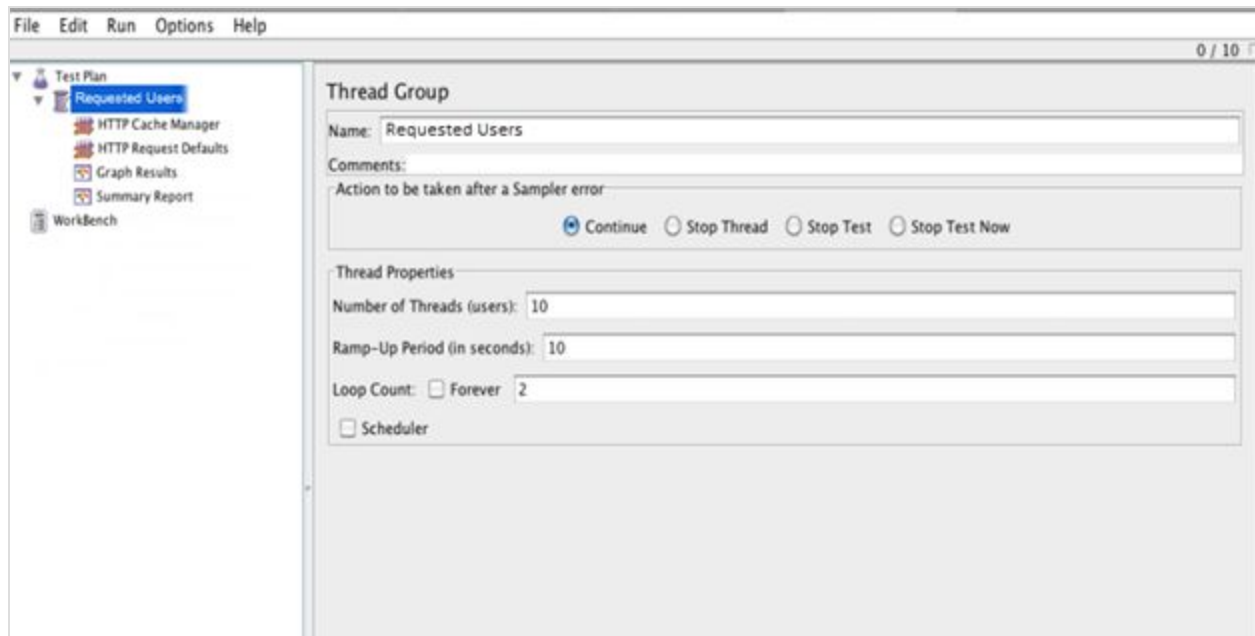


JMX is an open text based format, used across dissimilar systems. This format makes test plan to be open in a text editor and can use the “find and replace” function to speedily change a CMS name, web server name, or document ID that might appear hundreds of times throughout the test plan with very little determination.

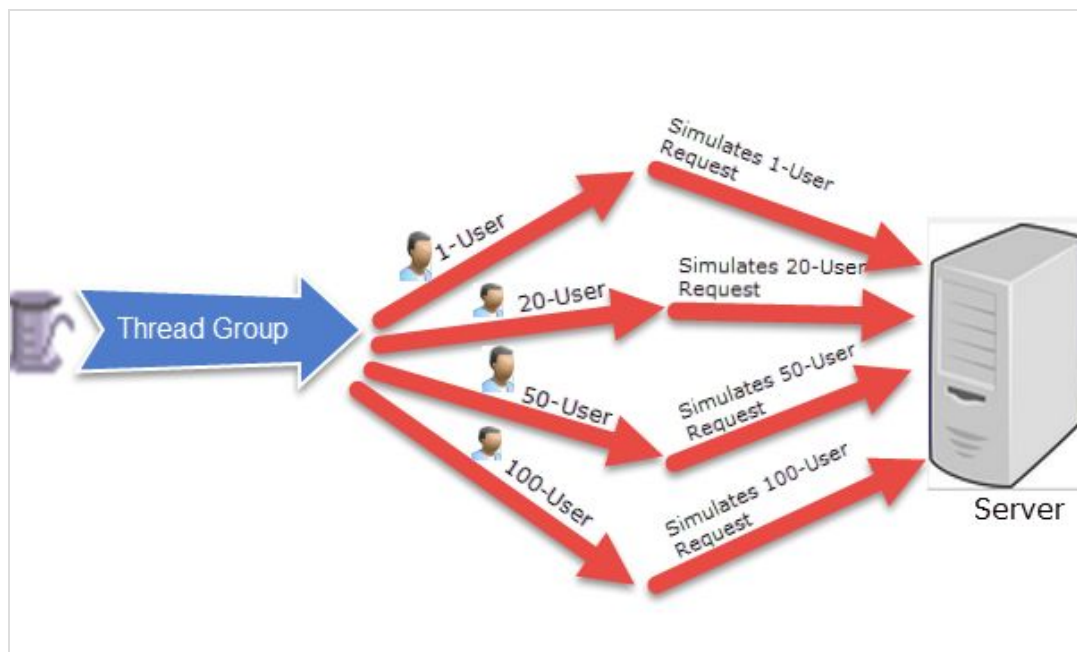
Thread Group

Thread Group is an initial stage of test plan. The name, Thread Groups represent a group of Threads. Under this group, each thread simulates one real user requests to the server.

A control on a thread group facilitates to set the number of threads on each group. Also can control Ramp Up Time and the number of test iterations.



If you mention 10 numbers of threads then JMeter will generate and simulate 100 user requests to the server under test.



Controllers

There are two types of controllers: 1) Samplers 2) Logical Controllers

1) Samplers

Samplers facilitate JMeter to deliver explicit types of requests to the server. It simulates a user's request for a page from the target server. So that, to avail POST, GET, DELETE functions on a HTTP service, user can add HTTP Request sampler. Apart from HTTP Request sampler, there are other simpler too,

- HTTP Request
- FTP Request
- JDBC Request
- Java Request
- SOAP/XML Request
- RPC Requests

An HTTP Request Sampler looks like this figure,

The screenshot shows the JMeter HTTP Request sampler configuration window. The window has a menu bar (File, Edit, Run, Options, Help) and a status bar (0 / 10). The left sidebar shows a tree view with 'Test Plan', 'Thread Group', 'HTTP Request', and 'WorkBench'. The main area is titled 'HTTP Request' and contains the following fields and options:

- Name:** (text field)
- Comments:** (text area)
- Web Server:**
 - Server Name or IP:** (text field)
 - Port Number:** (text field)
 - Timeouts (milliseconds):**
 - Connect:** (text field)
 - Response:** (text field)
- HTTP Request:**
 - Implementation:** (dropdown menu, currently 'Java')
 - Protocol [http]:** (text field, currently 'http')
 - Method:** (dropdown menu, currently 'POST')
 - Content encoding:** (text field)
 - Path:** (text field)
 - Options:**
 - ☐ Redirect Automatically
 - ☒ Follow Redirects
 - ☒ Use KeepAlive
 - ☐ Use multipart/form-data for POST
 - ☐ Browser-compatible headers
- Parameters:** (tabbed view, currently selected)
 - Send Parameters With the Request:** (table)
- Post Body:** (tabbed view, currently not selected)

The 'Send Parameters With the Request' table has the following structure:

Name	Encode?	Include Equals?
	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	<input checked="" type="checkbox"/>

At the bottom of the window are buttons: Detail, Add, Add from Clipboard, Delete, Up, and Down.

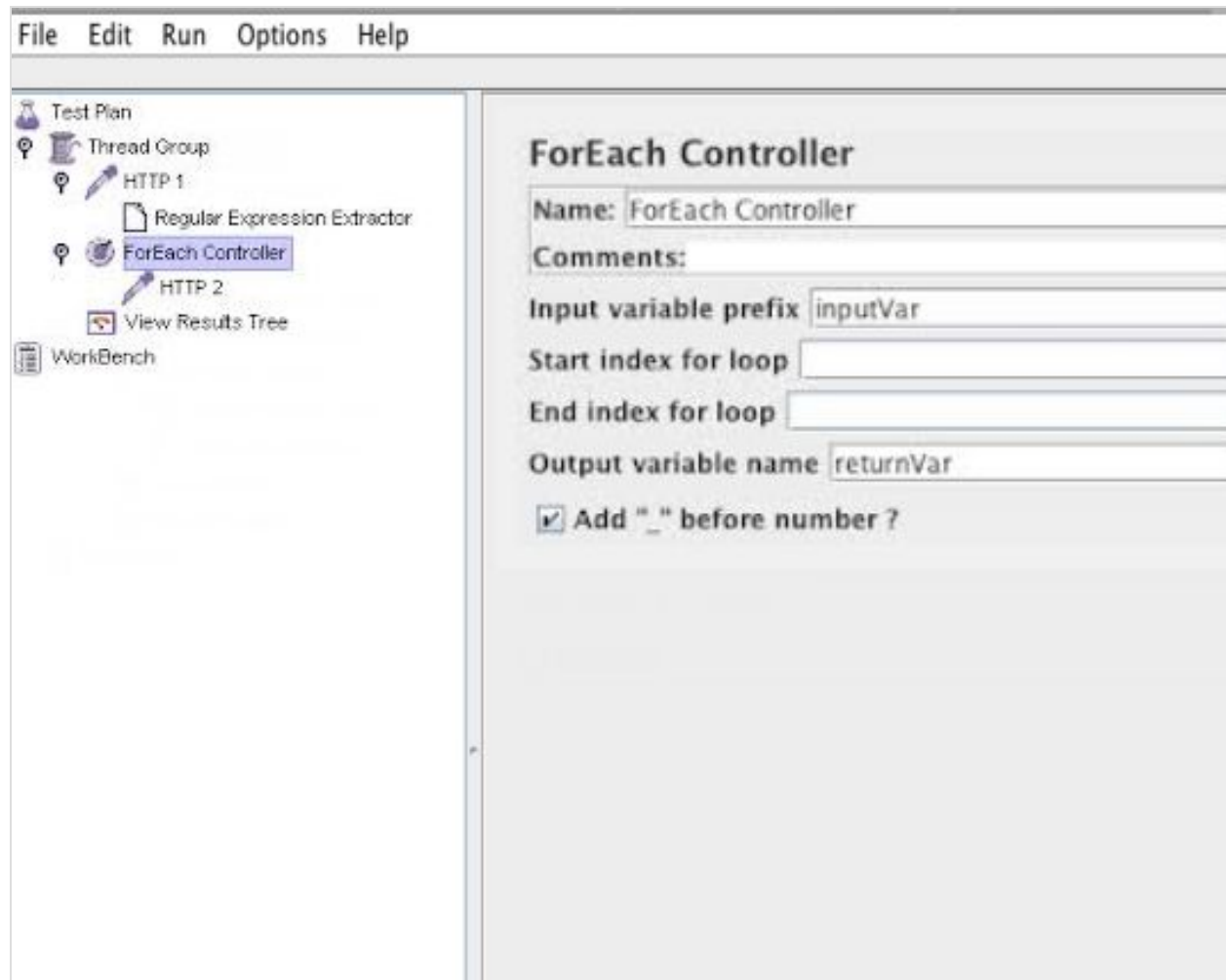
2) Logical Controllers

Logic Controllers decide the order of processing of Samplers in a Thread. It offers a mechanism to control the flow of the thread group. Logic Controllers facilitates to customize the logic that JMeter uses to resolve when to send requests. Logic Controllers can alter the order of requests coming from their child elements.

Logic Controllers of JMeter provide,

- Runtime Controller
- IFController
- Transaction Controller
- Recording Controller
- Simple Controller
- While Controller
- Switch Controller
- ForEach Controller
- Module Controller
- Include Controller
- Loop Controller
- Once Only Controller
- Interleave Controller
- Random Controller
- Random Order Controller
- Throughput Controller

A ForEach Controller Control Panel looks like the following figure,



Test Fragments

Test Fragments is a different type of element located at the same level as Thread Group element. It is eminent from a Thread Group in that it is not performed unless it is referenced by either a Module Controller or an Include_Controller. This element is morally for code re-use within Test Plans.

Listeners

Listeners facilitates viewers to view Samplers result in the form of tables, graphs, trees or simple text in some log files and also give pictorial access to the data collected by JMeter about those test cases as a Sampler component of JMeter is executed. Listeners offer a means to collect, save, and view the results of a test plan and store results in XML format, or a more efficient (but less detailed) CSV format. Their output can also be viewed directly within the JMeter console.

It basically adjusts anywhere in the test, moreover under the test plan, gather data only from elements at or below their level.

Listeners of JMeter provide these many things,

- Graph Results
- Spline Visualizer
- Assertion Results
- Simple Data Writer
- Monitor Results
- Distribution Graph (alpha)
- Aggregate Graph
- Mailer Visualizer
- BeanShell Listener
- Summary Report
- Sample Result Save Configuration
- Graph Full Results
- View Results Tree
- Aggregate Report
- View Results in Table

Timers

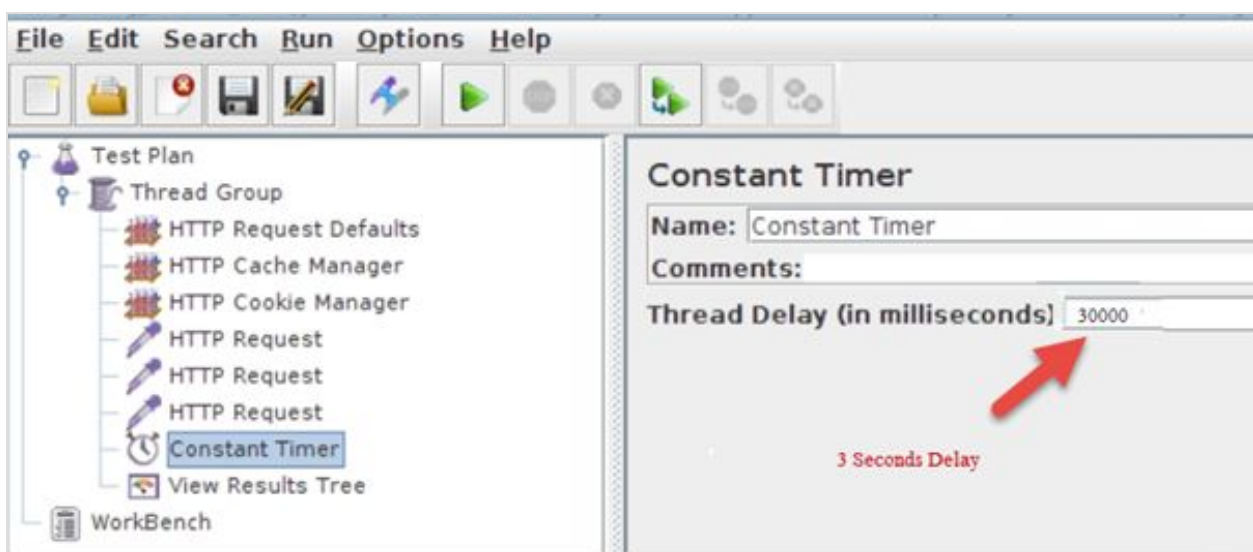
To perform load/stress testing on your application, you are using threads, controllers and samplers then JMeter will just blast your application with nonstop requests. This is not the real environment or characteristic of the real traffic. JMeter thread sends request without stopping between each sampler. This not exactly what you want. We can add a timer element which will permit us to define a period to wait between each request.

So, to simulate real traffic or create extreme load/stress – orderly, JMeter provides you Timer elements.

Just have a look on the types of Timers available ,

- Synchronizing Timer
- JSR223 Time
- BeanShell Time
- Gaussian Random Timer
- Uniform Random Timer
- Constant Throughput Timer
- BSF Time
- Poisson Random Time

As an example, the Constant Timer Control Panel looks like this:



Configuration Elements

Configuration Element is a simple element where you can collect the corporate configuration values of all samplers like webserver's hostname or database url etc.

These are some commonly used configuration elements in JMeter,

- Java Request Defaults
- LDAP Request Defaults
- LDAP Extended Request Defaults
- Keystore Configuration
- JDBC Connection Configuration
- Login Config Element
- CSV Data Set Config
- FTP Request Defaults
- TCP Sampler Config
- User Defined Variables
- HTTP Authorization Manager
- HTTP Cache Manager
- HTTP Cookie Manager
- HTTP Proxy Server
- HTTP Request Defaults
- HTTP Header Manager
- Simple Config Element
- Random Variable

Pre-Processor Elements

A Pre-Processor is an interesting thing which executes before a sampler executes. It is basically used to modify the settings of a Sample Request before it runs, or to update variables that are not extracted from response text.

These are the list made up of all the Pre-Processor Elements JMeter,

- JDBC PreProcessor
- JSR223 PreProcessor
- RegEx User Parameters
- BeanShell PreProcessor
- BSF PreProcessor
- HTML Link Parser
- HTTP URL Re-writing Modifier
- HTTP User Parameter Modifier
- User Parameters

Post-Processor Elements

A Post Processor gets executed after finishing a sampler execution process. This element is basically useful to procedure the response data, for example, to retrieve particular value for later use.

These are the list made up of all the Post-Processor Elements JMeter,

- CSS/JQuery Extractor
- BeanShell PostProcessor
- JSR223 PostProcessor
- JDBC PostProcessor
- Debug PostProcessor
- Regular Expression Extractor
- XPath Extractor
- Result Status Action Handler
- BSF PostProcessor

Test Elements, Execution series

These are the execution order of the test plan elements,

- Configuration elements
- Pre-Processors
- Timers
- Sampler
- Post-Processors (unless SampleResult is null)
- Assertions (unless SampleResult is null)
- Listeners (unless SampleResult is null)

Conclusion

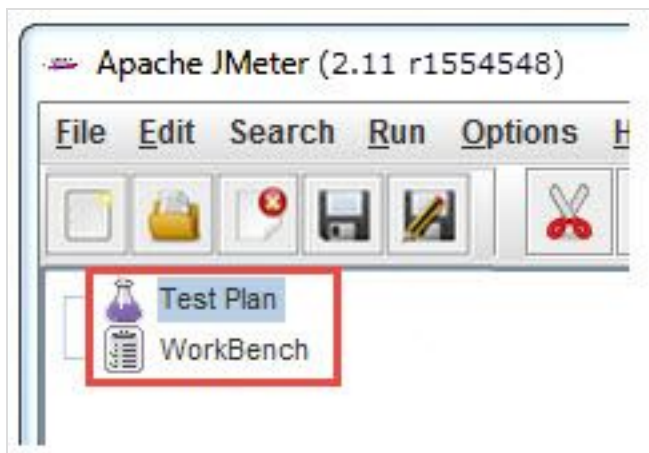
In this article we have learn about all components or Elements of Jmeter with full detail description.

Hands on with JMeter GUI

Until now we have seen about Introduction of JMeter, Installation Guide for JMeter & what all Elements supported by JMeter in earlier [JMeter training series](#). In this article we are focusing on what is mean by Test Plan & WorkBench in JMeter. We are covering how to add and remove JMeter Elements, How to Load and Save JMeter Elements. Alongwith that we are covering how we configure Jmeter Elements. Finally we will see How to save a JMeter Test Plan, Running & stopping Test in JMeter.

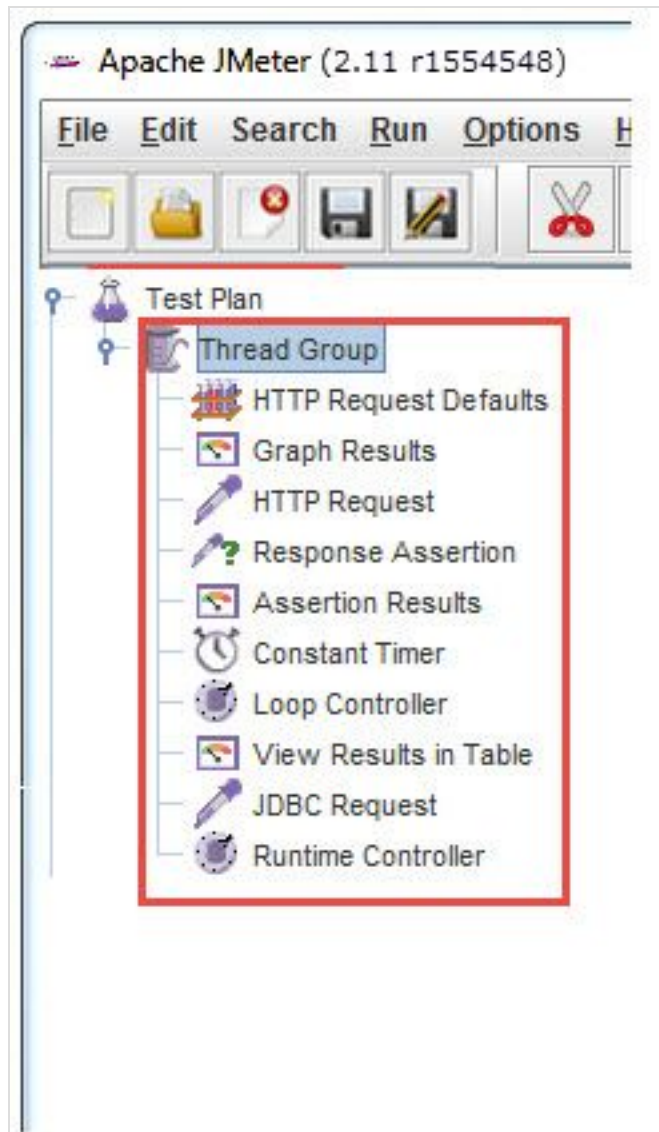
After opening JMeter. Firstly, our eyes go to these two Elements,

1. Test Plan
2. Workbench



Test Plan

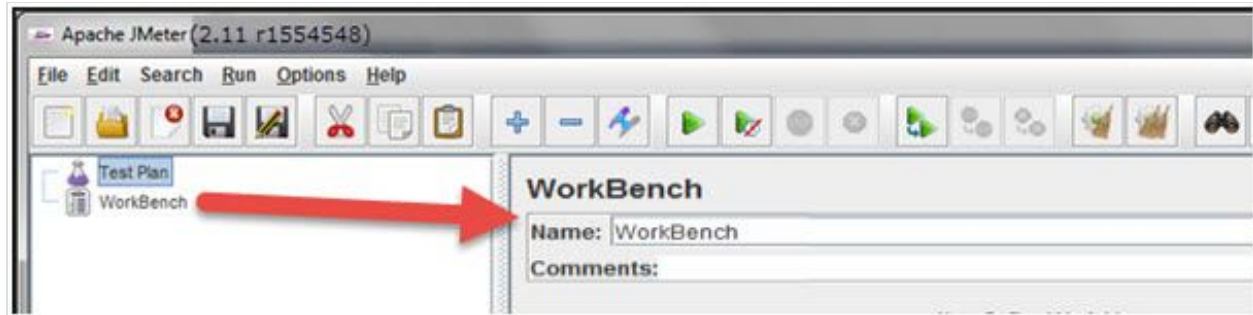
A test plan shows a series of Elements are useful in application tasting process. These Elements are: Thread Groups, logic controllers, sample generating controllers, listeners, timers, assertions, and configuration elements.



WorkBench

The WorkBench is simply a place which creates space to temporarily store test elements at the time of constructing test. In other word, it is a sandbox or portion of a test on which you are working on. That's why, if the designed test in the WorkBench is ready to proceed, copy that and move it into Test Plan. Jmeter only save the contents of Test Plan not WorkBench.

WorkBench keeps non-test Elements too: Http Mirror Server and Http Mirror Server, these elements are not available in the Thread Group and Test Plan.



Add and Remove JMeter Elements

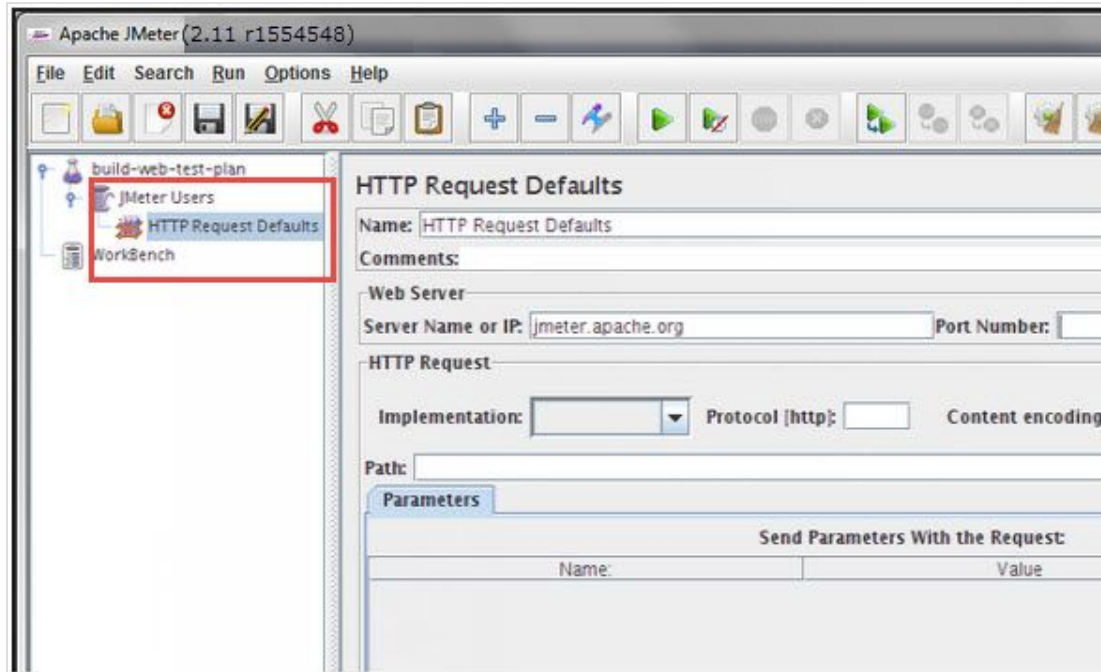
Adding Elements in Test Plan is very important step to execute your Test Plan. Without going through this, JMeter is unable to execute your Test Plan. A Test Plan is capable to include many Elements such as Listener, Controller, and Timer. To add Element in the Test Plan, right click on Test Plan which opens context-menu of Test Plan, then choose new elements from "**Add**" list. Alternatively, elements can be loaded from file and added by choosing the "merge" or "open" option.

Before adding any Element in the Test Pane, it is necessary to add first a Thread Group element. The Thread Group says JMeter the number of users you need to simulate, how frequently the users should send requests, and how many requests they should send.

So, first let us know how to add the Thread Group element, right click on Test Plan which opens context-menu of Test Plan and then select Add -> Thread Group.

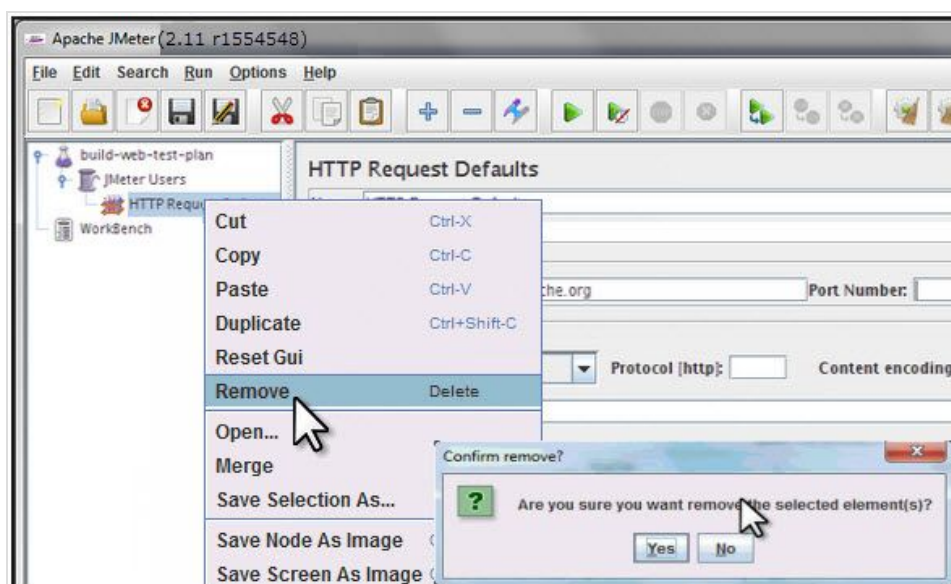
After adding the Thread Group, the Thread Group should display under Test Plan. If it is not visible, then "expand" the Test Plan tree by clicking on the Test Plan element.

After defining Users, define the task on which they will work. Let us take an example of HTTP Request Defaults Element, add this element by right clicking on Users and then choose Add -> Config Element -> HTTP Request Defaults.



As like adding, you can also remove the Element which is not required. Let us take an example of "HTTP Request Defaults" Element. To remove this Element, right click on "HTTP Request Defaults" and from context-menu choose "Remove" option. It will display the dialog box with message "Are you sure you want to remove the selected Element", if yes, then click on "Yes" button, or if no, then click on "No" button

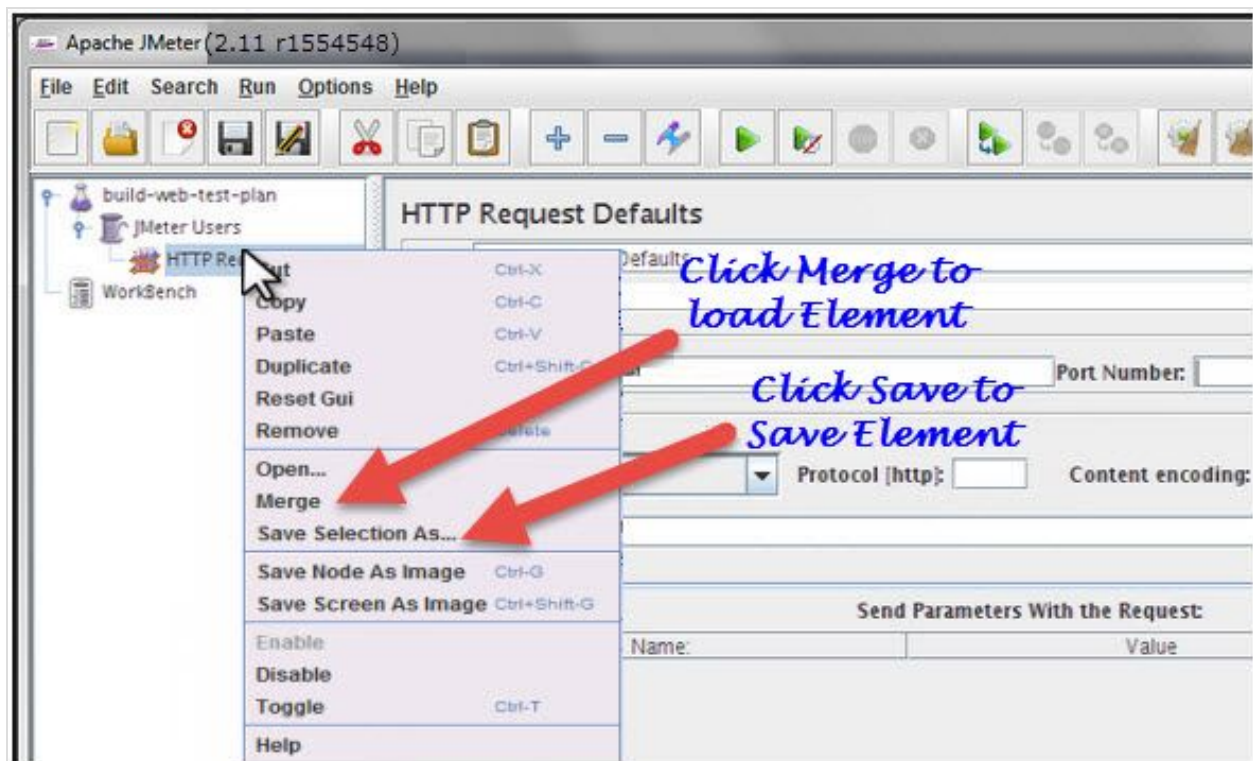
Click **Yes** to confirm delete this element on message box.



Load and Save JMeter Elements

a) Load Elements

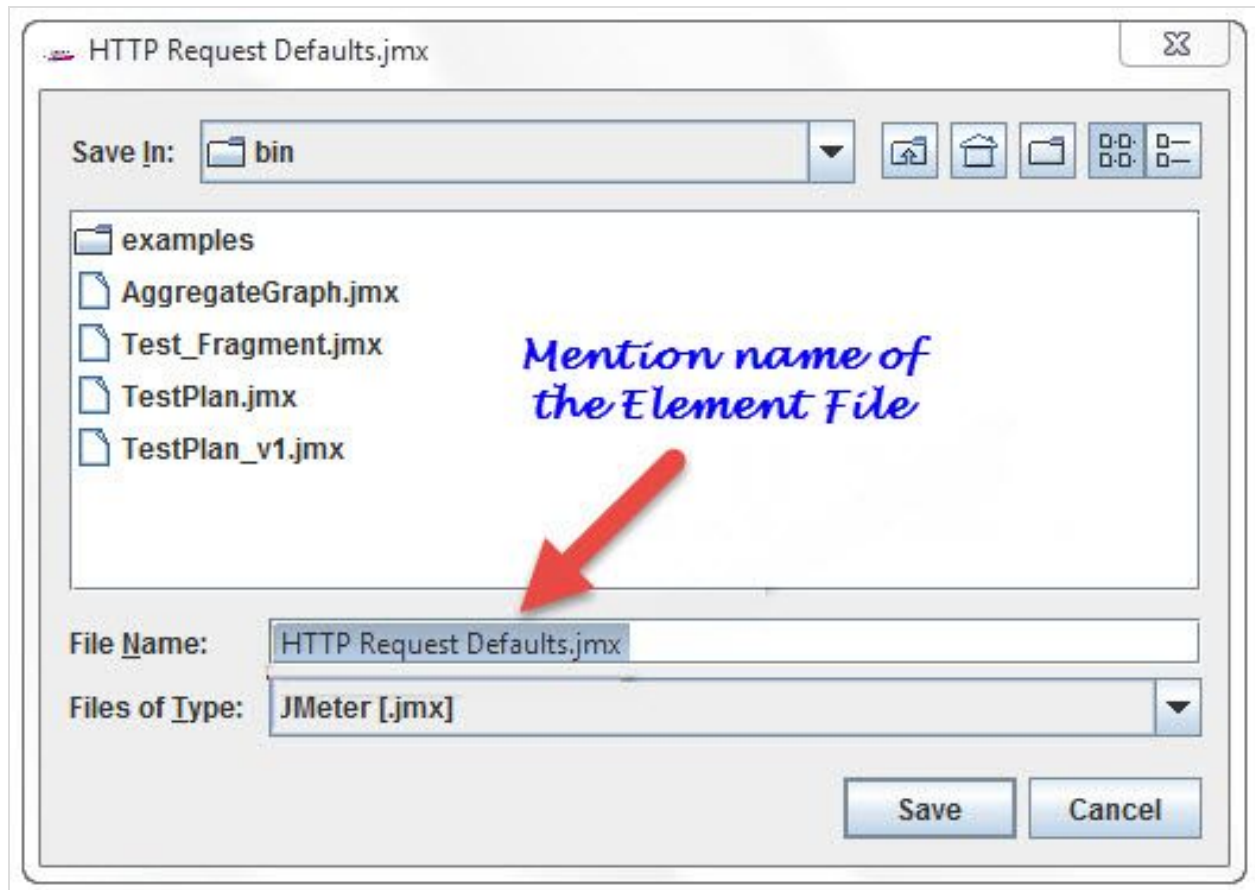
To load the element from existing file, choose the element from existing tree on which you want to load the other element, then right click on the selected element and click on "Merge" option. Choose the file from the existing saved Element. JMeter will merge the elements into the tree.



b) Save Elements

To save tree elements, choose the element from the tree, right click on an element and from the context menu option choose the "Save Selection As..." option. JMeter will save the selected element, plus all child elements beneath it. Created file cannot be saved in jmeter by default; you have to explicitly save the element.

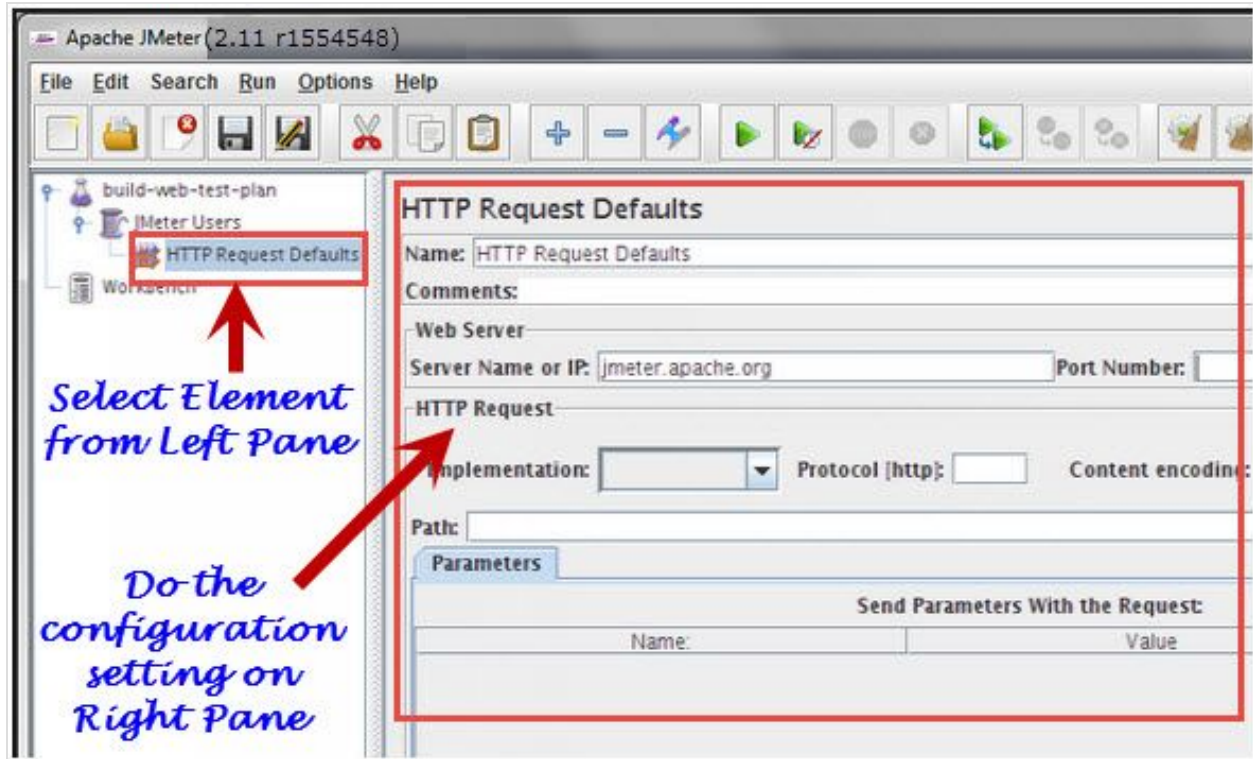
When you save the element, it will save with the default name of the element, you can save with the same name or you can change it. For example, The figure below shows the save dialog box with default name "**HTTP Request Defaults.jmx**". Either you can give the same name or can change it before clicking on "Save" button.



JMeter Test Elements and Test plan are stored in ***.JMX** format. **JMX** is standing for **Java Management Extensions**.

Configure Jmeter Elements

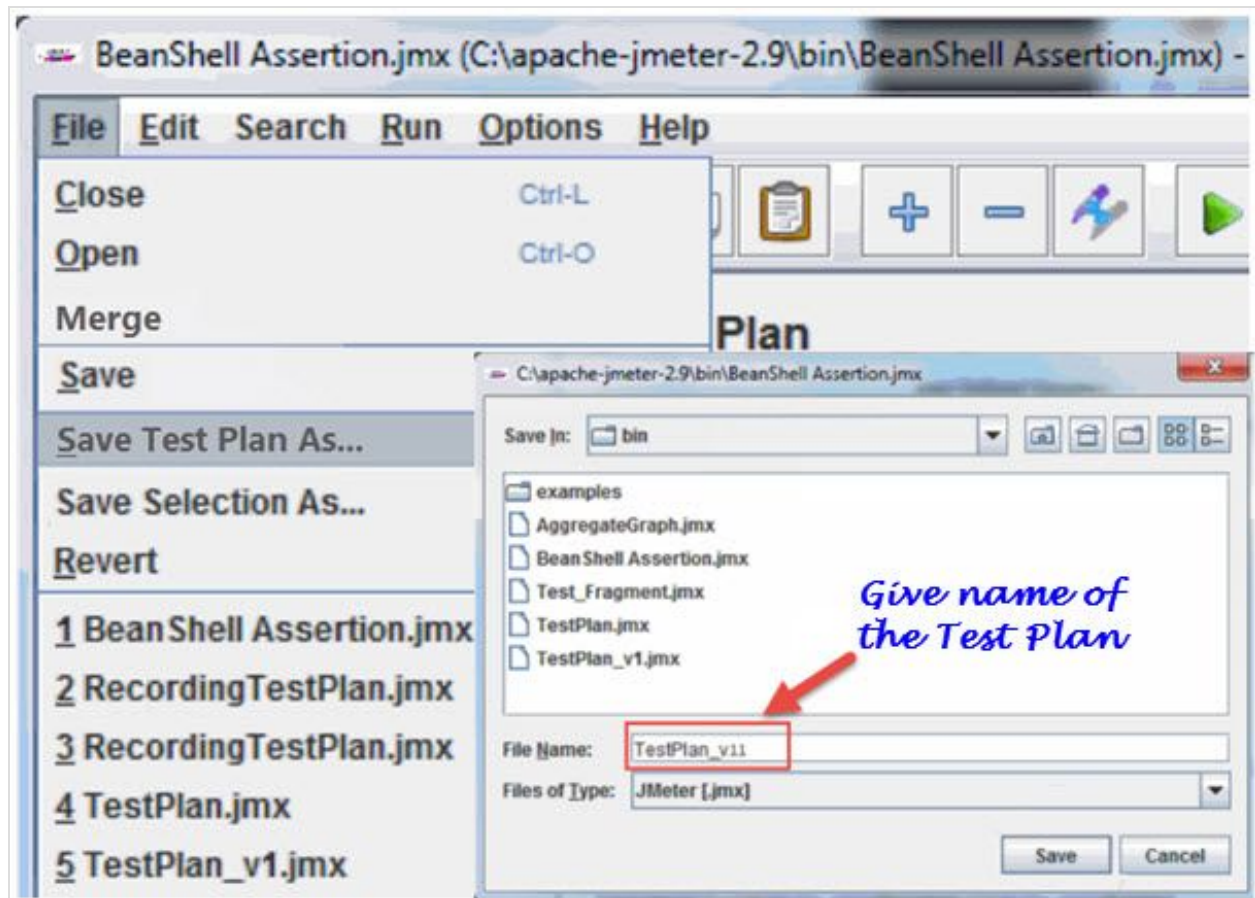
To configure any Element, first select the element of the Tree from Left Pane and do the configuration settings of the same element in the Right Pane.



You can configure any element of the Test Plan, using the element frame present in the right side of the JMeter window. These frames facilitate to configure the nature of the specific test element.

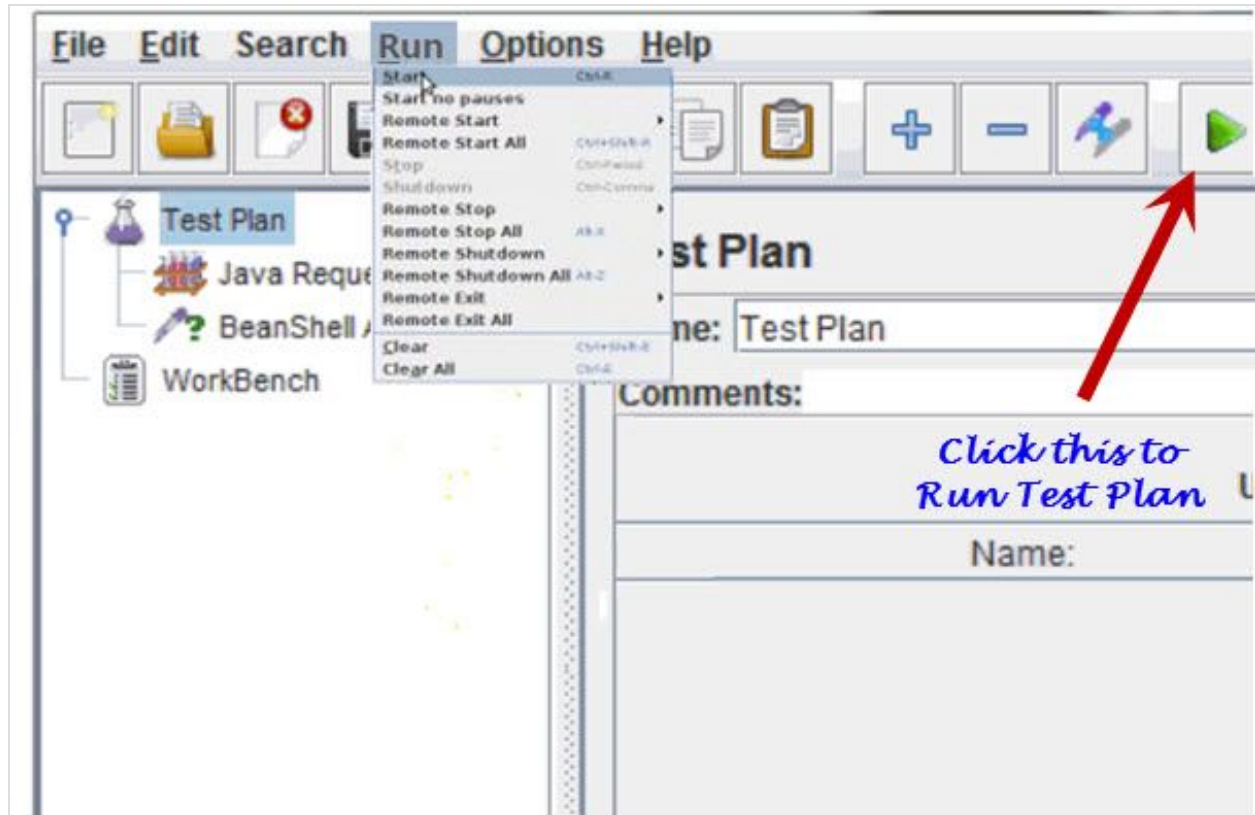
Saving the Test Plan

First save the Test Plan, before running a test. This saving helps to avoid surprising error comes at the time of running the test plan. To save the Test plan, first go to the File -> Save Test Plan. It displays the save Dialog box, give the file name of Test Plan and click on "Save".



Running a Test Plan

To run the Test Plan, from top of the menu item click on Run à Start or from the keyboard press "Control+R". Also, JMeter shows a small green button at the right hand side just under the menu bar – this is an alternative tool to Run Test Plan.



Stopping a Test

There are two ways to stop running test,

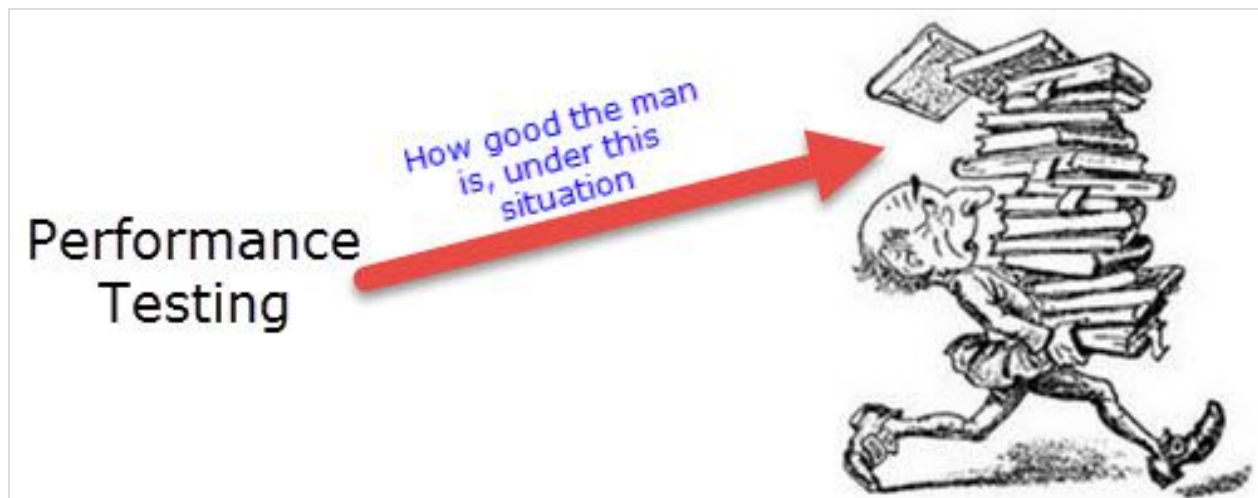
First Way: Click on "Stop", or from keyboard press Control + `'. This is an immediately process to stops the threads, if possible.

Second Way: Shutdown, or press -> Control + `'. This appeals the threads to stop at the end of any present work.

How to do Performance Testing using JMeter?

The focus of Performance testing is checking a software program's mainly focus on its Speed, Scalability and Stability. The features & functionality supported by software is not only the concerns for the performance testing. The goal of performance testing is not only the finding bugs in the code but also the find out the bottlenecks & eliminate them.

In terms of software testing process, performance testing is a general testing process works on the particular work loaded application, web application or system to determine how the application performs in terms of particular workload and how much the application is stable in work load situation. Performance testing also does examination, measurement, validation or verification to test scalability, reliability and resource usage of the application, web application or system. It can be used to analyse complete server performance under heavy load.

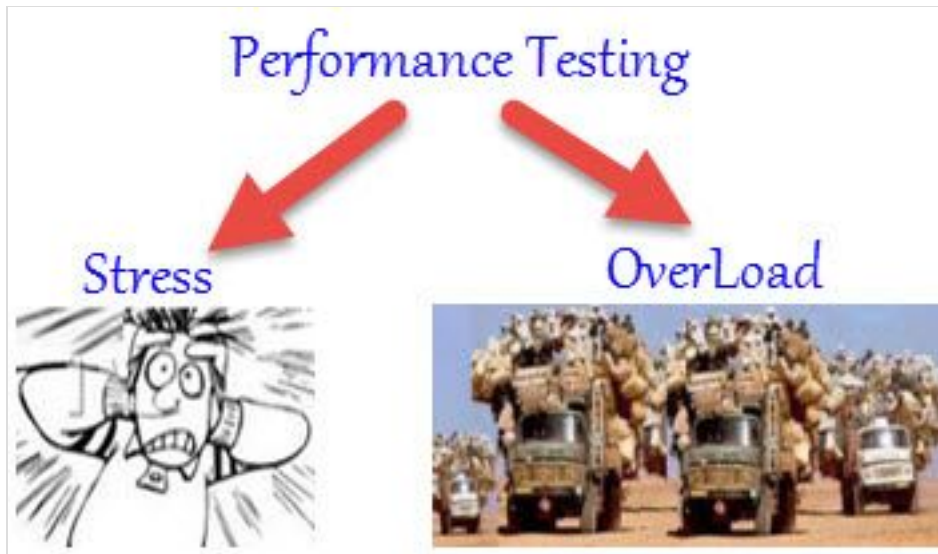


Using JMeter, Performance testing process provides following benefits to the system or application,

- JMeter is useful to test the performance of static resources such as JavaScript and HTML, and dynamic resources, such as JSP, Servlets, and AJAX.

- JMeter is useful to determine maximum number of simultaneous users that users website will be able to handle.
- JMeter is useful to provide a various graphical analyses of performance reports.

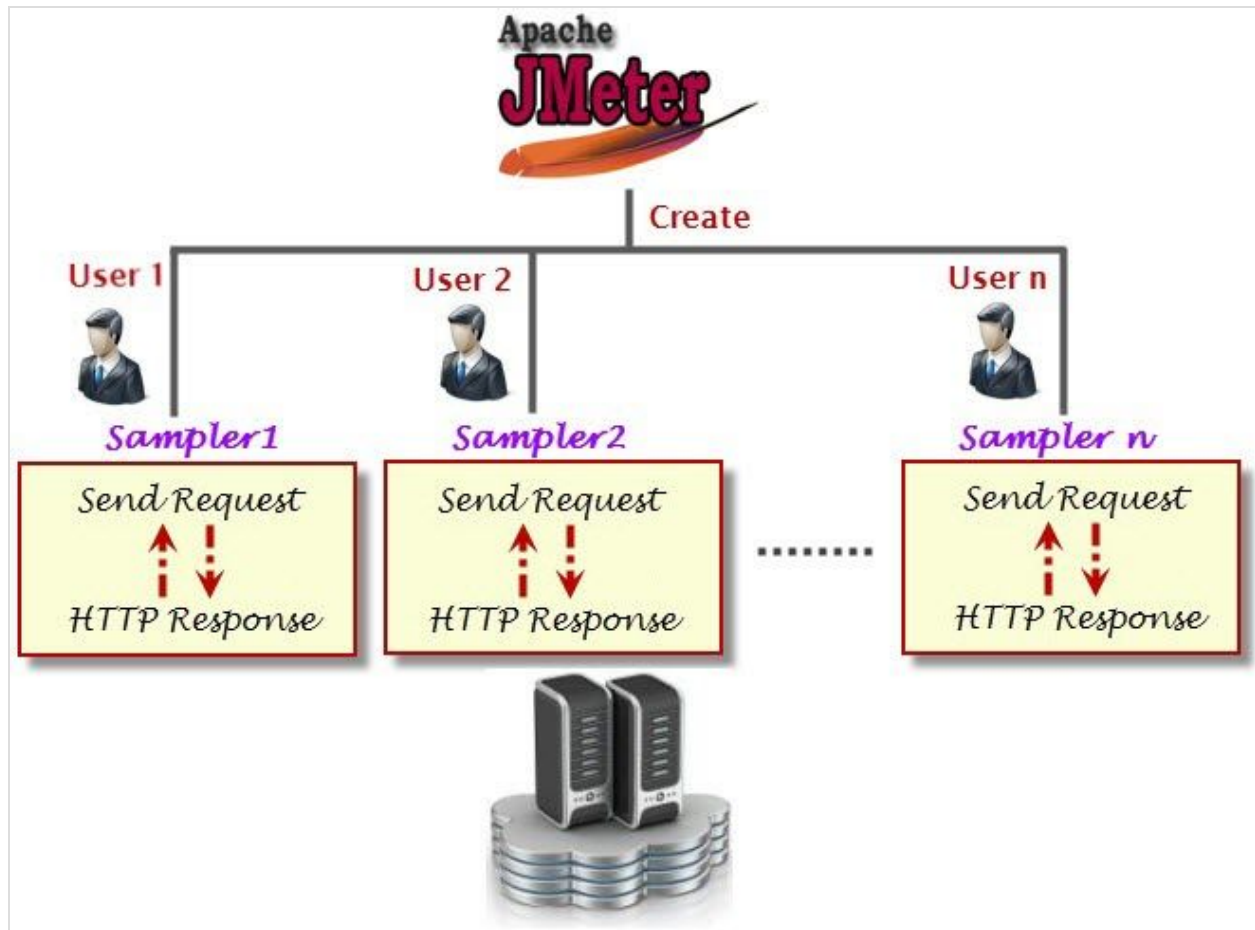
JMeter Performance Testing contains:



- **Load Testing:** Load testing is a part of performance testing to know the performance of the system under the load condition. Load testing can be performed when concurrent number of users performs of specific application at particular time period. Using JMeter, Load Testing models the estimated usage by simulating multiple user get the web services simultaneously.
- **Stress Testing:** Stress testing is important to recognize the upper limits of capacity within the system. Usually we go this testing to recognize the system's robustness in terms of excess load and assists application administrators to judge if the system will perform satisfactorily if the current load goes well above the ordinary maximum.

Any application or web server has a maximum load capacity, when server gets overloaded beyond its load capacity, it will start responding slowly and produce errors. In JMeter, Stress Testing is useful to determine the maximum load the web server can handle.

Following image will help you to understand how JMeter replicate heavy load conditions:



Steps to build-up a Performance Test Plan in JMeter

Here we will check the performance of Google.com accessed by 1000 users

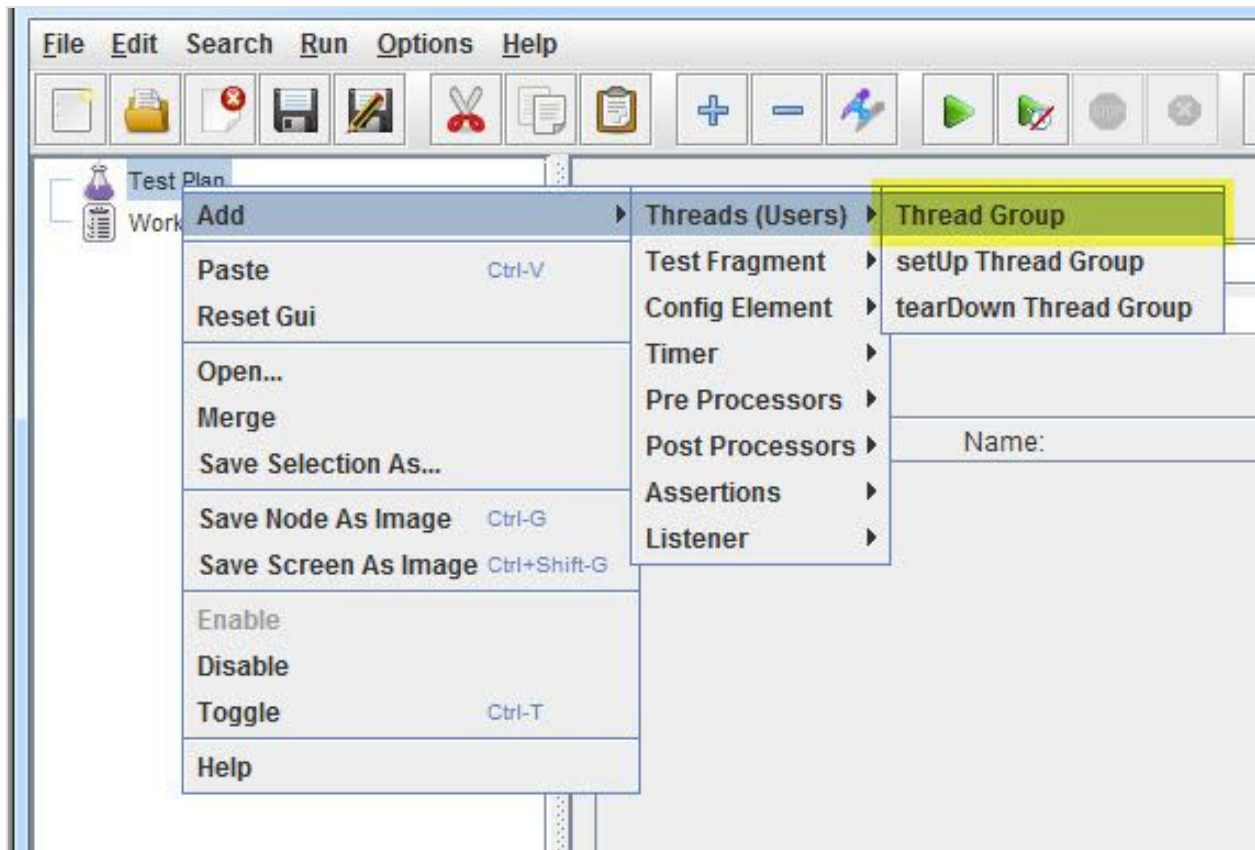
Before doing performance testing on web application, first to determine – Normal Load, Heavy Load and the target.

Normal Load determines that how many average number of users visit to the website.

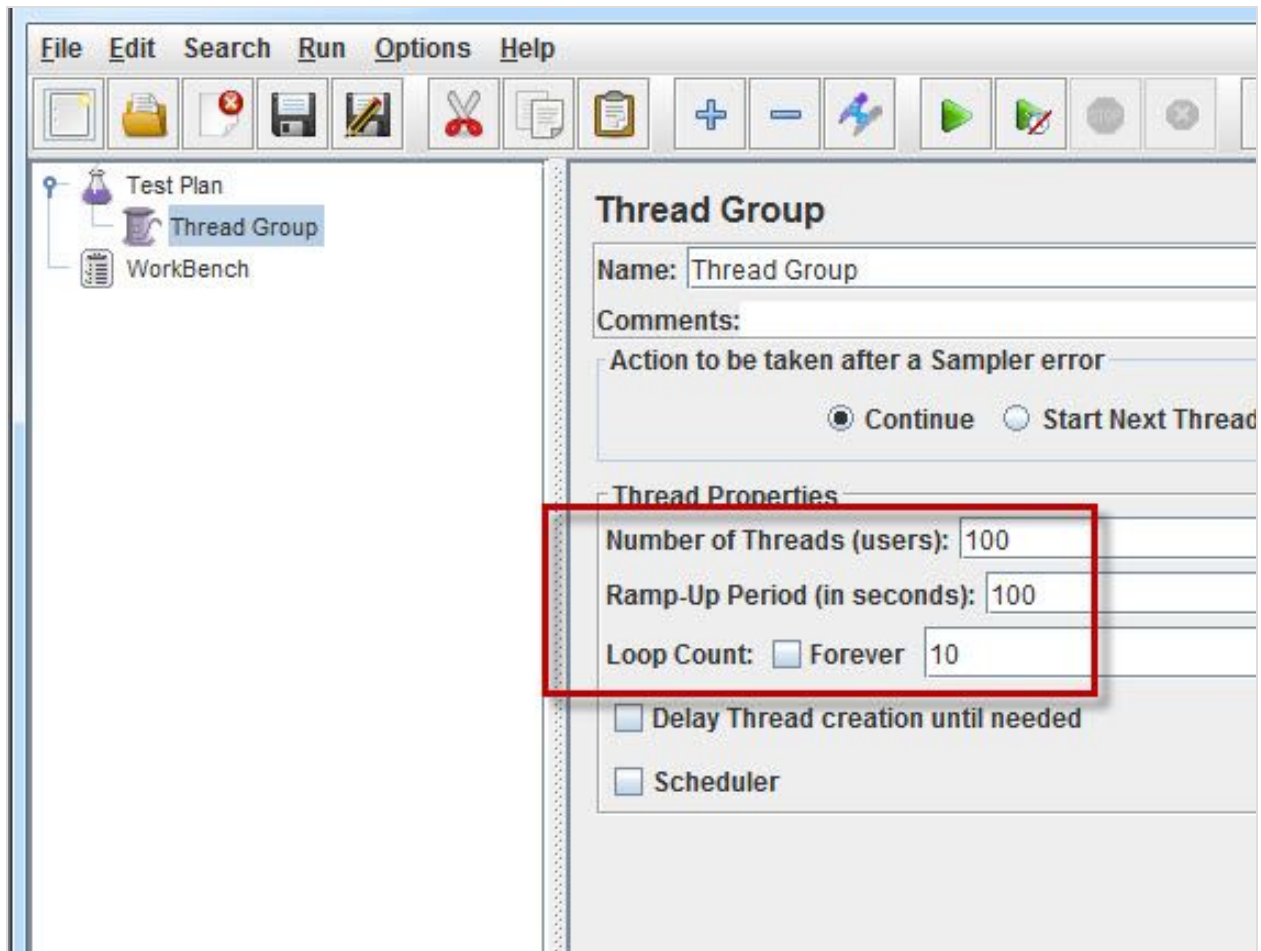
Heavy Load determines that how many maximum numbers of users visit your website and the target in this test.

Step 1: Create the Thread Group

To create the Thread Group, first run JMeter, from opened interface of JMeter choose Test Plan from the tree and right click to choose Add Threads (Users) Thread Group.

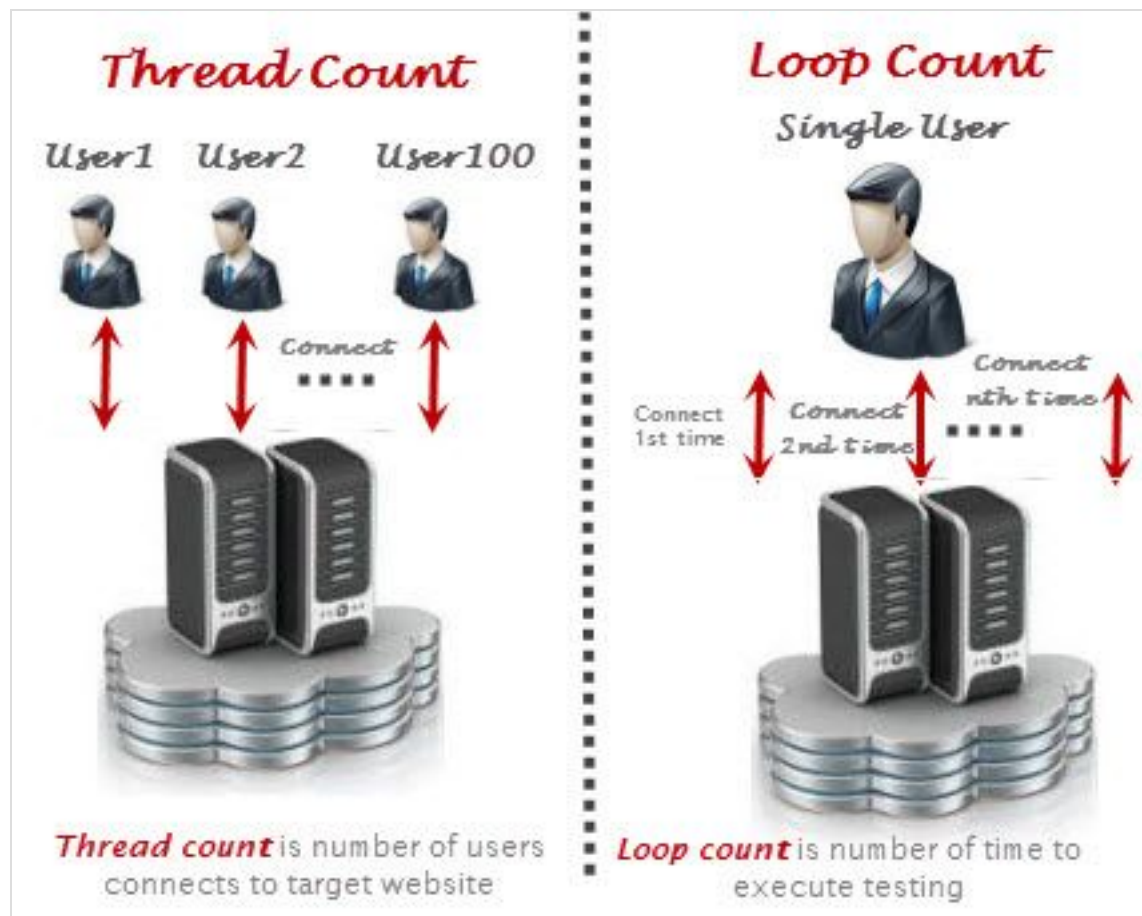


After opening thread Group, enter Thread Properties as shown in figure below,



In the above figure, **Number of Threads**: 100 numbers of users are connected to target website, **Loop Count**: execute testing 10 numbers of times, and **Ramp-Up Period**: 100.

The Thread Count and The Loop Counts both are **different**. In the Thread Count, it simulates 100 simultaneous users trying to connect the targeted website. In the Loop Counts, it simulates 1 user trying to connect the targeted website 10 times.



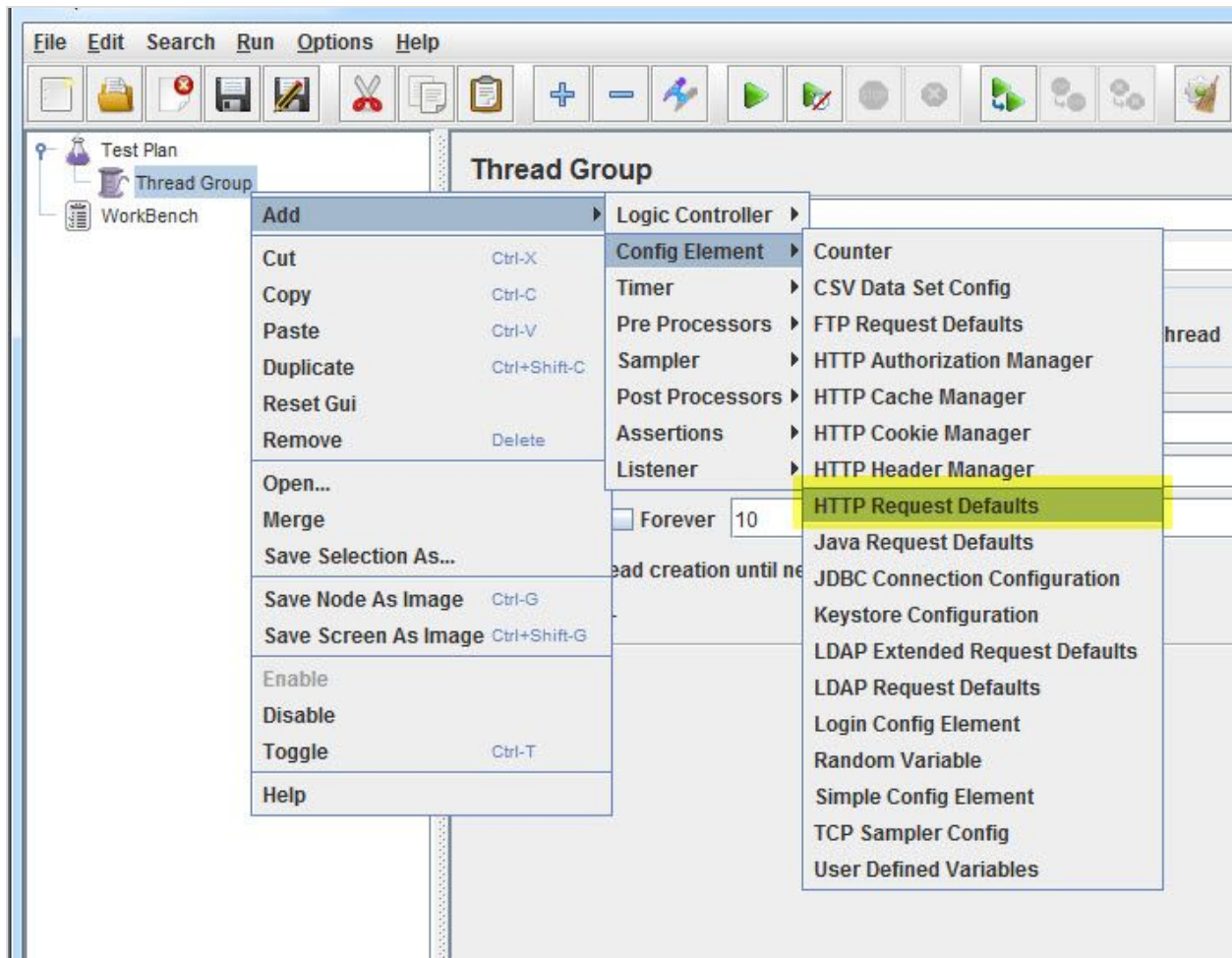
Step 2: Add JMeter elements

In this article we will Add,

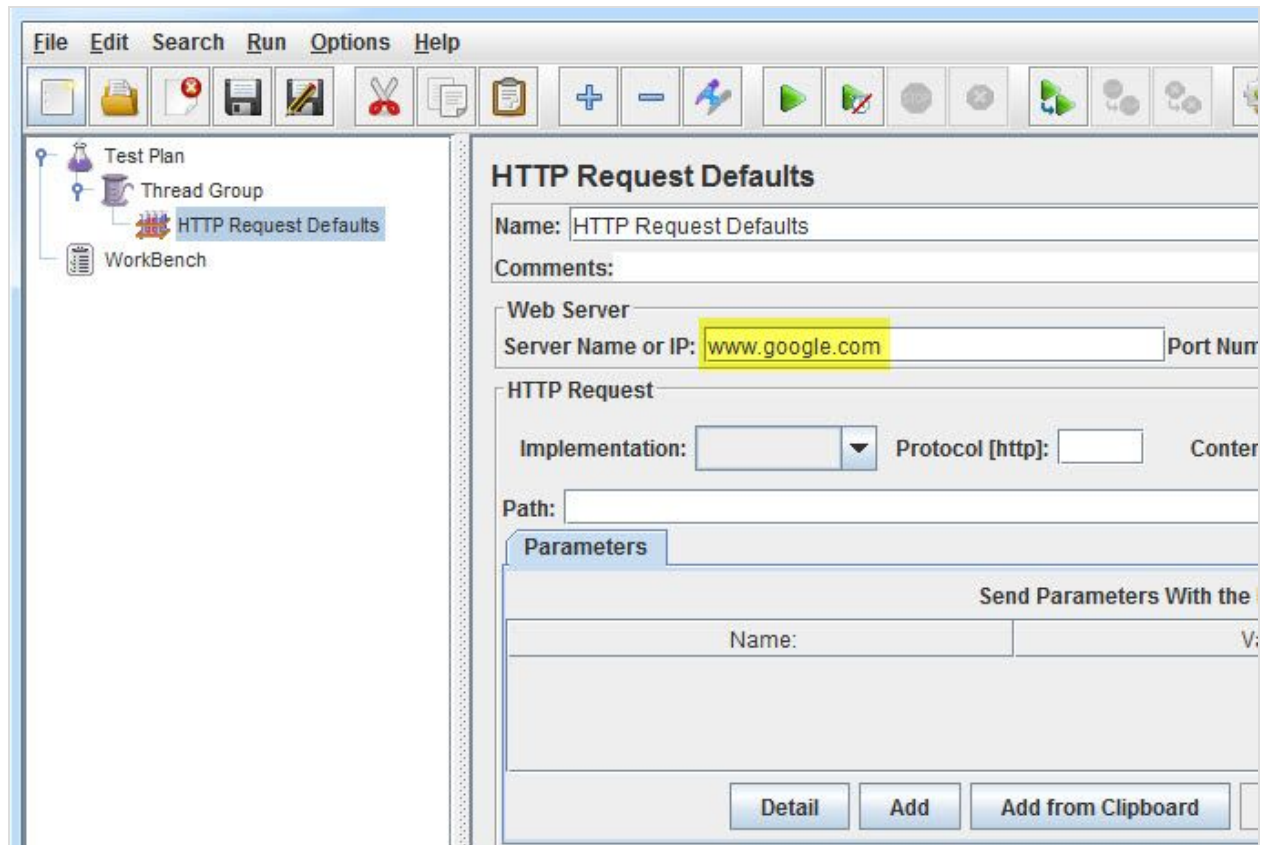
- **HTTP request Default element.**

To get this element, go to Thread Group and right-clicking, from context menu select

Add -> Config Element -> HTTP Request Defaults.



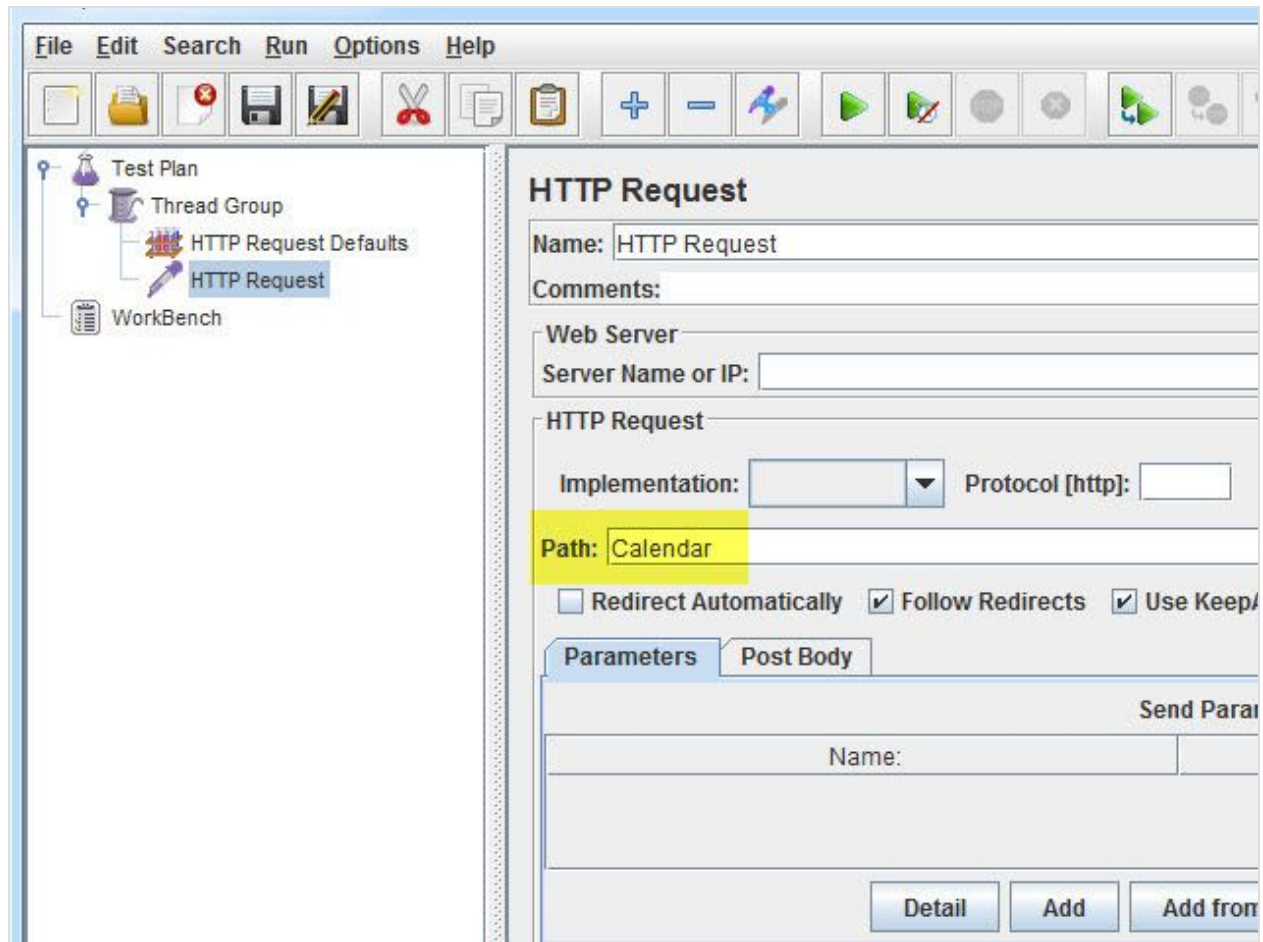
In the HTTP Request Defaults page, enter the Website name(www.google.com) under Web server à Server name or IP.



- **HTTP Request**

To get this element, go to Thread Group and right-clicking, from context menu select

Add -> Sampler -> HTTP Request.



"Path" field in HTTP Request pages how's that which **URL request** you need to direct to Google server like: enter "*calendar*" text in Path field. JMeter will generate the URL request <http://www.google.com/calendar> to Google server. If you leave the Path field blank, will URL request will be <http://www.google.com>.

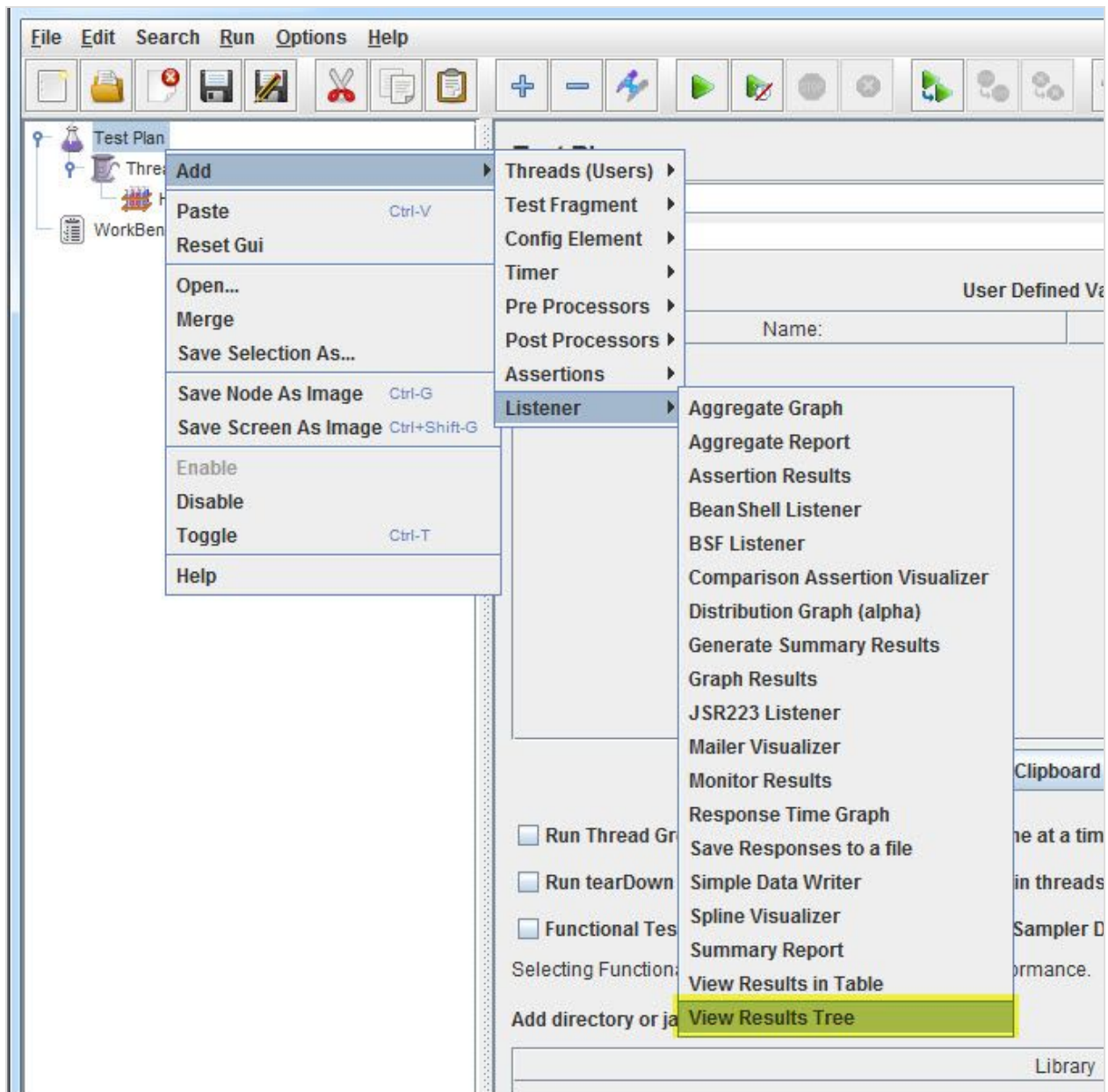
Step 3: JMeter Test Result in Graphical Format

A) View Results Tree:

View Result Tree is an most commonly used listener to get the full detail along with response time, and response code. Most of the performance tester is use this listener while doing performance testing.

To add View Results Tree listener, right click on Test Plan and click on

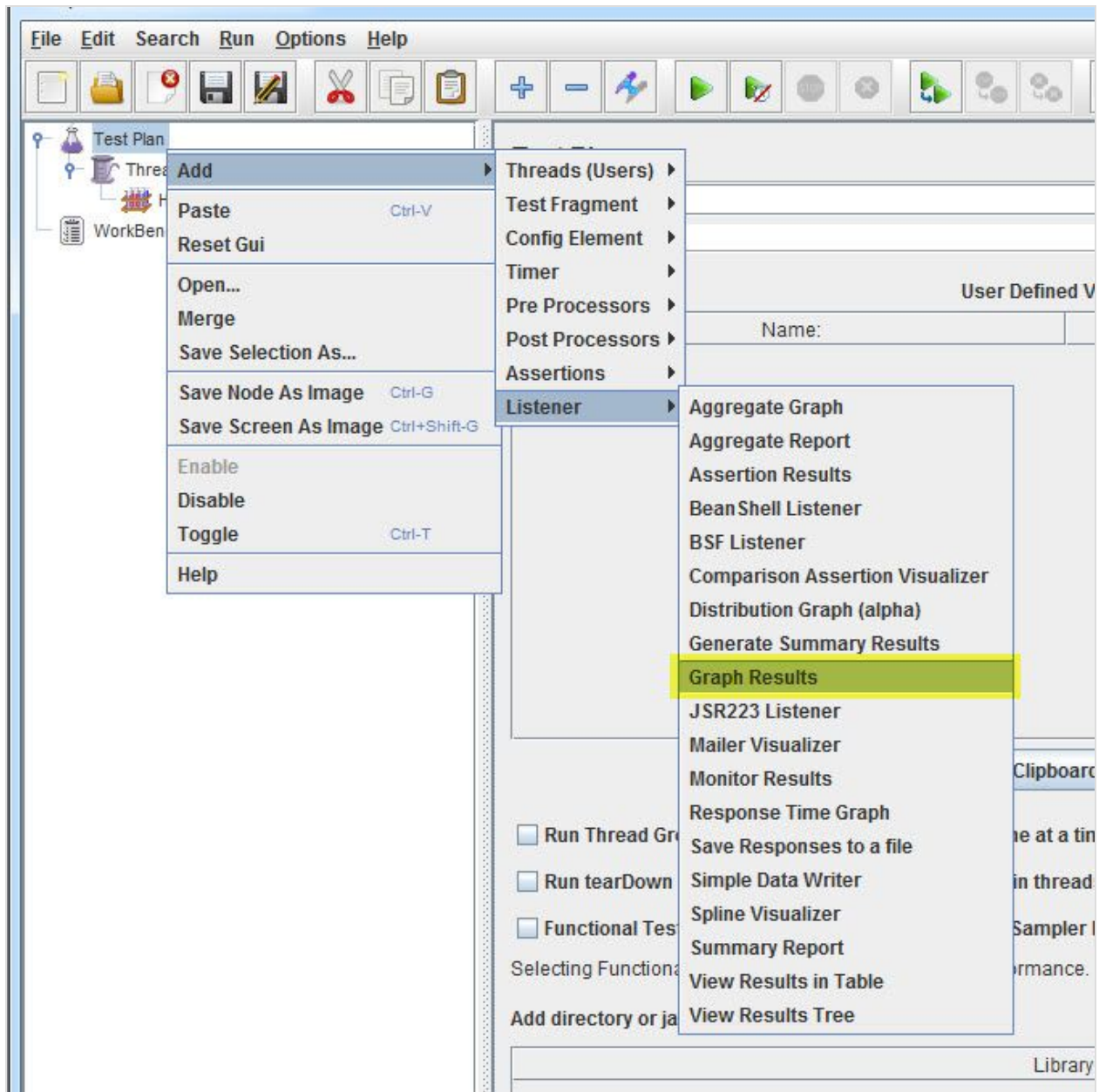
Add -> Listener -> View Results Tree.



B) Graph Results:

JMeter test result can also be shown in Graphical format. To do that, right click on Test Plan and click on

Add -> Listener -> Graph Results.

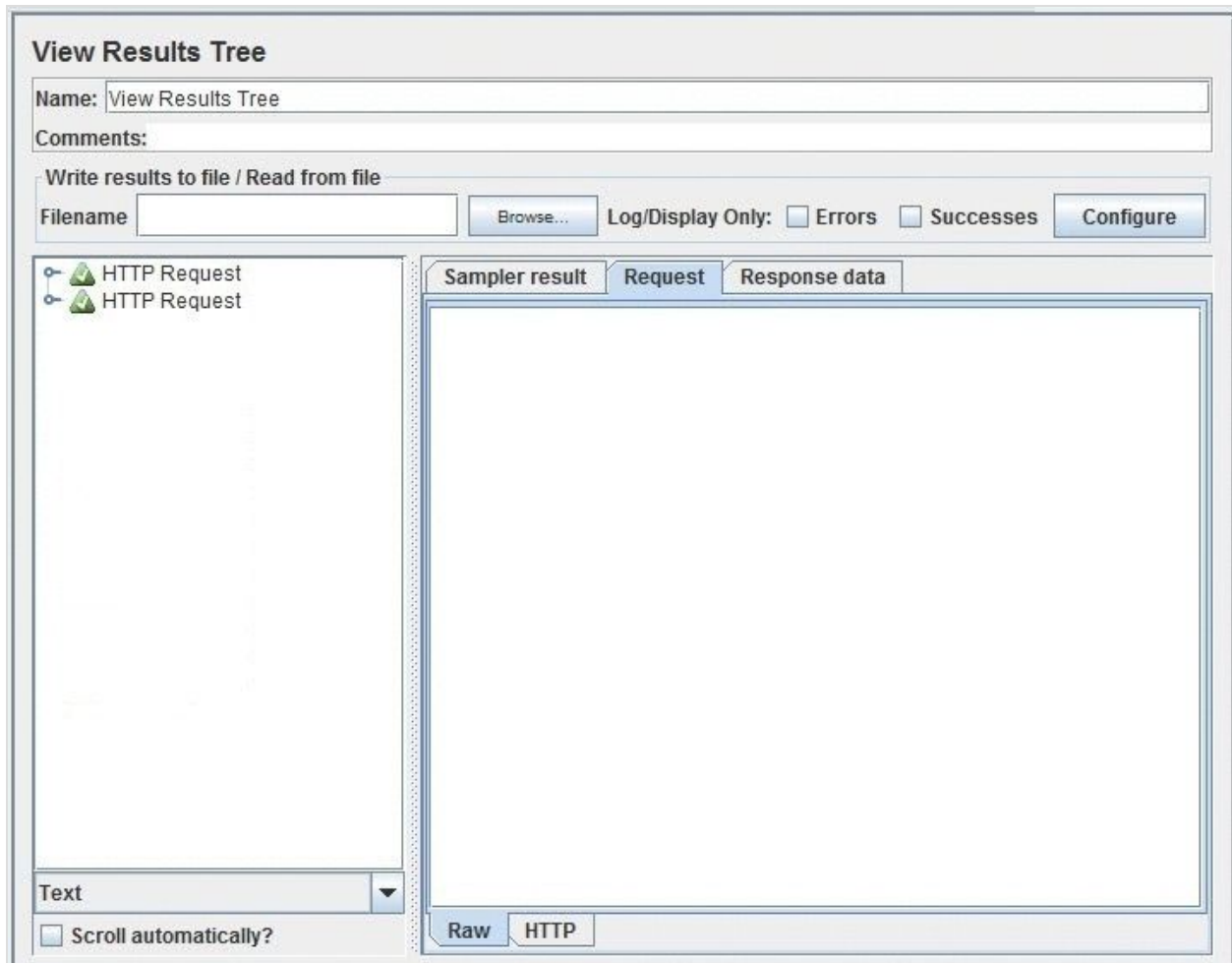


Step 4: Get the test result in Graph

In JMeter interface, click on **Run** button or press Ctrl + R on Toolbar to execute the test. It will display the real time test result in Graph.

The figure shows the View result tree & graph of a test plan, where 100 users are simulated and access the website www.google.com.

View Result Tree



Graph Results

Graph Results *Step 1*

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only:

☐

Errors

☐

Successes

Configure

Graphs to Display

☒

Data

☒

Average

☒

Median

☒

Deviation

☒

Throughput

1228 ms

0 ms

No of Samples 1000

Deviation 221

Latest Sample 813

Throughput 556.308/minute

Average 891

Median 817

Timers in JMeter -

By default, JMeter sends request without stopping anywhere between each request.

Generally, user spends lots of time while browsing i.e. before submitting confidential form or reading important document. So, it is a good practice to involve some delay between requests. Here, at very short period of times, JMeter can overwhelm your test server by creating various requests. For example: To test web server, send thousands requests in few seconds and face overload issue.

Using Timers, JMeter delays each request while sending which is made by thread. Because of this reason, Timer resolves the server overload issue.

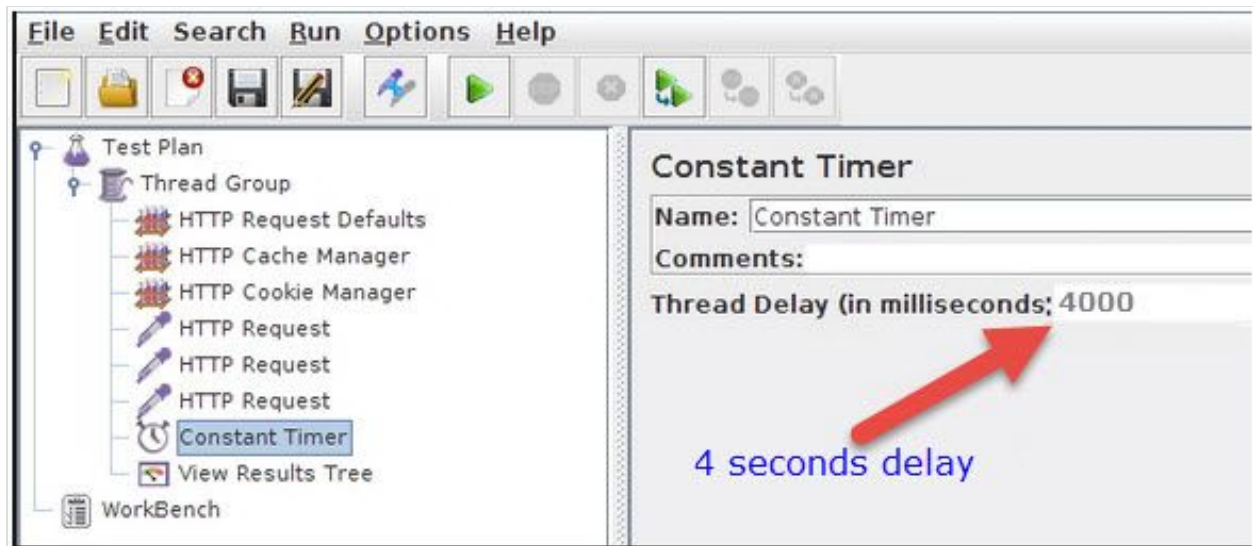
When you add more than one timer to a Thread Group, JMeter keeps the amount of the timers and pauses for that amount of time before executing the samplers to which the timers apply. Timers can be added as children of samplers or controllers in order to limit the samplers to which they are participated.

Types of Timers

- Constant Timer
- Constant Throughput Timer
- Uniform Random Timer
- Gaussian Random Timer
- BeanShell Timer
- Poisson Random Timer
- BSF Timer
- JSR223 Timer
- Synchronizing Timer

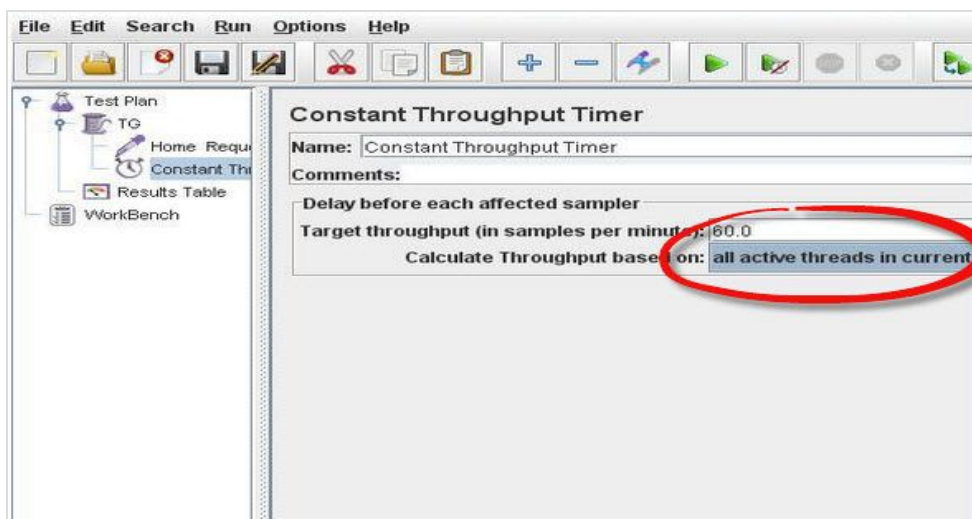
Constant Timer:

To delay every user request for the same amount of time use Constant Timer.



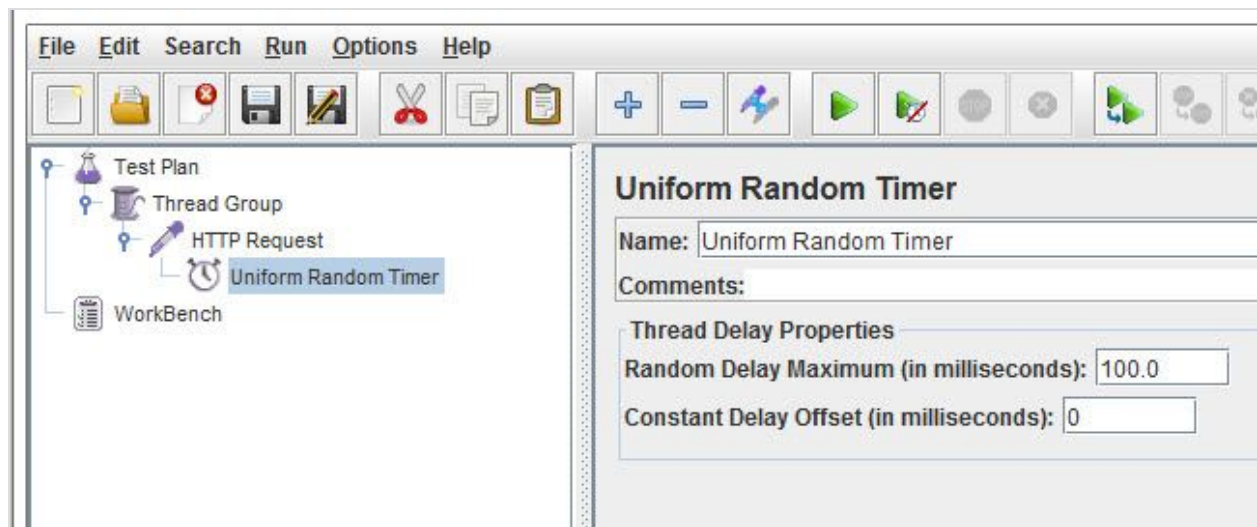
Constant Throughput Timer:

Constant Throughput Timer presents random delays between requests in a manner that a load/stress of necessary throughput is sent to the application. Figure shows, Constant Throughput Timer for 60 requests per minute.



Uniform Random Timer:

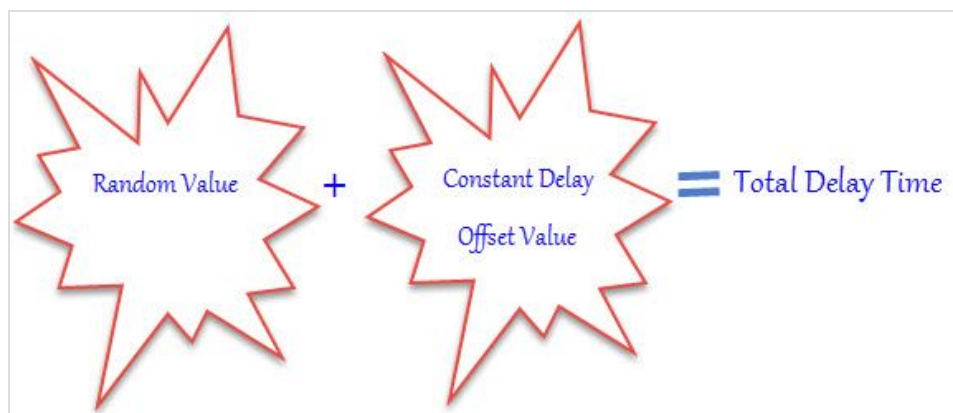
To delay every user request for random amount of time use Uniform Random Timer with every time interval having the same possibility of occurring.



In the above figure,

Random Delay Maximum Value: Maximum random number of milliseconds to delay

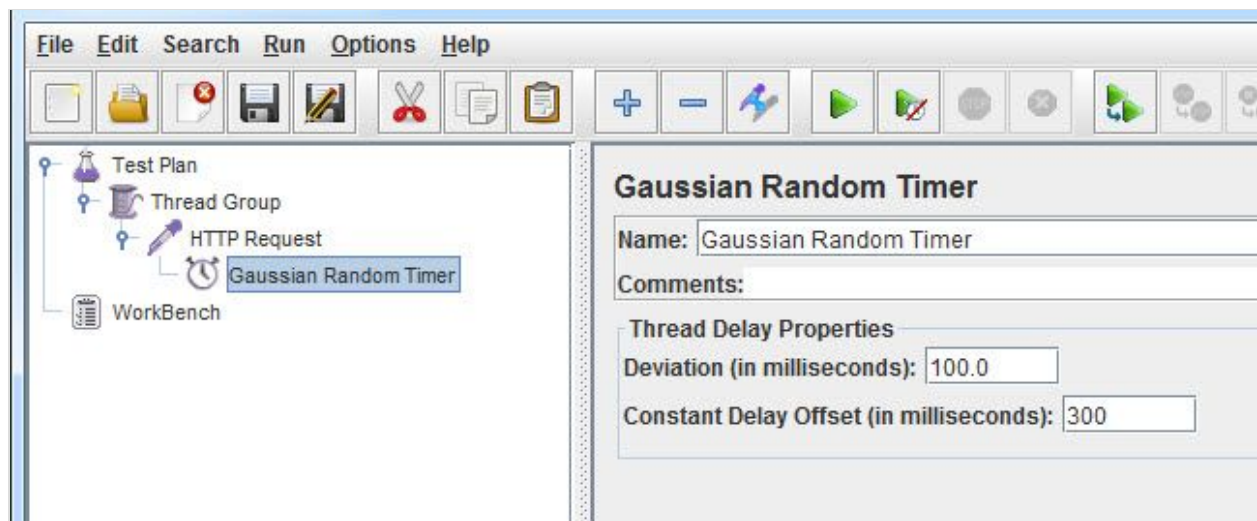
Constant Delay Offset Value: Additional value



Gaussian Random Timer:

To delay every user request for random amount of time use Gaussian Random Timer with most of the time intervals happening near a specific value.

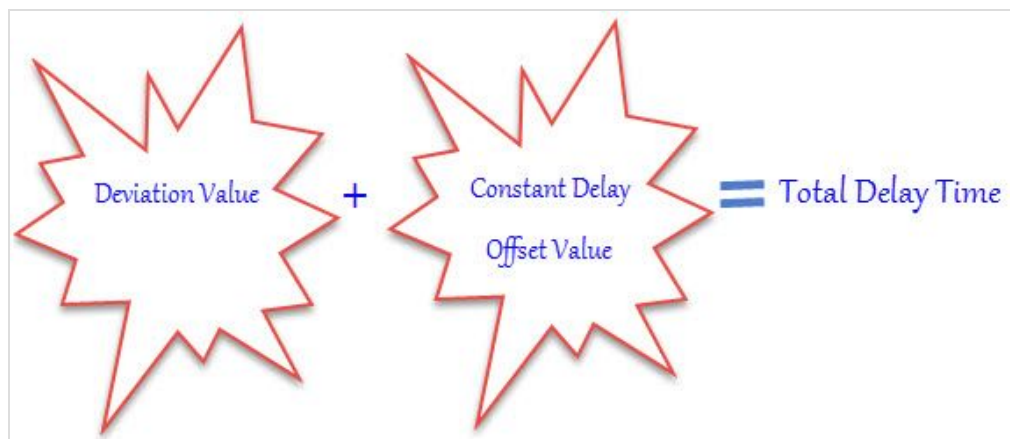
Gaussian Random means with most of the time intervals arising near a specific value [variance in any amount](#) [...](#) i.e. constant interval & varying between constant interval + deviation.



In the above figure,

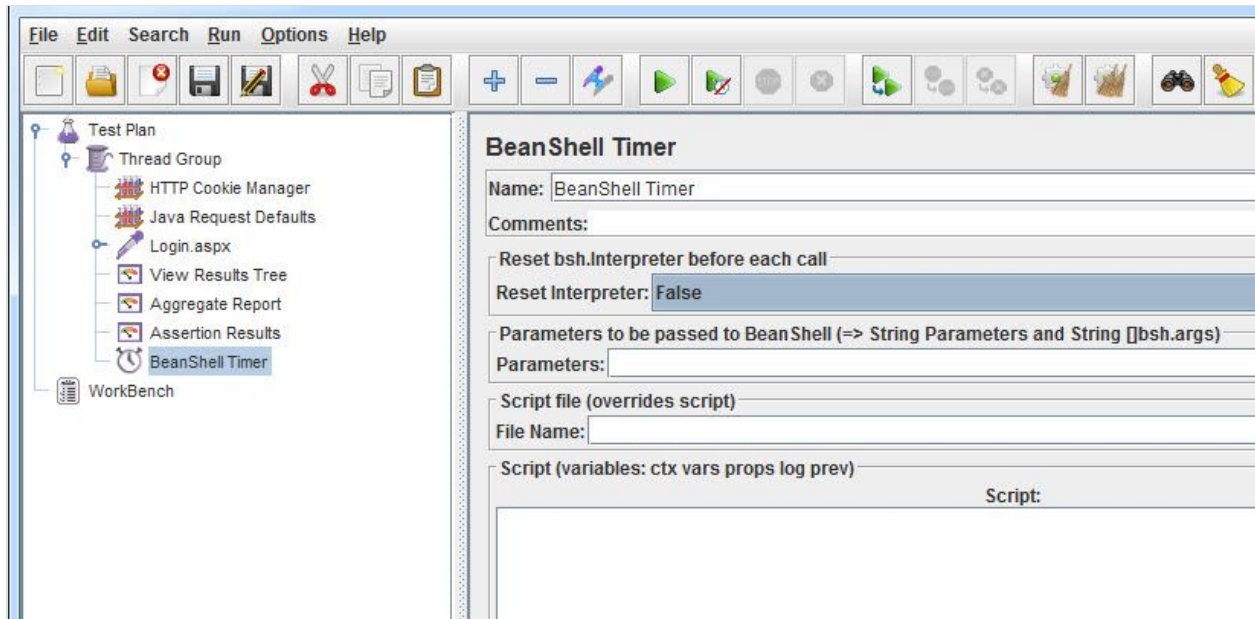
Deviations (milliseconds): A parameter of Gaussian Distribution Function

Constant Delay Offset (milliseconds): Additional value



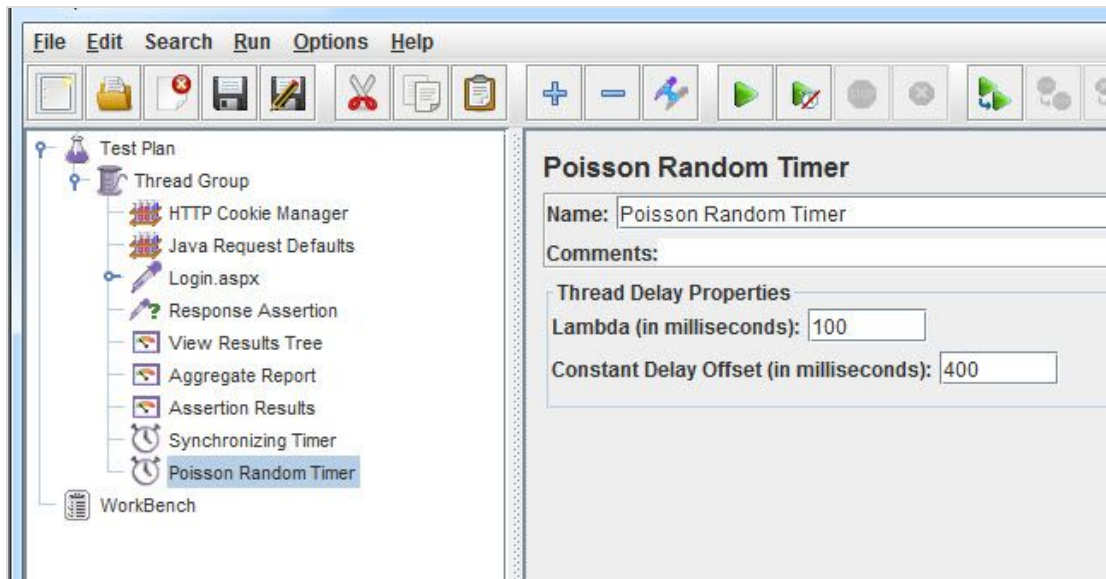
BeanShell Timer:

Between each user request, the [BeanShell](#) Timer can be used to create a delay time. (We will see each timer in detailed in upcoming articles with example.)

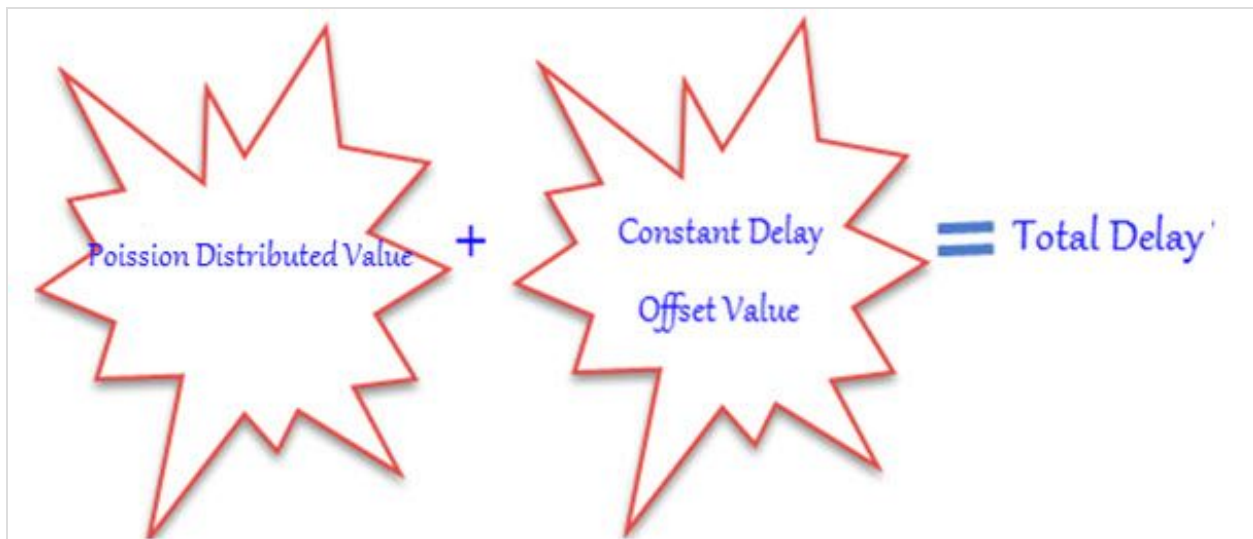


Poisson Random Timer:

To pause each and every thread request for random amount of time use Poisson Random Timer with most of the time intervals occurring close a specific value.



Sum of **Poisson Distributed Value** and **Offset Value** generate **Total Delay** value.



BSF Timer:

Between each user request, the BSF Timer can be used to **create** a delay time using [BSF](#) scripting language.

BSF Timer

Name:

Comments:

Script language (e.g. beanshell, javascript, jexl)

Language:

Parameters to be passed to script (=> String Parameters and String []args)

Parameters:

Script file (overrides script)

File Name:

Script (variables: ctx vars props sampler log Label Filename Parameters args[] OUT)

Script:

```
1
```

JSR223 Timer:

Between each user request, the JSR223 Timer can be used to **create** a delay time using a [JSR223](#) scripting language.

Synchronizing Timer:

Synchronizing Timer is used to synchronize requests of various threads, means; it will increase delays between requests. So that, all threads fire at the same time there for regenerating heavy load bursts on your application. It is same like Rendezvous Point in LoadRunner and create big load at various places in JMeter plan.

Assertions in JMeter -

In previous article we have seen [What all types of Timers supported in JMeter](#) & what is meaning of each timer? However if you adding timer for reason where expected to load some value with the specified time & specified value is getting properly as per mentioned time then timer will work for your test cases. But sometimes due some reason the expected value is not received with specified time frame then your test cases execution will fail. Under such circumstances you should think of using different types of Assertions in JMeter.

Assertion or verification plays an important role in testing. It helps to verify and ensure that testing process is going in a right direction and the server which is under test returns the expected results. For example; when user sends requests to the server and wait for the response, after a long wait when user gets response then how user will be ensure that the coming response from the server is the correct one. To resolve this issue, use assertion or verification to ensure that the response comes from server at run time is correct, if the response will be not correct then test will get fail for that request.

These are some types of Assertion used in JMeter,

- Response Assertion
- Duration Assertion
- Size Assertion
- XML Assertion
- HTML Assertion

Response Assertion

The Response assertion facilitates users to add pattern strings to be compared against several areas of the server response.

For example, if somebody sends request to the website <http://www.Google.com> and wait for the server response. In this case, Response Assertion role is to verify that the server response has probable pattern string, e.g. "OK" or not.

Response Assertion

Name: Response Assertion

Comments:

Apply to:

☒ Main sample only ☐ Sub-samples only ☐ Main sample and sub-samples ☐ JMeter Variable

Response Field to Test

☒ Text Response ☐ URL Sampled ☐ Response Code ☐ Response Message ☐ Response Headers ☐ Ignore Status

Pattern Matching Rules

☒ Contains ☐ Matches ☐ Equals ☐ Substring ☐ Not

Patterns to Test

Patterns to Test
(?is)<head_<title>.*</title></head>
<!--no error on page -->

Add Delete

These are the server response field used by JMeter to test

Duration Assertion

The role Duration Assertion in JMeter to test that each response comes from server has received within a **given amount** of time or not. If any server response takes longer time than the given number of milliseconds specified by the user then the response gets fail.

Duration Assertion

Name: Duration Assertion - 1500ms

Comments:

Apply to:

☒ Main sample only ☐ Sub-samples only ☐ Main sample and sub-samples

Duration to Assert

Duration in milliseconds: 1500

User Specified Time

For example, , if somebody sends request to the website <http://www.Google.com> by JMeter and receives a response within **limited** time 5 ms then test case pass, else test case fails.

Size Assertion

The role of Size Assertion in JMeter to test that each response comes from server holds the expected number of byte. Size Assertion facilitates users to specify the size i.e, equal to, greater than, less than or not equal to a given number of bytes.

For example, , if somebody sends request to the website <http://www.Google.com> by JMeter and receives response packet with less than **expected** byte 5000 bytes in size then a test case pass, else a test case fails.

The screenshot shows the 'Size Assertion' configuration window in JMeter. It includes fields for 'Name' (set to 'Size Assertion') and 'Comments'. The 'Apply to' section has radio buttons for 'Main sample only' (selected), 'Sub-samples only', 'Main sample and sub-samples', and 'JMeter Variable' with an adjacent text box. The 'Response Size Field to Test' section has radio buttons for 'Full Response' (selected), 'Response Headers', 'Response Body', 'Response Code', and 'Response Message'. The 'Size to Assert' section contains a 'Size in bytes' text box with the value '5000' and a 'Type of Comparison' list with radio buttons for '=', '!=', '>', '<', '>=', and '<='.

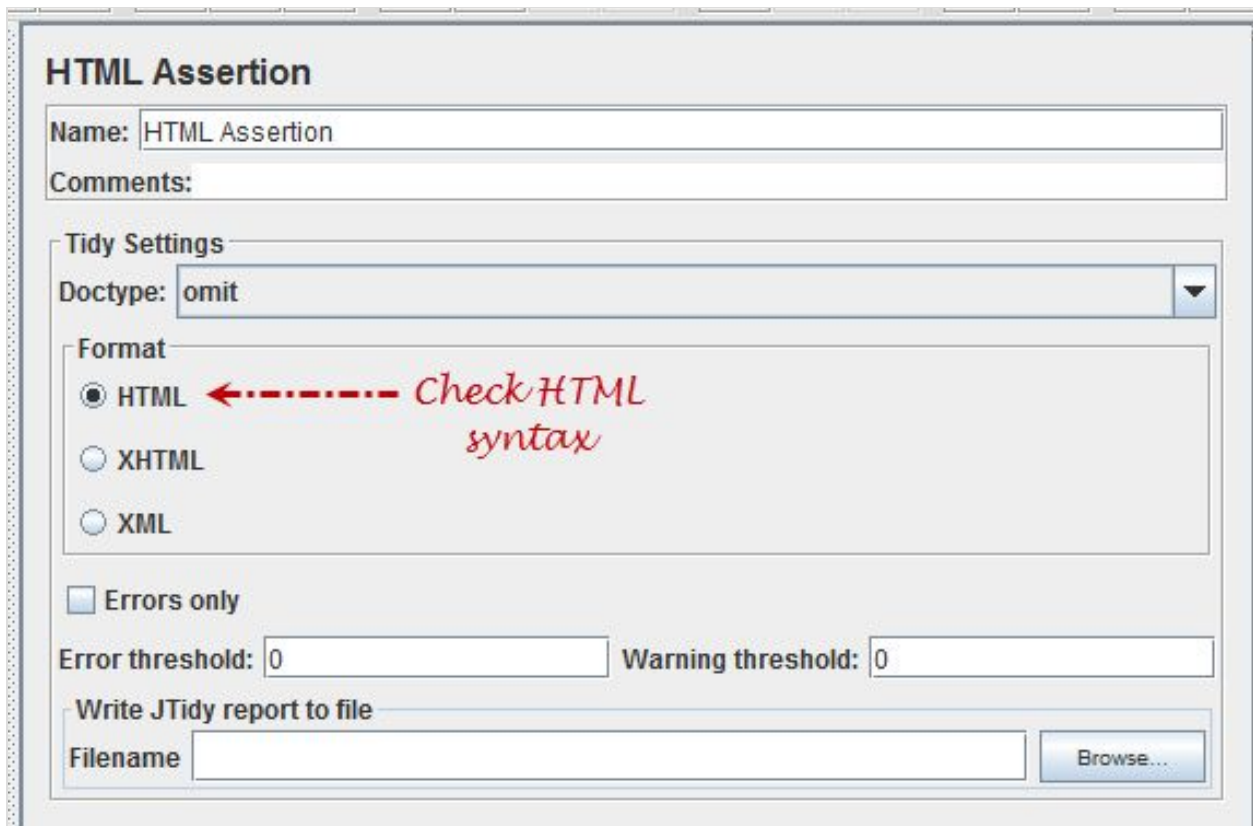
Size Assertion	
Name:	Size Assertion
Comments:	
Apply to:	
<input checked="" type="radio"/> Main sample only	<input type="radio"/> Sub-samples only
<input type="radio"/> Main sample and sub-samples	<input type="radio"/> JMeter Variable <input type="text"/>
Response Size Field to Test	
<input checked="" type="radio"/> Full Response	<input type="radio"/> Response Headers
<input type="radio"/> Response Body	<input type="radio"/> Response Code
<input type="radio"/> Response Message	
Size to Assert	
Size in bytes: <input type="text" value="5000"/>	Type of Comparison
	<input checked="" type="radio"/> =
	<input type="radio"/> !=
	<input type="radio"/> >
	<input type="radio"/> <
	<input type="radio"/> >=
	<input type="radio"/> <=

XML Assertion

The role XML Assertion in JMeter to test that each response comes from server holds the data in correct XML document.

HTML Assertion

The role of HTML Assertion in JMeter is to permit users to check the HTML syntax of the response data. That means, the response data should contain the HTML syntax.



The screenshot shows the 'HTML Assertion' configuration window. It includes a 'Name' field set to 'HTML Assertion' and an empty 'Comments' field. Under 'Tidy Settings', the 'Doctype' is set to 'omit'. In the 'Format' section, the 'HTML' radio button is selected, with a red dashed arrow pointing to it and the handwritten text 'Check HTML syntax'. Below this, there are checkboxes for 'Errors only', 'Error threshold' (set to 0), and 'Warning threshold' (set to 0). At the bottom, there is a section 'Write JTidy report to file' with a 'Filename' field and a 'Browse...' button.

How to use Assertion?

Here we will take an example of **Response Assertion** to test and compare the response packet comes from the server "<http://www.Google.com>" matches your expected string.

Also, the response assertion facilitates users to add pattern strings to be compared against various fields of the response sends by server.

To test this, we will proceed in the following manner,

First: Add Response Assertion

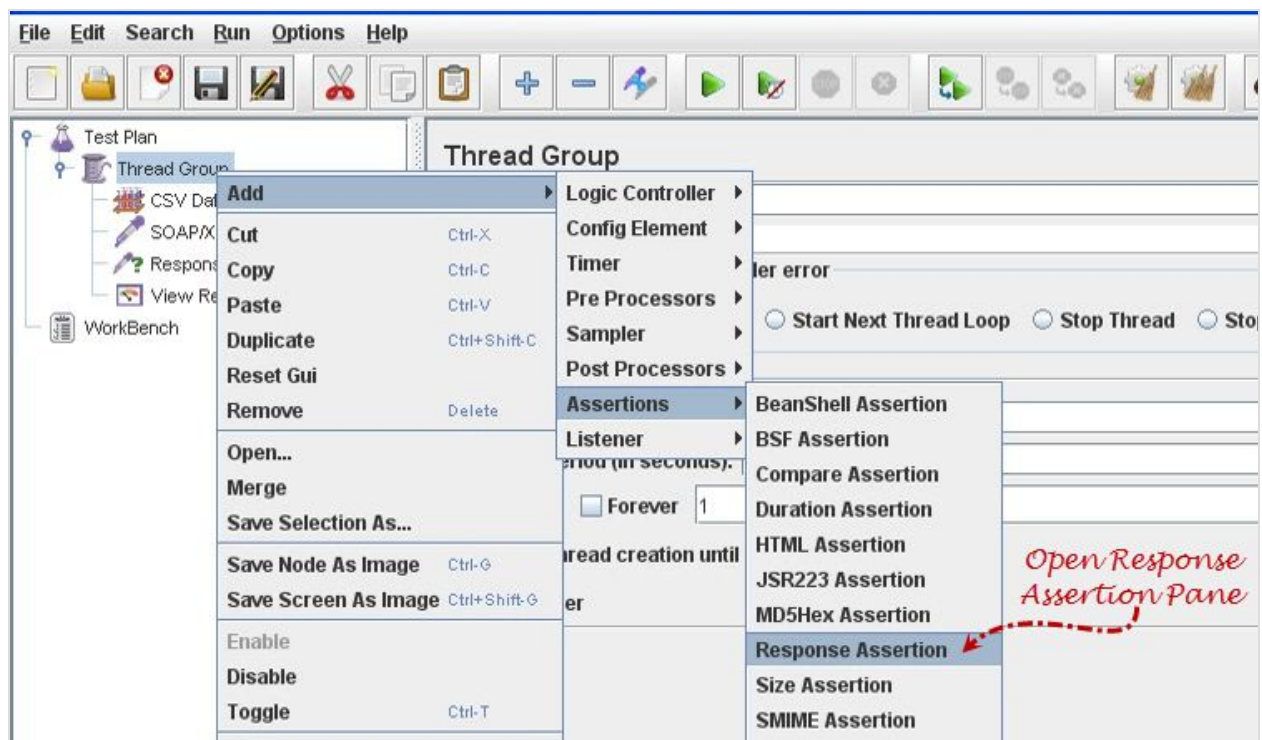
Second: Add Pattern

Third: Add Response Assertion result

Fourth: Run Test Result

First: Add Response Assertion

After opening JMeter interface, right-Click on **Thread Group** -> **Add** -> **Assertions** -> **Response Assertion**



Given below figure displays the Response Assertion Pane,

The screenshot shows the 'Response Assertion' configuration window in JMeter. It includes fields for 'Name' (set to 'Response Assertion') and 'Comments'. The 'Apply to' section has radio buttons for 'Main sample only' (selected), 'Sub-samples only', 'Main sample and sub-samples', and 'JMeter Variable' with an adjacent text box. The 'Response Field to Test' section has radio buttons for 'Text Response' (selected), 'URL Sampled', 'Response Code', 'Response Message', 'Response Headers', and a checked 'Ignore Status' checkbox. The 'Pattern Matching Rules' section has radio buttons for 'Contains' (selected), 'Matches', 'Equals', 'Substring', and a checked 'Not' checkbox. The 'Patterns to Test' section contains a table with two rows of patterns: '(?is)<head><title>.*</title></head>' and '<!-- no errors on page -->'. Below the table are 'Add' and 'Delete' buttons.

Patterns to Test
(?is)<head><title>.*</title></head>
<!-- no errors on page -->

Second: Add Pattern

When user sends request to the Google server, it sends response with some response code, these codes explain the status of the server, some common response codes are,

200 – The server successfully sent the response

302–Generally it happens that, when user access www.google.com from outside USA then the Web server redirect to other page. That means; URL of the Google re-directs to country specific website page URL. As shown below, www.google.com redirects to www.google.co.in for Indian Users.

404 – Server error because request doesn't exist

503 – The server is temporarily unavailable

Let us verify the responses code pattern **302** of the Google.com web server,

To check the responses code pattern **302**, In **Response Assertion** pane, go to "**Response Field To Test**" and choose "Response Code", then click on "Add", a new text box displays below the "Pattern to Test" field. In "Pattern to Test" text box enter 302.

The screenshot shows the 'Response Assertion' configuration window in JMeter. It includes fields for Name, Comments, and Apply to. The 'Response Field to Test' section has radio buttons for Text Response, URL Sampled, Response Code, Response Message, Response Headers, and Ignore Status. The 'Pattern Matching Rules' section has radio buttons for Contains, Matches, Equals, Substring, and Not. The 'Patterns to Test' section has a table with one row containing the value '302'. Below the table are 'Add' and 'Delete' buttons. Handwritten red annotations provide a three-step guide: Step 1 points to the 'Response Code' radio button; Step 2 points to the 'Add' button; Step 3 points to the '302' entry in the 'Patterns to Test' table.

Response Assertion

Name: Response Assertion

Comments:

Apply to:

☒ Main sample only ☐ Sub-samples only ☐ Main sample and sub-samples ☐ JMeter Variable

Response Field to Test

☒ Text Response ☐ URL Sampled ☐ Response Code ☐ Response Message ☐ Response Headers ☐ Ignore Status

Pattern Matching Rules

☒ Contains ☐ Matches ☐ Equals ☐ Substring ☐ Not

Patterns to Test

Patterns to Test
302

Add Delete

Step 1: Check the response code

Step 2: Click on Add button.

Step 3: Add pattern to check the response code.

Third: Add Response Assertion result

From left side on the pane of Thread Group, right click on Thread Group, and from the context menu click on **Add -> Listener -> Assertion Results**.

Response Assertion result pane will display like this,

Assertion Results

Name:

Comments:

Write results to file / Read from file

Filename Log/Display Only: ☐ Errors ☐ Successes

Assertions:

Assertion Results pane

Fourth: Run Test Result

After opening **Assertion Result pane**, click on **Run** button from the menu bar, or from the keyboard press **Ctrl+R** will display the test result on Assertion Results pane. If Google server **response code** pattern is **302**, the test case is passed and the result window displays shown in the figure below

Assertion Results

Name:

Comments:

Write results to file / Read from file

Filename Log/Display Only: ☐ Errors ☐ Successes

Assertions:

HTTP Request
HTTP Request
HTTP Request
HTTP Request
HTTP Request
HTTP Request
HTTP Request
HTTP Request

Assertion Results pane

Uses of Controllers in JMeter -

In previous Day 7 article from JMeter Tutorials Series we have seen different [Assertions used in JMeter](#). Today we are concentrating on list of supported Controllers in JMeter.

JMeter in-built function controller is basically a Logic Controller provides control on “when & how” to send a user request to a web server under test. Logic Controller having command of the order of the request sends to the server.

Logic Controllers

Logic Controllers facilitates users to describe the sequence of processing request sends to the server in a Thread like; JMeter’s Random Controllers can be used to send HTTP requests randomly to the server. Logic Controllers define the order in which user request are executed.

Some useful Logic Controllers comes in operation are,

- Recording Controller
- Simple Controller
- Loop Controller
- Random Controller
- Module Controller

Recording Controller

After recording of testing process happens in JMeter, Recording Controller role is to store those recordings.



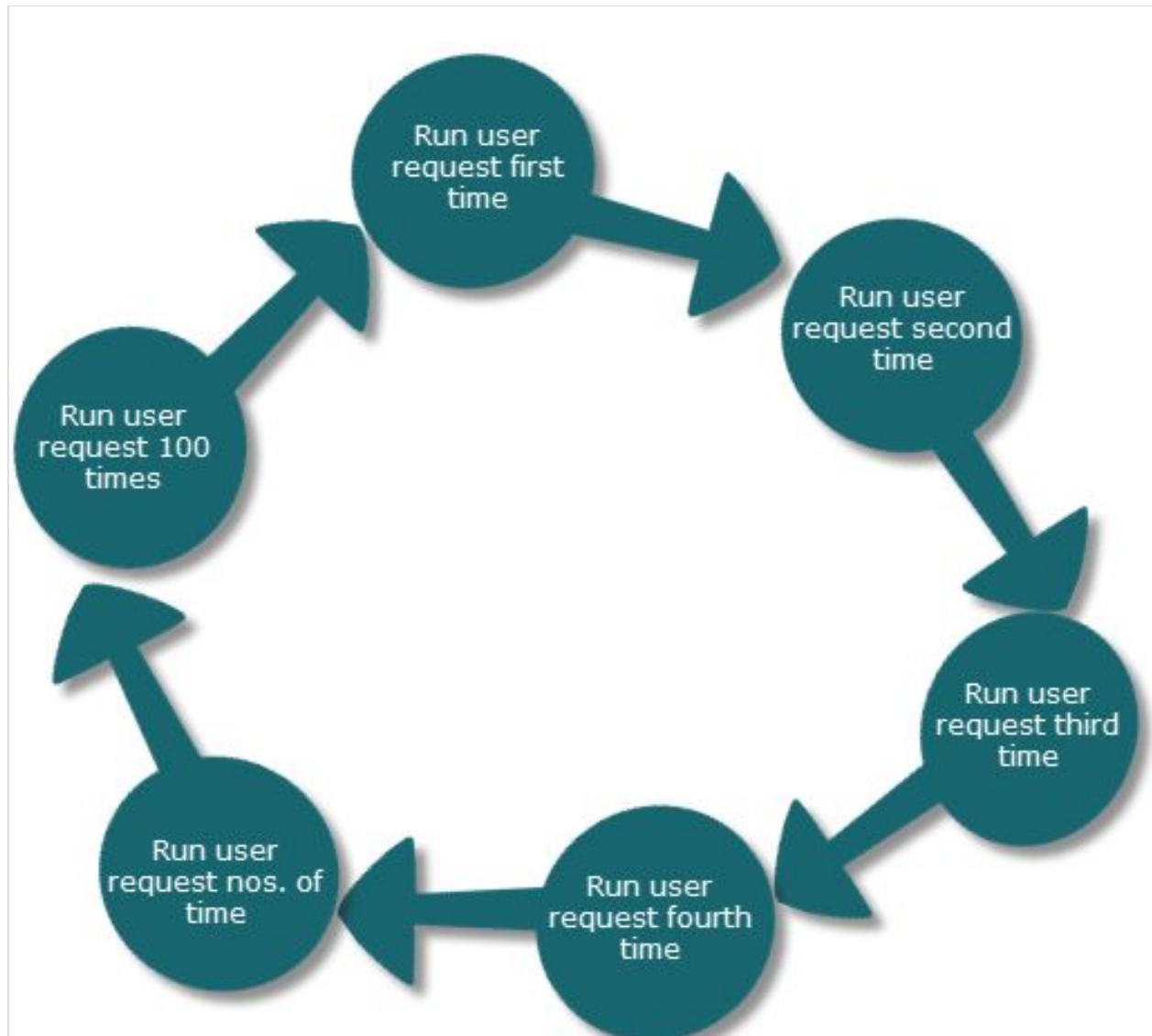
Simple Controller

Simple Controller is a container, holds user request. It does not provide any customization, randomization or change of loop count, etc as other logic controllers do.



Loop Controller

Loop Controller creates a situation where user request runs either; for limited **time** or **forever**, which is better explained in the figure,

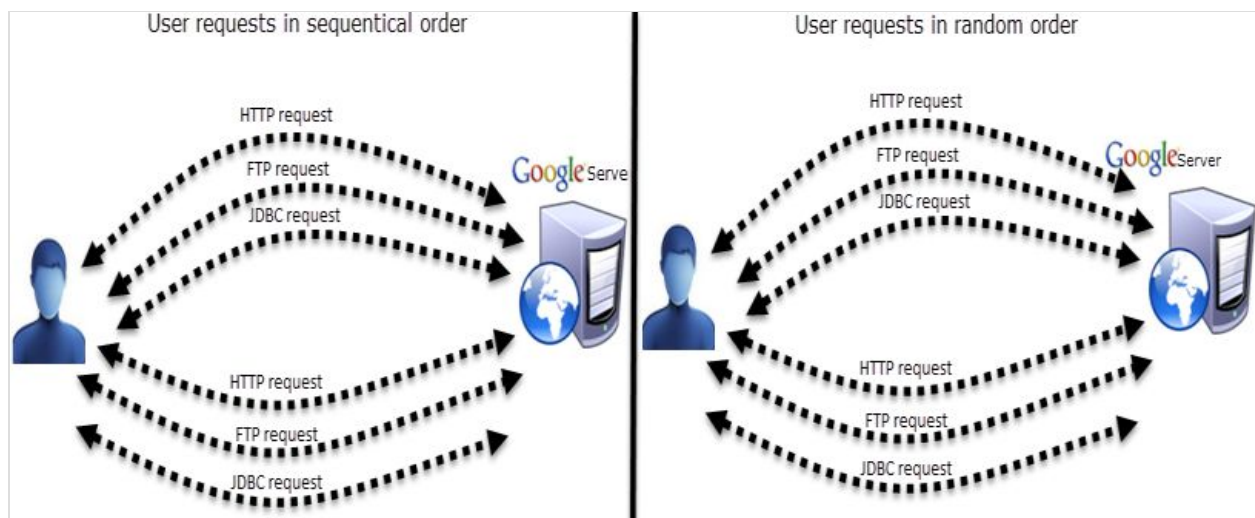


Random Controller

Random Controller creates a situation where user request runs in **random** order in each loop period. Suppose, the website <http://www.google.com> is having 3 user requests in following order,

1. HTTP request
2. FTP request
3. JDBC request

Each user request runs 5 times i.e, HTTP request runs 5 times, FTP request runs 5 times and JDBC request runs 5 times, means; total user requests send to the Google server is 15 = 5×3 by JMeter.



In each loop, user requests send **sequentially** in following order,

HTTP request -> FTP request-> JDBC request

In each loop, user requests send **randomly** in following order,

FTP request -> HTTP request-> JDBC request

Or

JDBC request -> FTP request-> HTTP request

Module Controller

Module Controller sets the modularity in each module of JMeter function stored in Simple Controller. For example; any web application consists of numbers of functions like; sign – in, sign – out, and account creation, password change, Simple Controller stores these functions as modules then Module Controller chooses which module needs to run. Suppose, simulation happens when 50 users sign-in, 100 users sign-out, and 30 users search on website www.google.com.



Using JMeter, user can create 3 modules where each module simulates each user activity: sign-in, sign-out, and Search. Module controller selects which module wants to run.

Some More Vital Controllers

Interleave Controller: Select only one user request which runs in one particular loop of thread.

Runtime Controller: Determines that till **how long** its children are permitted to run. For example; if user has detailed Runtime Controller 10 seconds then JMeter will run the test for 10 seconds.

Transaction Controller: Evaluates the **overall time** taken to **complete** a test execution

Include Controller: It is created to utilize an external test plan. This controller permits you to utilize multiple test plans in JMeter.

Brief Look on Loop Controller

Here you will get to know each and every step involved in adding **Loop Controller** to at present performance test plan.

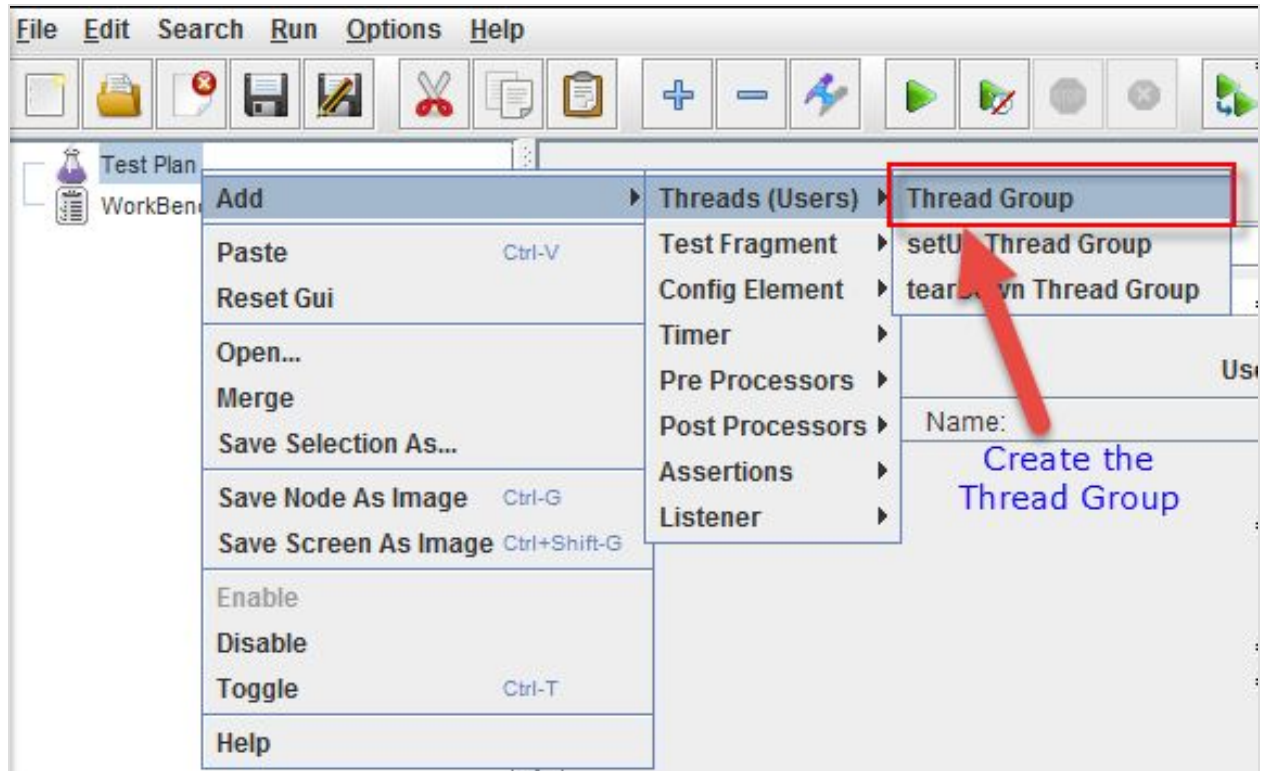
The Loop Controller helps samplers run at specific number of times with the loop value you definite for the Thread Group. Suppose user, add one HTTP Request to a Loop Controller with a loop count 50, configure the Thread Group loop count to 2, and then JMeter will send a total of $50 * 2 = 100$ HTTP Requests. These examples are explained below in better manner,

- Configuring Thread Group
- Adding Loop Controller
- Configuring Loop Controller
- Add View Result in Table
- Run the Test

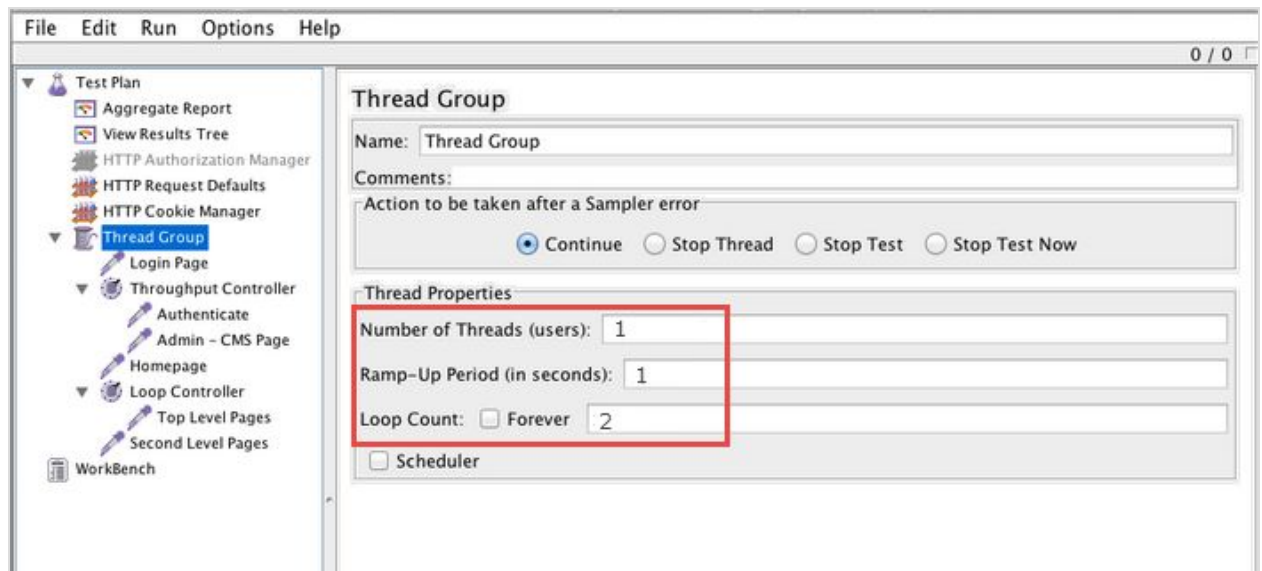
First Step: Configuring Thread Group

Add Thread Group:

To add the Thread Group, first run JMeter, from opened interface of JMeter choose Test Plan from the tree and right click to choose Add -> Threads (Users) -> Thread Group.



After opening thread Group, enter Thread Properties as shown in figure below,



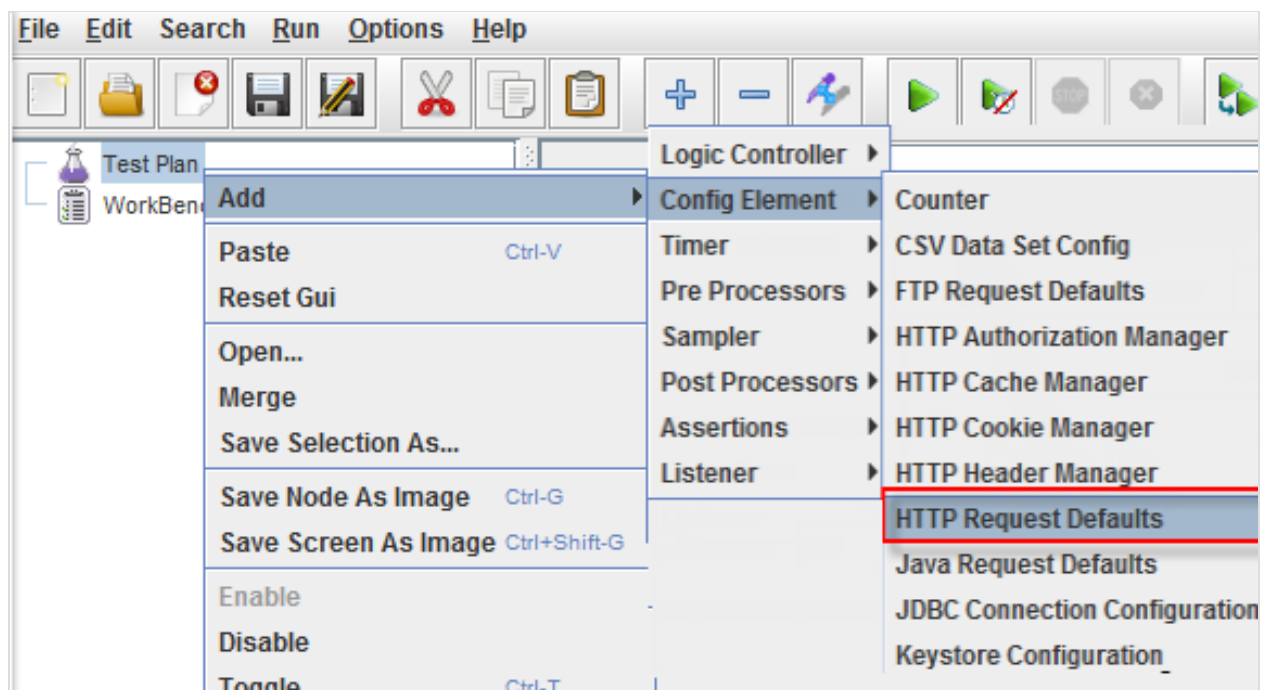
In the above figure, Number of Threads: 1 user is connected to target website, Loop Count: run it 2 times, and Ramp-Up Period: 1.

The Thread Count and The Loop Counts both are different. In the Thread Count, it simulates 1 user trying to connect the targeted website. In the Loop Counts, it simulates 1 user trying to connect the targeted website 2 times.

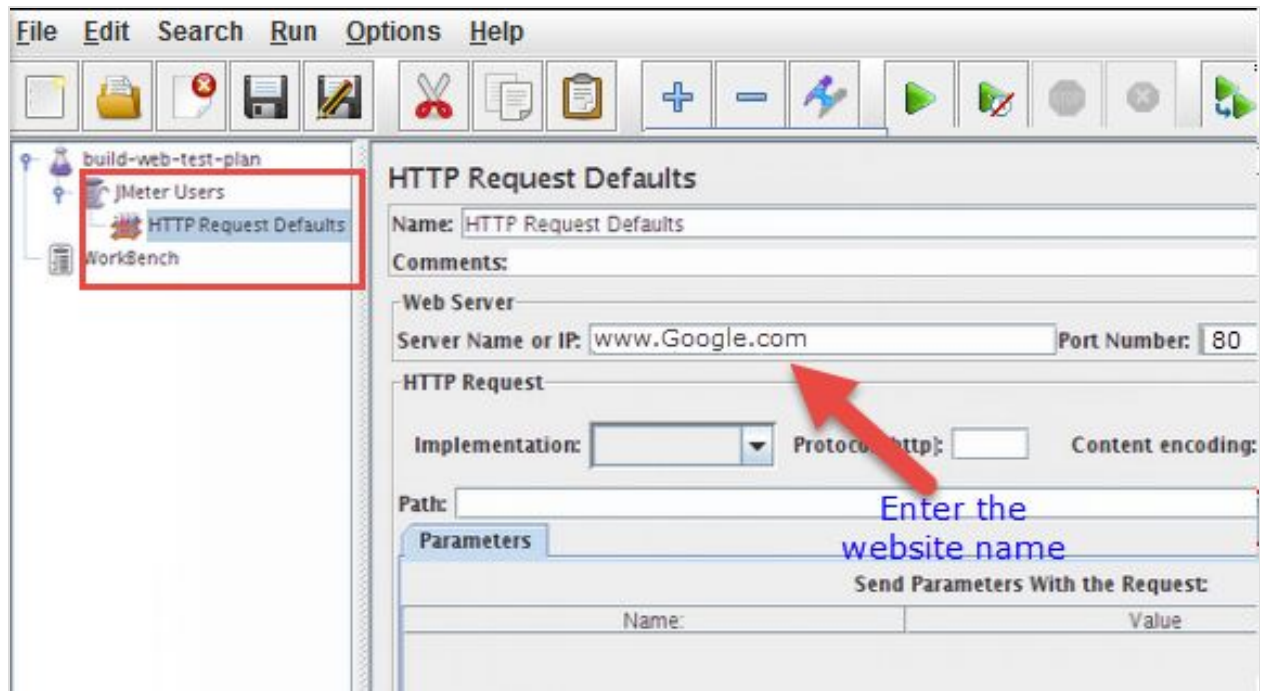
Add JMeter elements:

Here we will Add HTTP request Default" element.

To get this element, go to Thread Group and right-clicking, from context menu select **Add** -> **Config Element** -> **HTTP Request Defaults**.

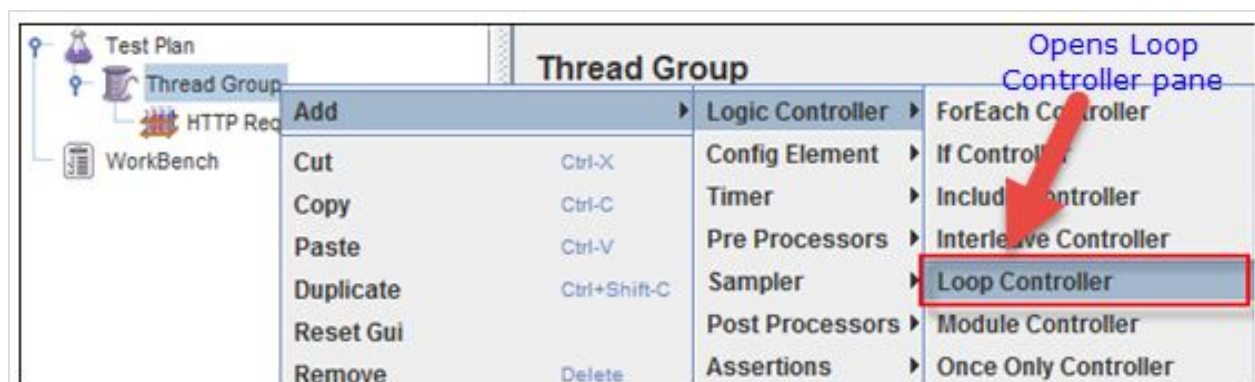


In the HTTP Request Defaults page, enter the Website name (www.google.com) under Web server -> Server name or IP.



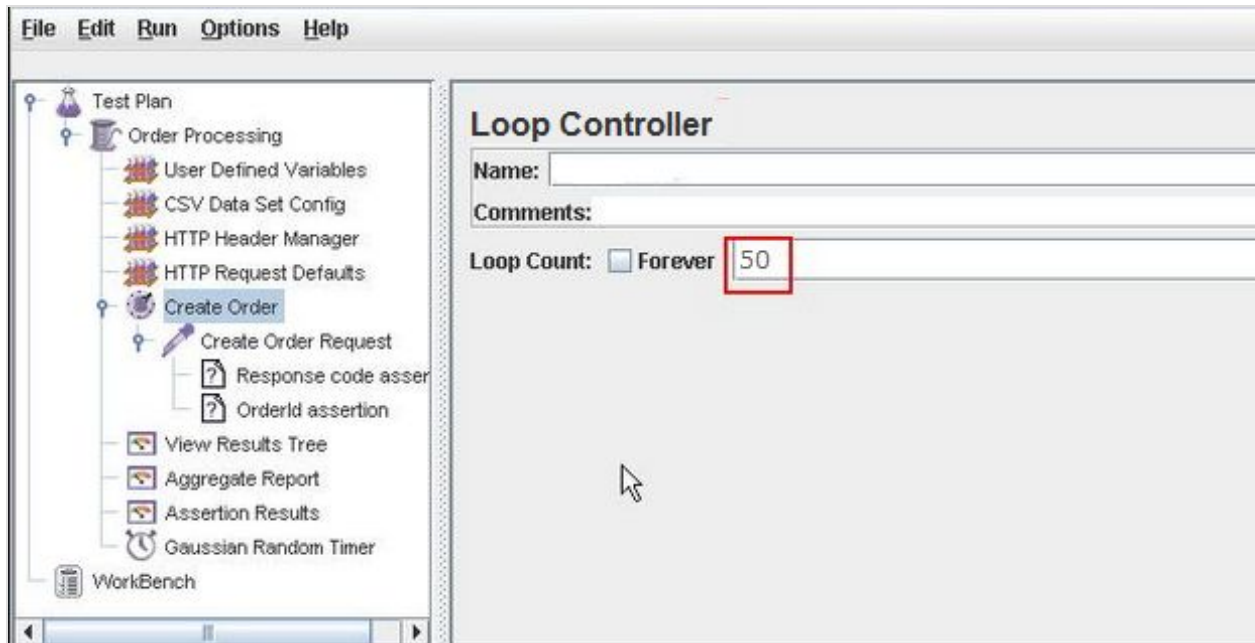
Adding Loop Controller:

To open the Loop Controller pane, right click on Thread Group -> Logic Controller -> Loop Controller.

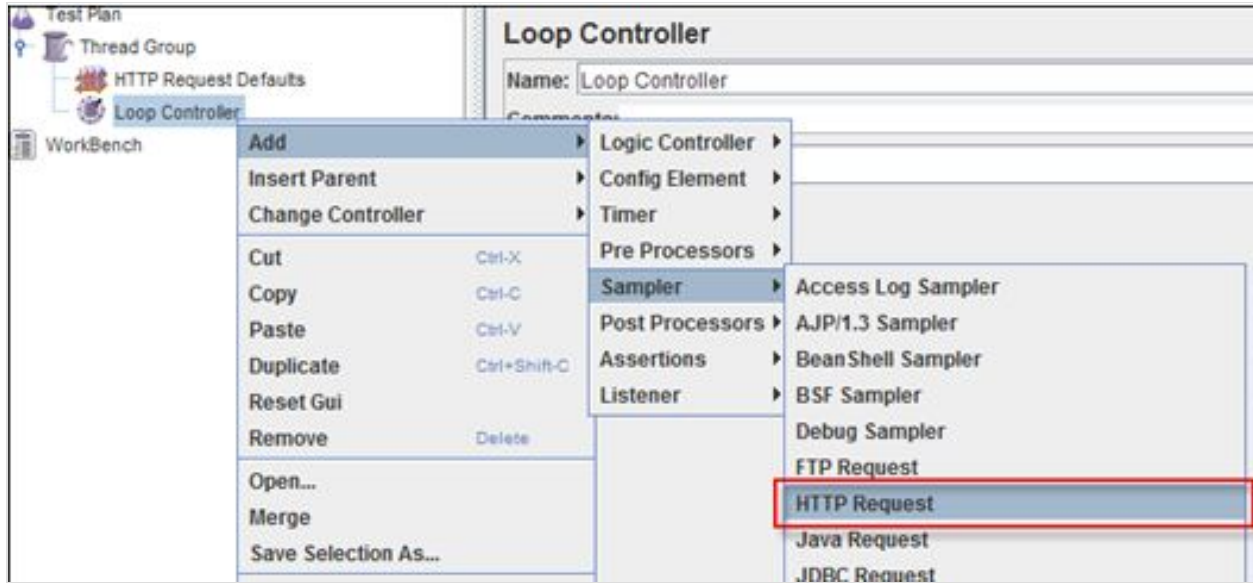


Second Step: Configuring Loop Controller:

Given below figure shows Loop Count: 50 which means one user request sends to the web server google.com run **50** times. Jmeter with take loop value =2 specified for Thread Group in the above figure to send the total HTTP Requests: $2 * 50 = 100$.

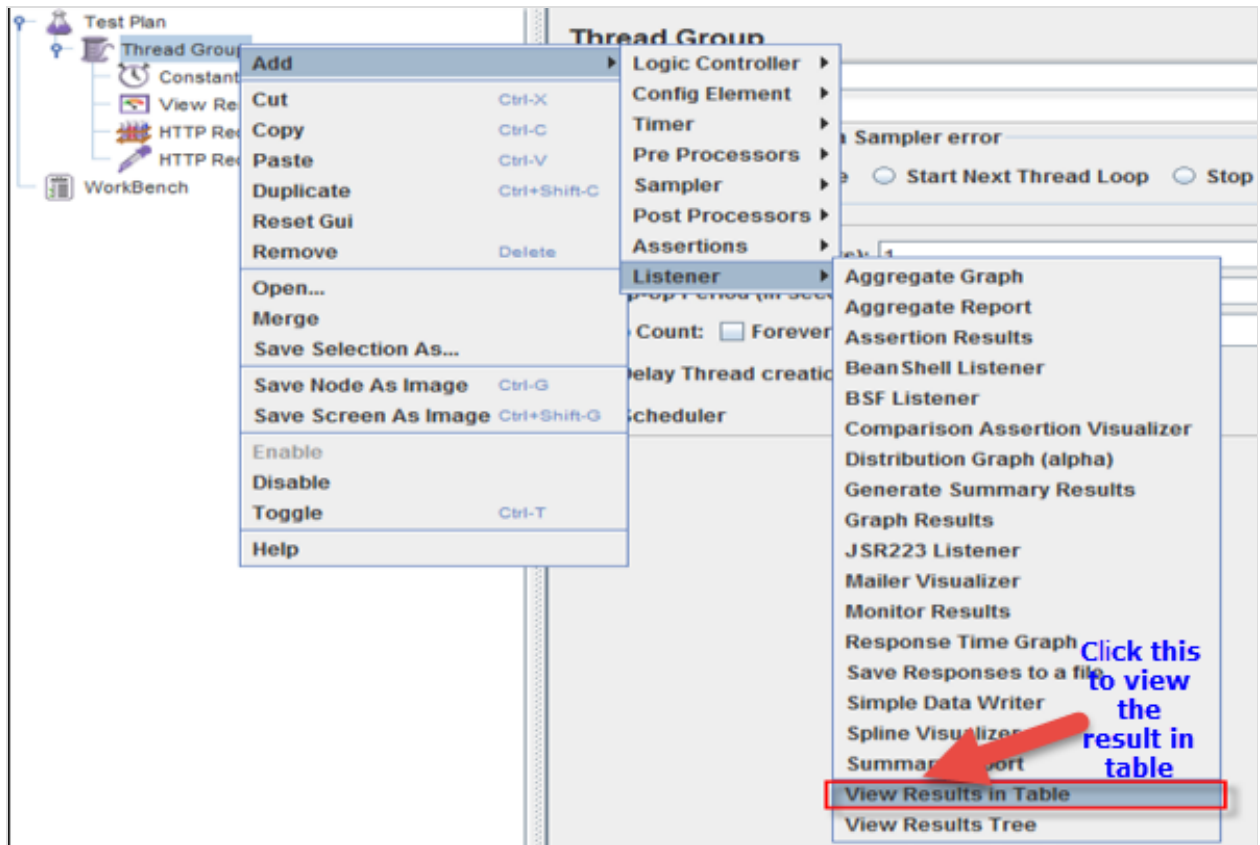


Right click on Loop Controller, from context menu, click on Add -> Sampler -> HTTP request.

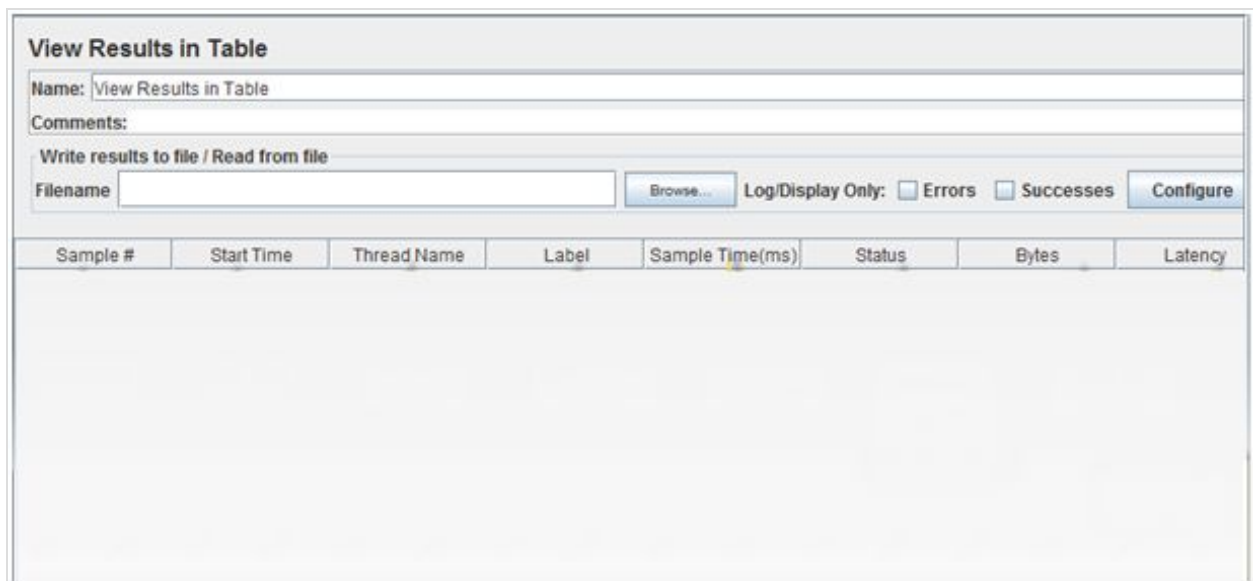


Third Step: Add View Results in Table:

To view the test result in tabular format, right click on **Add -> Listener -> View Result in Table**



View Results in Table will display like given below figure,



Fourth Step: Run the test:

After opening View Results in Table, click on Start button on Menu bar (Ctrl+R) to run test.

Uses of Processor in JMeter

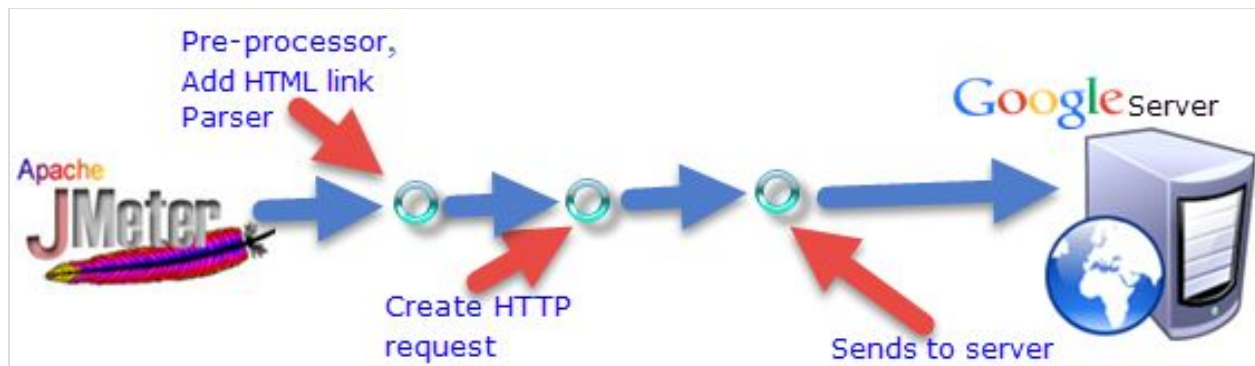
Processor in JMeter is helpful in modifying the Samplers as per their choice.

In JMeter Processors are of two type,

1. Pre-processor
2. Post-processor

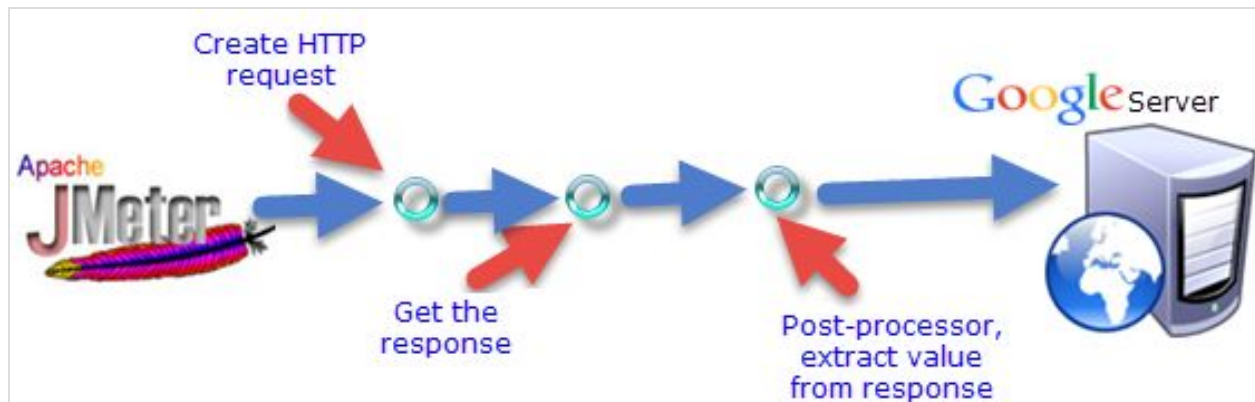
Pre-processor:

Pre-processor performs some task before building Sampler Request. Suppose, somebody wanted that JMeter to “spider” through website under test, parse link which find out all links on the page and return the HTML. Before creating an HTTP request, you have to add action “HTML link parser” to your controller.



Post-processor:

Post-processor performs some task after building Sampler Request. Suppose, using JMeter, user sends HTTP request to the web server “www.google.com” and get the response, under test. You will definitely like to stop the test, after getting response error by web server. To handle this situation, use post-processor, like this,



How Processor Works in JMeter?

Here you will get to know each and every step involved in creating Post-processor in JMeter. Before proceeding on this, first we should get to know that how to execute a simple test script,

First, using JMeter, user sends HTTP request to the web server "www.google.com", under test.

Second, user gets response from the Google server.

Third, if server sends the response "**error**", the test will be stopped by JMeter.

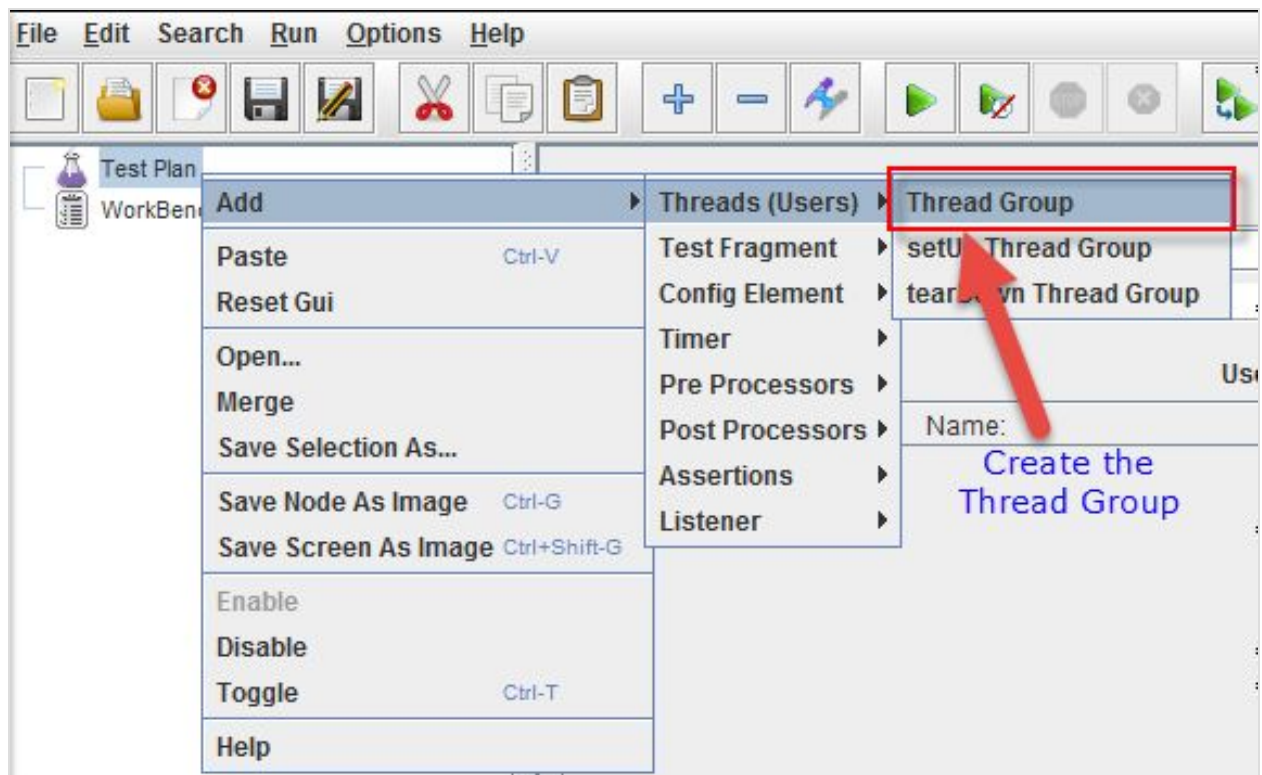
Fourth, if server sends the response "**OK**", the test will be continued by JMeter.

To do further process on this example,

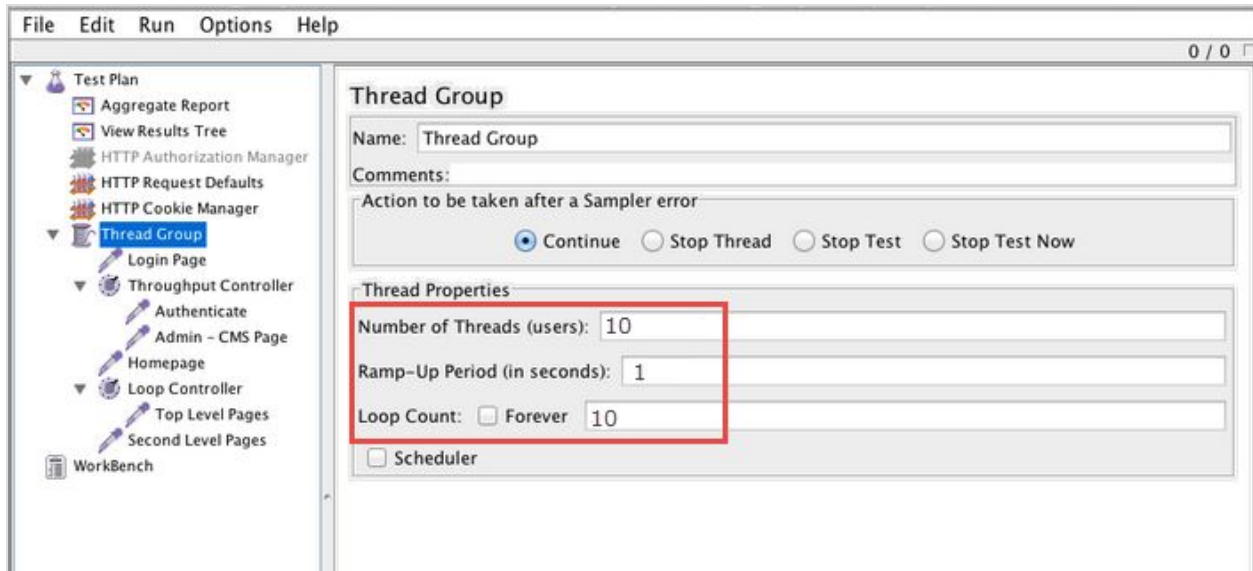
- Add Post-processor Element through Thread Group
- Add JMeter element
- Add Post-Processor Element
- Add Result view Tree
- Run the test

First step: Add Post-processor Element through Thread Group

To create the Thread Group, first run JMeter, from opened interface of JMeter choose Test Plan from the tree and right click to choose Add -> Threads (Users) -> Thread Group.



After opening thread Group, enter Thread Properties as shown in figure below,



In the above figure, **Number of Threads**: 10 numbers of users are connected to target website, **Loop Count**: execute testing 10 numbers of times, and **Ramp-Up Period**: 1.

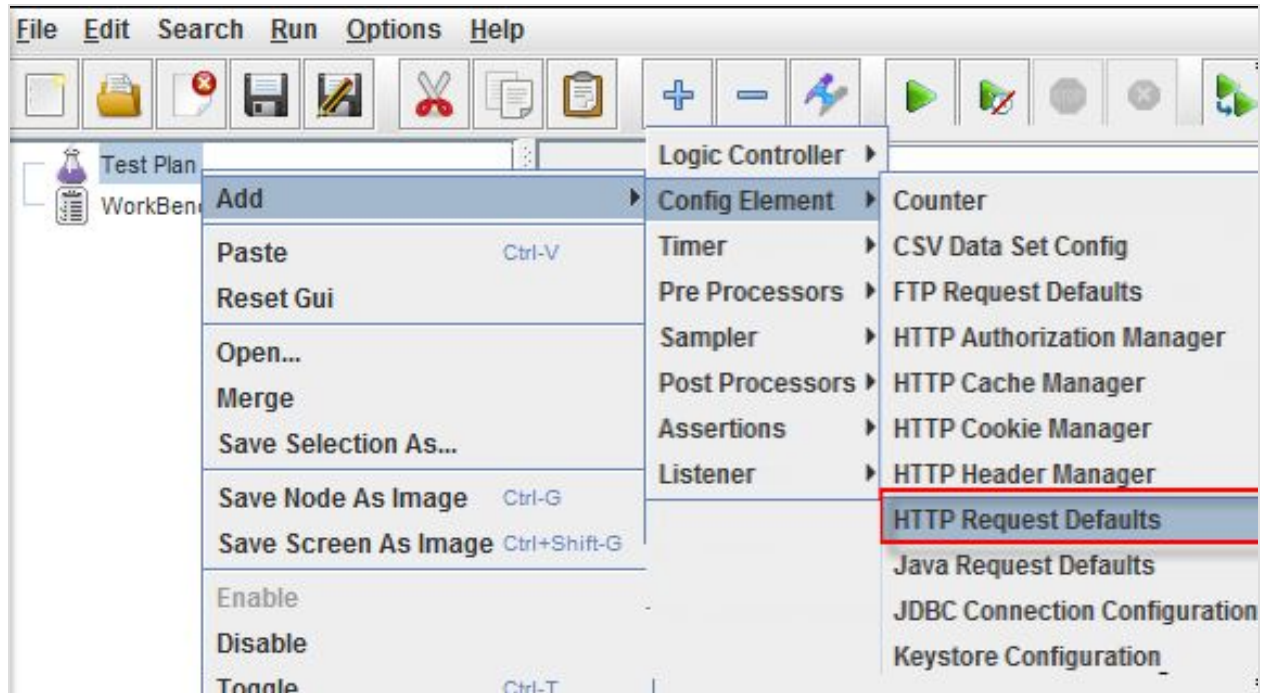
According to this setting, 10 user requests sends to web site <http://www.google.com> 10 times.

Second Step: Add JMeter elements

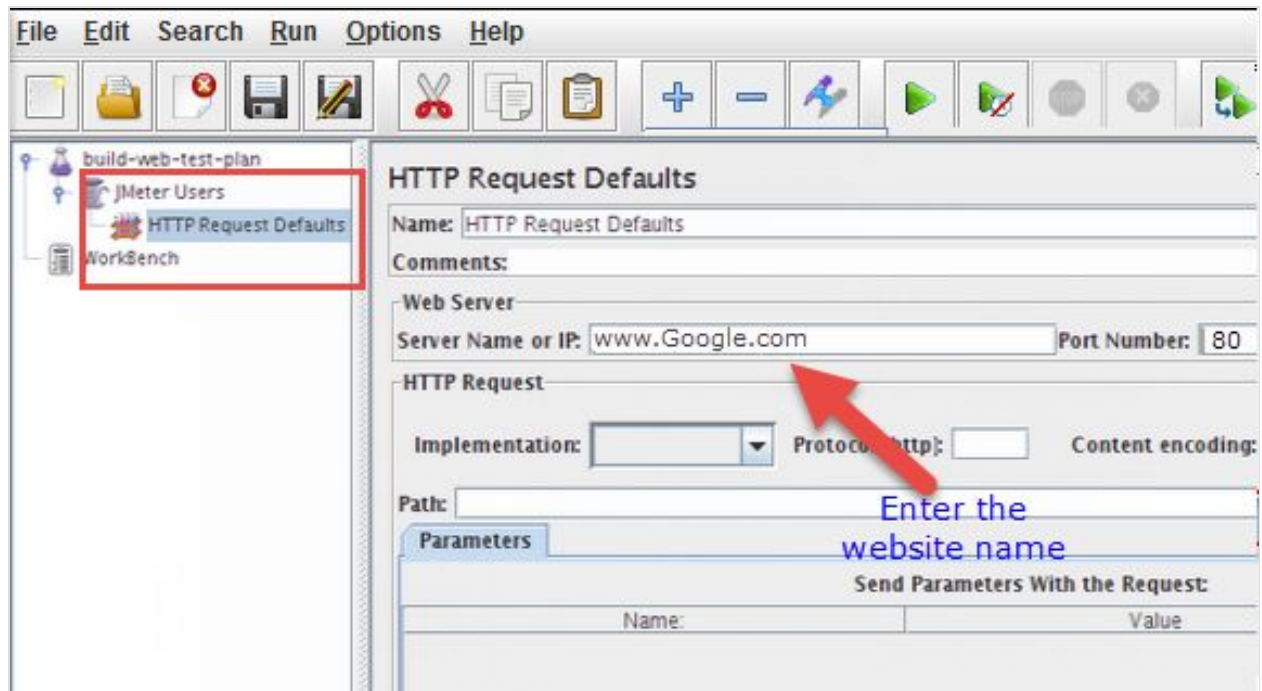
In this article we will add,

- HTTP request Default element
- HTTP Request

To get this element, go to Thread Group and right-clicking, from context menu select **Add** -> **Config Element** -> **HTTP Request Defaults**.

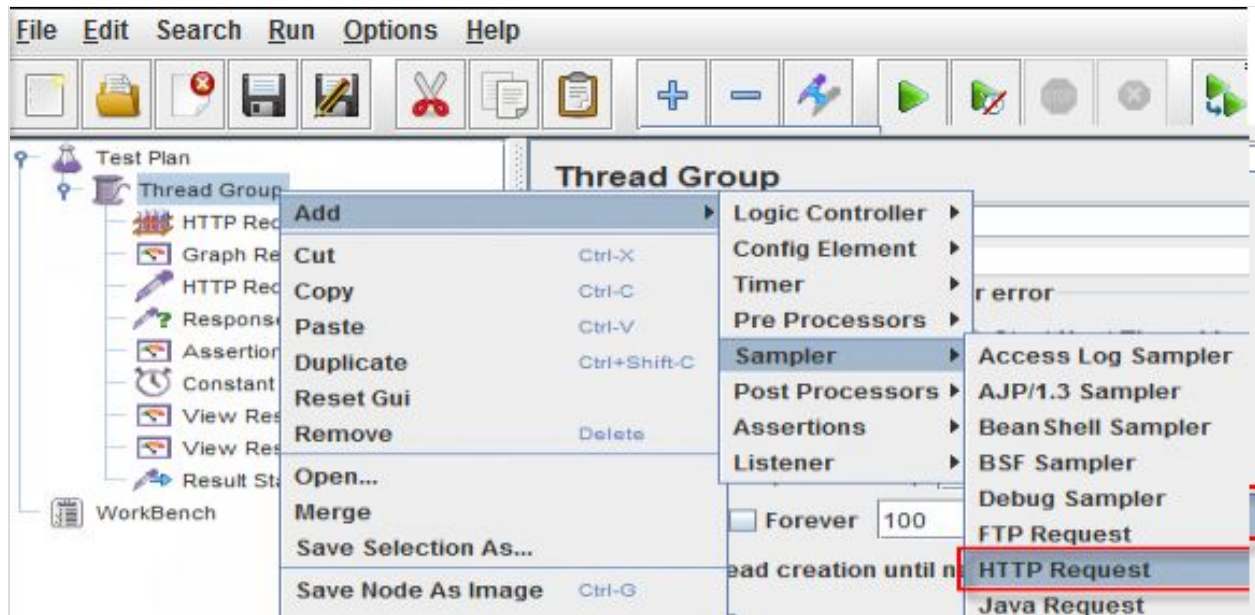


In the HTTP Request Defaults page, enter the Website name (www.Google.com) under Web server -> Server name or IP.

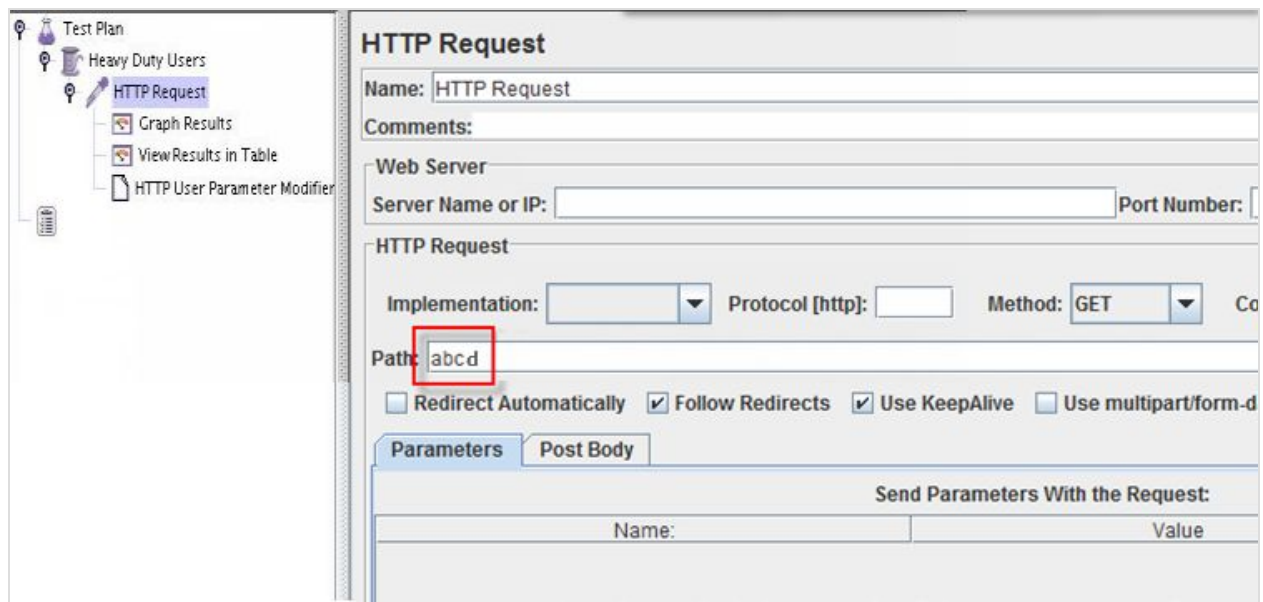


HTTP Request:

To get this element, go to Thread Group and right-clicking, from context menu select **Add**
-> **Sampler** -> **HTTP Request**.

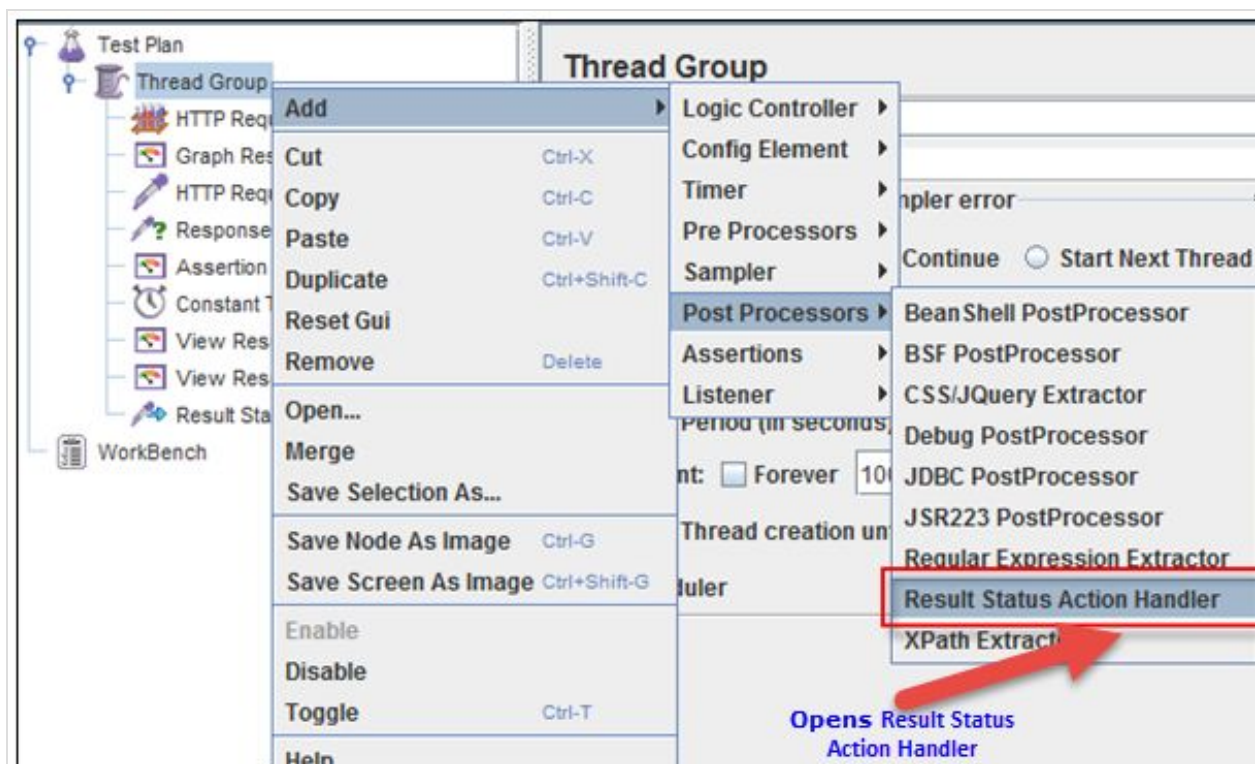


"Path" field in HTTP Request page shows that which **URL request** you need to direct to Google server like: enter "abcd" text in Path field. JMeter will generate the URL request <http://www.google.com/abcd> (wrong URL request) to Google server. If you leave the Path field blank, will URL request will be <http://www.google.com>.



Third Step: Add Post-Processor Element

To add Post-Processor Element, right click on **Thread Group** -> **Add** -> **Post Processor** -> **Result Status Action Handler** permits user to stop the thread or the whole test whenever user request get fail.



The following figure shows "Result Status Action Handler" pane. From "Result Status Action Handle" choose "**Stop Test Now**". This selection will stop the test if JMeter get the error from server response.

Result Status Action Handler

Name:

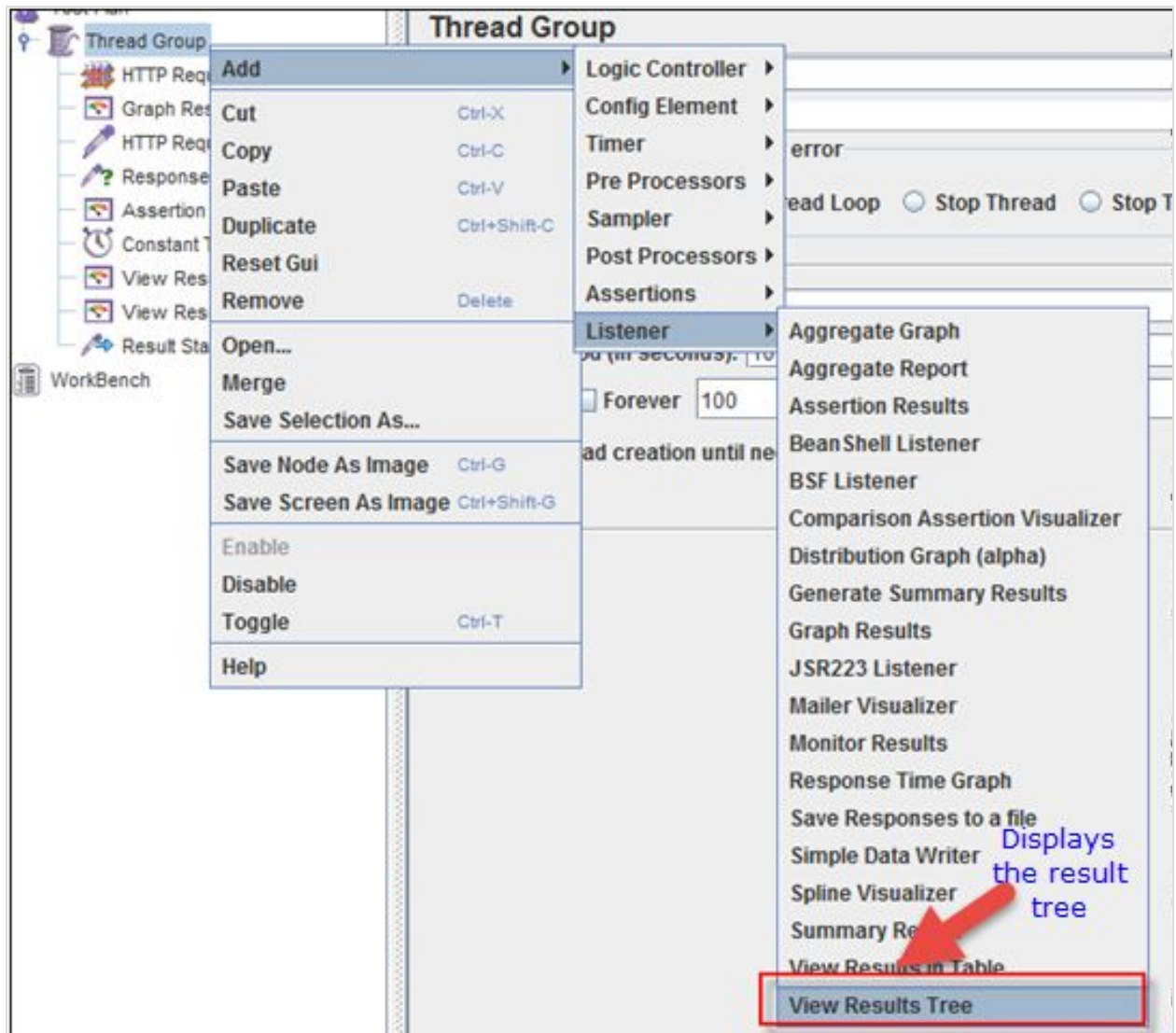
Comments:

Action to be taken after a Sampler error

☐ Continue ☐ Start Next Thread Loop ☐ Stop Thread ☐ Stop Test ☒ Stop Test Now

Fourth Step: Add View Result Tree

To open result tree, right click on Thread Group -> Add -> Listener -> View Result Tree, shown in the given below figure,



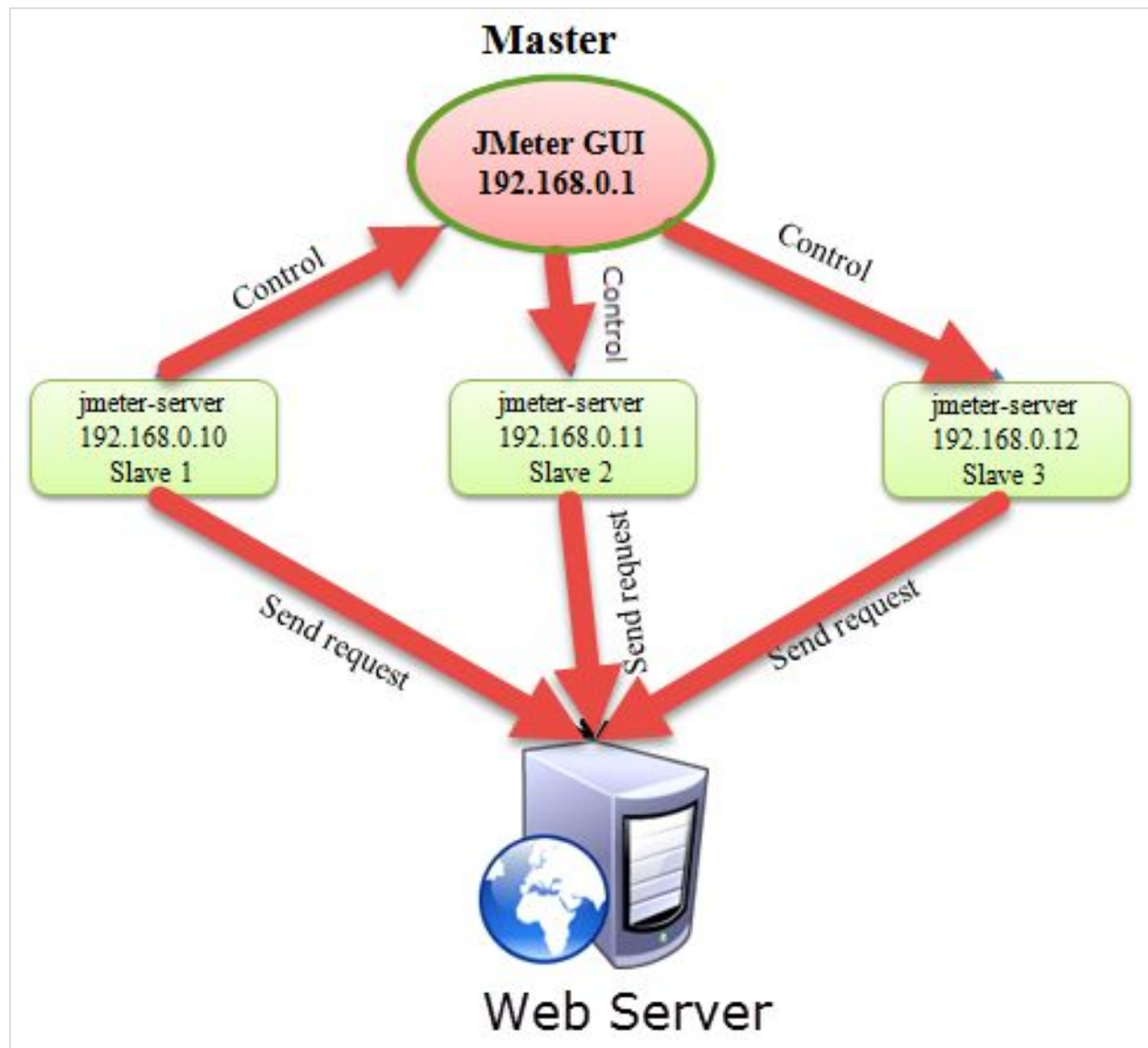
Fifth Step: Run the Test

In View Result Tree pane, press on Run button on Menu bar to see the result. It will display error message, sent by Google server and the test will stop.

JMeter's Distributed Testing

Distributed testing is a testing process, supports multiple systems to implement stress testing. Distributed testing is helpful to test those web sites and web server applications who work with multiple clients simultaneously.

Figure shows the client-server model of Distributed testing,



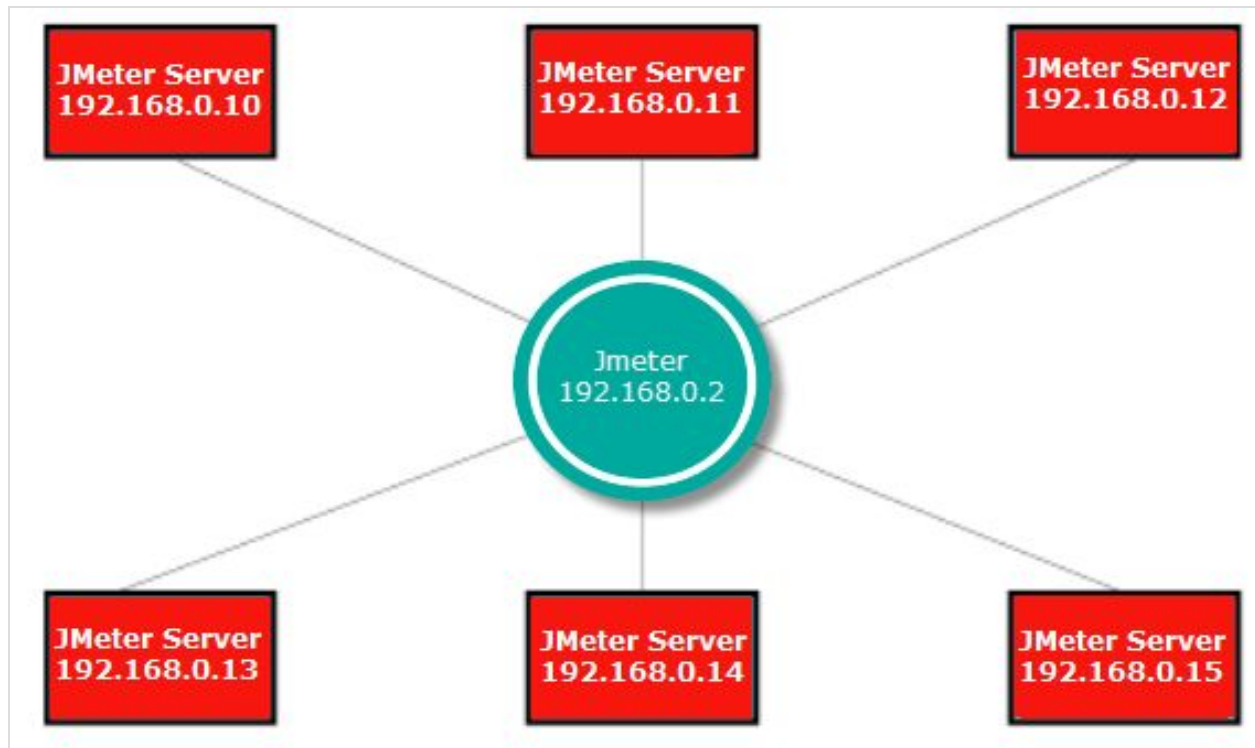
Where,

- **Master:** A JMeter GUI which controls each slave.
- **Slave:** A JMeter-server which gets command from the master and sends the request to server under test.
- **Target:** The web server gets request from slaves, under test

Copy all JMeter folders to all slave machines. Installation of JMeter makes environment variables for **JMETER_HOME** and **JMETER_BIN** in all the slave machines (even on Master machine).

To perform stress testing on multiple systems, there are certain things to check,

- System should have latest [JDK](#) on the host (slave) machine and latest [JMeter](#) on the host (slave) machine.
- Multiple systems firewall should be turned off
- Environment variables should define for JMeter(JMETER_HOME, JMETER_BIN)
- All clients should be in same subnet
- If web server belongs to "192.x.x.x" or "10.x.x.x" IP addresses, the web server will be in the same subnet. If the web server doesn't belongs to "192.x.x.x" or "10.x.x.x" IP addresses then there will be a problem
- Make sure, Jmeter can access the web server without any restriction
- All systems should have same version of Jmeter. Different versions may not work properly



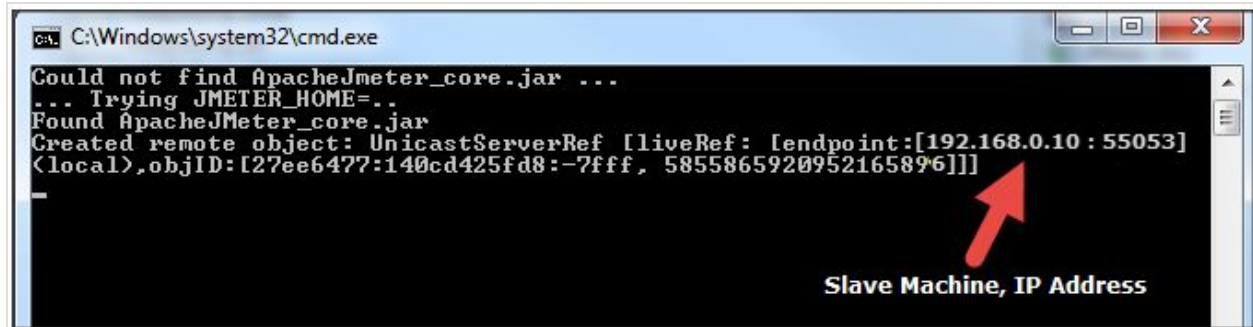
Step-by-Step Testing Process

- **System configuration**
- **Run the test**
- **Troubleshooting**

First Step: System configuration

Go to the **slave** systems -> jmeter/bin directory to execute "jmeter-server.bat" file.

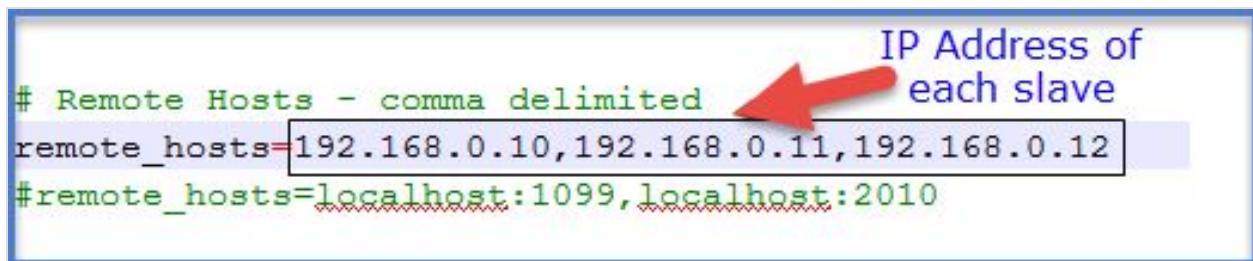
On windows, a slave machine which belongs to 192.168.0.10 IP address, looks like the given below figure,



```
C:\Windows\system32\cmd.exe
Could not find ApacheJmeter_core.jar ...
... Trying JMeter_HOME=..
Found ApacheJMeter_core.jar
Created remote object: UnicastServerRef [liveRef: [endpoint:[192.168.0.10 : 55053]
(local),objID:[27ee6477:140cd425fd8:-7fff, 5855865920952165896]]]
```

Slave Machine, IP Address

In the **master** systems, just visit /bin directory to edit **jmeter.properties** file, adds IP slave machine as shown below:



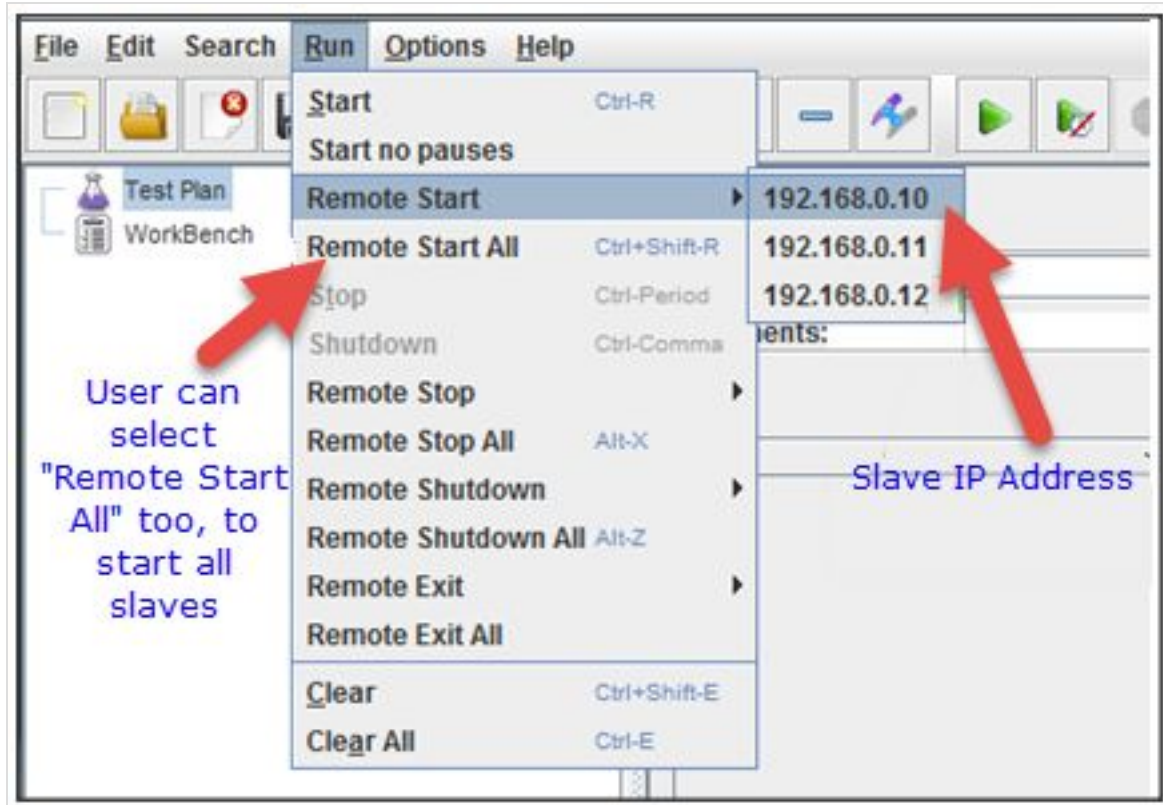
```
# Remote Hosts - comma delimited
remote_hosts=192.168.0.10,192.168.0.11,192.168.0.12
#remote_hosts=localhost:1099,localhost:2010
```

IP Address of each slave

Second Step: Run the test

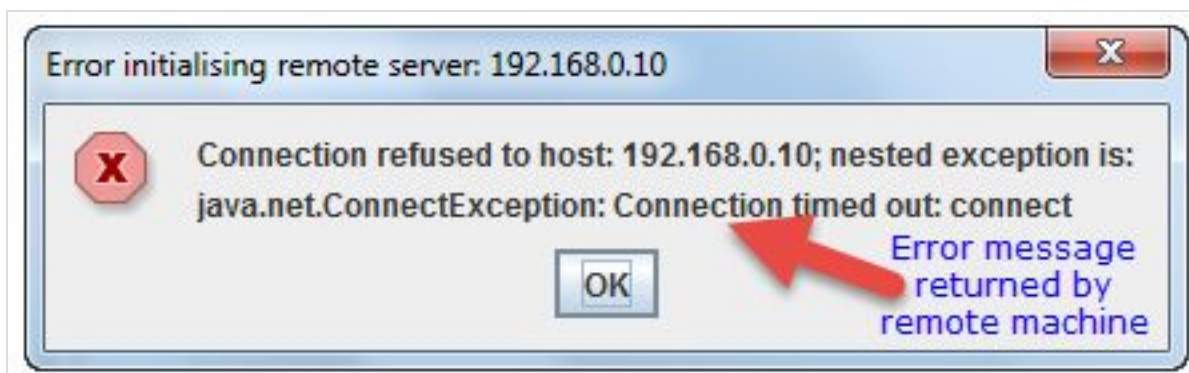
Open the test plan to run JMeter GUI, using the master machine.

Click on Run on the menu bar then choose **Remote start** -> select **the IP address** of slave machine.



Third Step: Troubleshooting

If you are unable to run the test and it returns error message, say administrator of the slave machine to run the jmeter-server.bat file, shown in the figure below.



Tips to Follow:

In some cases, Symantec Anti-Virus and Firewall blocks RMI traffic.

To stop Symantec firewall from windows services, go to the control panel – > administrative tools, double click services, find-out the Symantec antivirus, right click it to stop.

To stop window firewall, open network connections, choose the network connection, right click and select properties, select advanced tab and unchecked internet connection firewall.

Listener: Use “Aggregated Report” listener to see the aggregated report from your all host machine.

Limitations:

While proceeding, there are some limitations of distributed testing,

- Server and all clients should belong to **the same** subnet.
- Distributed testing needs target server with large processing power. The target Server should be capable to take **overloaded**, in case of getting too many requests by distributed JMeter tests.
- A single JMeter should be capable to handle a limited number of threads (100- 300 threads).
- The distributed JMeter testing is bit complex and difficult for beginner to understand and build.

Best Practices from JMeter

Procedure to reduce limitations of JMeter in distributed environment, following are the guidelines will assist in creating a real and continuous load:

- Decrease resource necessity
- Find out the Scoping Rules and design consequently
- Find the JMeter logs
- Before executing scripts, find out the default browser Connectivity settings
- Reduce and limit the Number of Threads
- Don't use functional mode
- Only use same sampler in a loop rather than lots of similar samplers, and also use variables (CSV Data Set) to contrast the sample. Possibly, use Access Log Sampler
- Use a proxy server and variables
- Fresh the Files tab earlier to every test run
- Delete the local path from CSV Data Set Config and follow file naming resolution



Best Practices from JMeter

JMeter has some limitations especially with distributed environment. To use JMeter powerfully for testing, follow these guidelines,

Limit the Number of Threads

System's hardware has capability to limit the number of threads which you run effectively with JMeter. It depends on how fast the server is (a faster server makes JMeter to work fast and response quickly). Due to overload of work on JMeter, each thread has to wait to get access to the CPU and more inflated the timing info gets.

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.

Using variables

There are some test plans essential to use different values for different users/threads. For example, If you like to test a sequence that needs a unique login for each user. This is simple to do using variables.

Decrease resource necessity

The GUI mode of the system takes a lot of computer memory under heavy load, origins performance issues.

These are some guidelines to decrease resource requirement:

- Use non-GUI mode – `JMeter -n -t test.jmx -l test.jtl`
- Only save the data or test result which is required. JMeter could take a long time to save very detailed test results.
- Deactivate the “View Result Tree” listener during the Load test. Since it consumes more memory and origins JMeter running to run out of memory.
- Deactivate all JMeter graphs results
- Use CSV test result format.

Check the JMeter logs

Any errors of the test plan or test execution should be recorded in the log files. Checking the log file assist you to discovery the error first.

Erase the local path from CSV Data Set Config

If you are using an existing CSV data file which you produced on your local computer, you should delete the current local path (Current path of CSV file). If you don't delete the local path, JMeter cannot find the CSV data file on your local PC.

Follow file naming convention

Save test plan with simple file name, use **only alphanumeric** characters.