Config element : jmeter

jMeter is an Open Source testing software. It is 100% pure Java application for load and performance testing. jMeter is designed to cover categories of tests like load, functional, performance, regression, etc., and it requires JDK 5 or higher. This tutorial will give you great understanding on jMeter framework needed to test an enterprise level application to deliver it with robustness and reliability.

Before going into the details of JMeter, let us first understand a few jargons associated with the testing of any application.

* **Performance Test** − This test sets the best possible performance expectation under a given configuration of infrastructure. It also highlights early in the testing process if any changes need to be made before the application goes into production.
* **Load Test** − This test is basically used for testing the system under the top load it was designed to operate under.
* **Stress Test** − This test is an attempt to break the system by overwhelming its resources.

What is JMeter?

JMeter is a software that can perform load test, performance-oriented business (functional) test, regression test, etc., on different protocols or technologies.

**Stefano Mazzocchi** of the Apache Software Foundation was the original developer of JMeter. He wrote it primarily to test the performance of Apache JServ (now called as Apache Tomcat project). Apache later redesigned JMeter to enhance the GUI and to add functional testing capabilities.

JMeter is a Java desktop application with a graphical interface that uses the Swing graphical API. It can therefore run on any environment / workstation that accepts a Java virtual machine, for example − Windows, Linux, Mac, etc.

The protocols supported by JMeter are −

* Web − HTTP, HTTPS sites 'web 1.0' web 2.0 (ajax, flex and flex-ws-amf)
* Web Services − SOAP / XML-RPC
* Database via JDBC drivers
* Directory − LDAP
* Messaging Oriented service via JMS
* Service − POP3, IMAP, SMTP
* FTP Service

JMeter Features

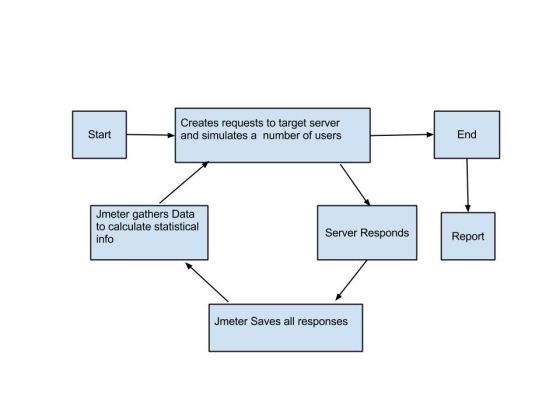
Following are some of the features of JMeter −

* Being an open source software, it is freely available.
* It has a simple and intuitive GUI.
* JMeter can conduct load and performance test for many different server types − Web - HTTP, HTTPS, SOAP, Database via JDBC, LDAP, JMS, Mail - POP3, etc.
* It is a platform-independent tool. On Linux/Unix, JMeter can be invoked by clicking on JMeter shell script. On Windows, it can be invoked by starting the jmeter.bat file.
* It has full Swing and lightweight component support (precompiled JAR uses packages javax.swing.\* ).
* JMeter store its test plans in XML format. This means you can generate a test plan using a text editor.
* Its full multi-threading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.
* It is highly extensible.
* It can also be used to perform automated and functional testing of the applications.

How JMeter Works?

JMeter simulates a group of users sending requests to a target server, and returns statistics that show the performance/functionality of the target server/application via tables, graphs, etc.

Take a look at the following figure that depicts how JMeter works −



JMeter is a framework for Java, so the very first requirement is to have JDK installed in your machine.

## Step 1 − Verify Java Installation

First of all, verify whether you have Java installed in your system. Open your console and execute one of the following **java** commands based on the operating system you are working on.

|  |  |  |
| --- | --- | --- |
| **OS** | **Task** | **Command** |
| Windows | Open Command Console | c:\> java -version |
| Linux | Open Command Terminal | $ java -version |
| Mac | Open Terminal | machine: ~ joseph$ java -version |

If you have Java installed in your system, you would get an appropriate output based on the OS you are working on.

If you do not have Java installed, install the Java Software Development Kit (SDK) from<http://www.oracle.com/technetwork/java/javase/downloads/index.html>. We are assuming Java 1.7.0\_25 as the installed version for this tutorial.

## Step 2 − Set Java Environment

Set the **JAVA\_HOME** environment variable to point to the base directory location, where Java is installed on your machine. For example −

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | Set the environment variable JAVA\_HOME to C:\Program Files\Java\jdk1.7.0\_25 |

Append Java compiler location to System Path.

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | Append the string; C:\Program Files\Java\jdk1.7.0\_25\bin to the end of the system variable, Path. |

Verify Java Installation using **java -version** command as explained above.

Step 3 − Download JMeter

Download the latest version of JMeter from <http://jmeter.apache.org/download_jmeter.cgi>. For this tutorial, we downloaded *apache-jmeter-2.9* and copied it into C:\>JMeter folder.

The directory structure should look like as shown below −

* apache-jmeter-2.9
* apache-jmeter-2.9\bin
* apache-jmeter-2.9\docs
* apache-jmeter-2.9\extras
* apache-jmeter-2.9\lib\
* apache-jmeter-2.9\lib\ext
* apache-jmeter-2.9\lib\junit
* apache-jmeter-2.9\printable\_docs

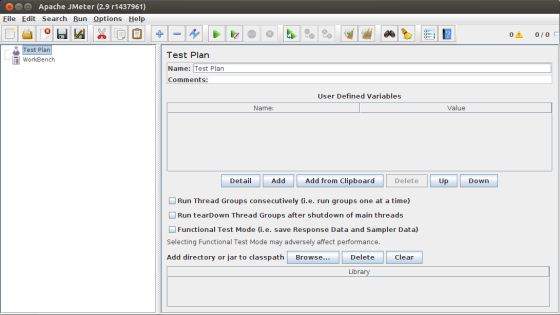
You can rename the parent directory (i.e. apache-jmeter-2.9) if you want, but do not change any of the sub-directory names.

Step 4 − Run JMeter

After downloading JMeter, go to the *bin* directory. In this case, it is **/home/manisha/apache-jmeter-2.9/bin**. Now click on the following −

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | jmeter.bat |
| Linux | jmeter.sh |
| Mac | jmeter.sh |

After a short pause, the JMeter GUI should appear, which is a Swing application, as seen in the following screenshot −



## What is a Test Plan?

A Test Plan can be viewed as a container for running tests. It defines what to test and how to go about it. A complete test plan consists of one or more elements such as thread groups, logic controllers, sample-generating controllers, listeners, timers, assertions, and configuration elements. A test plan must have at least one thread group.

## Writing a Test Plan

Follow the steps given below to write a test plan −

### Step 1 − Start the JMeter Window

Open the JMeter window by clicking jmeter batch file.The JMeter window will appear as shown above −

This is a plain and blank JMeter window without any additional elements added to it. It contains two nodes −

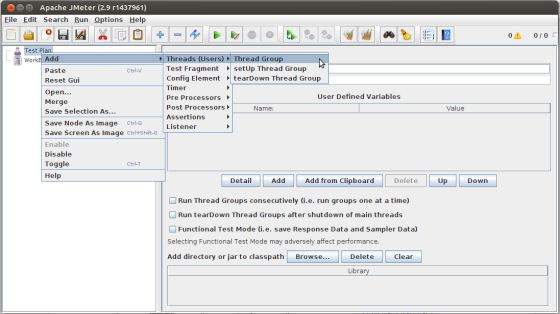
* **Test Plan node** − is where the real test plan is kept.
* **Workbench node** − It simply provides a place to temporarily store test elements while not in use, for copy/paste purposes. When you save your test plan, Workbench items are not saved with it.

### Step 2 − Add/Remove Elements

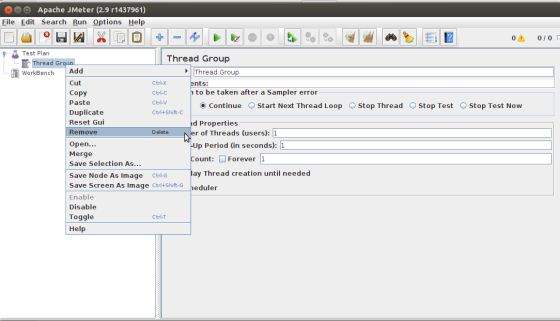
Elements (which will be discussed in the next chapter [Test Plan Elements](https://www.tutorialspoint.com/jmeter/jmeter_test_plan_elements.htm)) can be added to a test plan by right-clicking on the Test Plan node and choosing a new element from the "add" list.

Alternatively, you can load an element from a file and add it by choosing the "merge" or "open" option.

For example, let us add a Thread Group element to a Test Plan as shown below −



To remove an element, make sure the element is selected, right-click on the element, and choose the "remove" option.

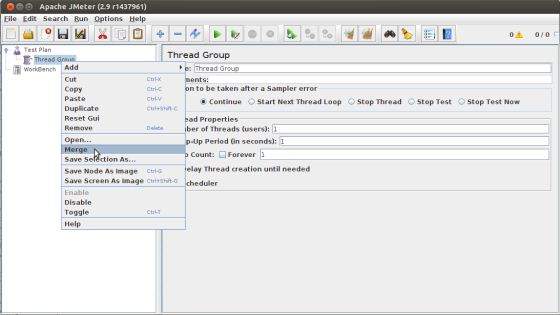


### Step 3 − Load and Save the Elements

To load an element from file −

* Right-click on the existing tree element to which you want to add the loaded element.
* Select Merge.
* Choose the file where you saved the elements.
* JMeter will merge the elements into the tree.

By default, JMeter does not save the element, you need to explicitly save it.



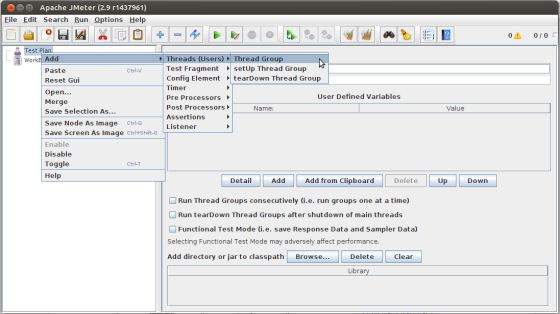
To save tree elements −

* Right-click on the element.
* Choose the *Save Selection As* ... option.

JMeter will save the element selected, plus all the child elements beneath it. By default, JMeter doesn't save the elements, you need to explicitly save it as mentioned earlier.

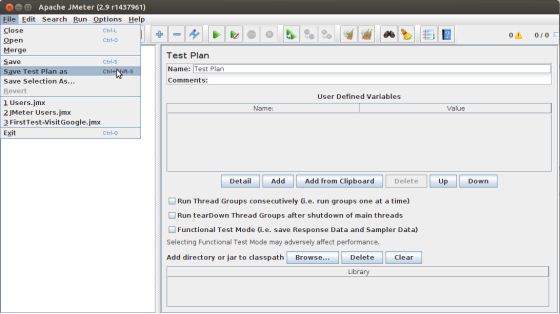
### Step 4 − Configuring the Tree Elements

Any element in the Test Plan can be configured using the controls present in JMeter's right-hand side frame. These controls allow you to configure the behavior of that particular test element. For example, the Thread Group can be configured for a number of users, ramp up periods, etc., as shown below −



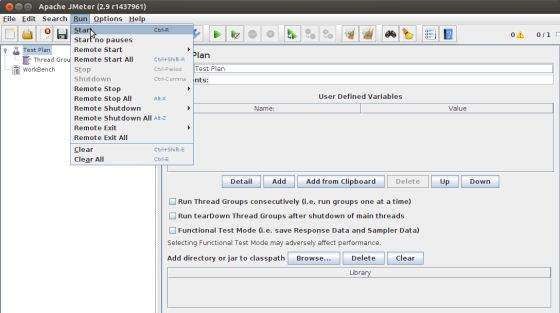
### Step 5 − Saving the Test Plan

You can save an entire Test Plan by using either **Save** or **"Save Test Plan As ..."** from the File menu.



### Step 6 − Run the Test Plan

You can run the Test Plan by clicking **Start**(Control + r) from the **Run** menu item. When JMeter starts running, it shows a small green box at the right-hand end of the section just under the menubar.



The numbers to the left of the green box are the number of active threads / total number of threads. These only apply to a locally run test; they do not include any threads started on remote systems when using client-server mode.

### Step 7 − Stop the Test Plan

You can stop your test in two ways −

* Using **Stop** (Control + '.'). It stops the threads immediately if possible.
* Using **Shutdown** (Control + ','). It requests the threads to stop at the end of any current work.

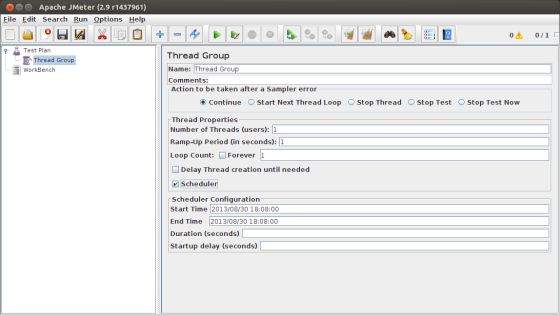
A Test Plan comprises of at least one Thread Group. Within each Thread Group, we may place a combination of one or more of other elements − Sampler, Logic Controller, Configuration Element, Listener, and Timer. Each Sampler can be preceded by one or more Pre-processor element, followed by Post-processor element, and/or Assertion element.

## Thread Group

Thread Group elements are the beginning points of your test plan. As the name suggests, **the thread group elements control the number of threads JMeter will use during the test**. We can also control the following via the Thread Group −

* Setting the number of threads
* Setting the ramp-up time
* Setting the number of test iterations

The Thread Group Control Panel looks like this −



The Thread Group Panel holds the following components −

* **Action to be taken after a Sampler error** − In case any error occurs during test execution, you may let the test either −
  + **Continue** to the next element in the test
  + **Stop Thread** to stop the current Thread.
  + **Stop Test** completely, in case you want to inspect the error before it continues running.
* **Number of Threads** − Simulates the number of users or connections to your server application.
* **Ramp-Up Period** Defines how long it will take JMeter to get all threads running.
* **Loop Count** − Defines the number of times to execute the test.
* **Scheduler checkbox** − Once selected, the Scheduler Configuration section appears at the bottom of the control panel.
* **Scheduler Configuration** − You can configure the start and end time of running the test.

## Controllers

JMeter has two types of Controllers − *Samplers* and *Logic Controllers*.

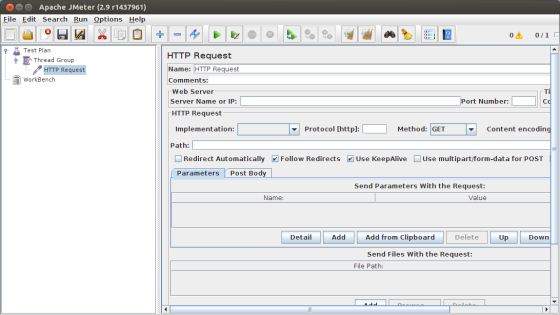
### Samplers

Samplers allow JMeter to send specific types of requests to a server. They simulate a user request for a page from the target server. For example, you can add a HTTP Request sampler if you need to perform a POST, GET, or DELETE on a HTTP service.

Some useful samplers are −

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

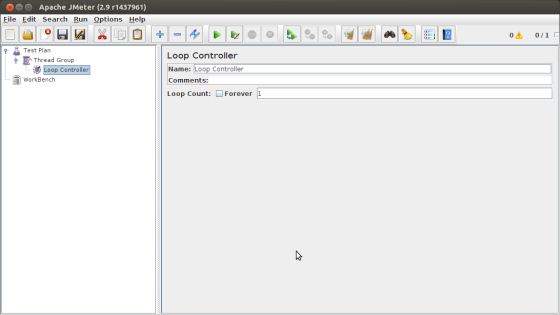
The following screenshot shows an HTTP Request Sampler Control Panel −



### Logic Controllers

Logic Controllers let you control the order of processing of Samplers in a Thread. Logic controllers can change the order of a request coming from any of their child elements. Some examples are − ForEach Controller, While Controller, Loop Controller, IF Controller, Run Time Controller, Interleave Controller, Throughput Controller, and Run Once Controller.

The following screenshot shows a Loop Controller Control Panel −



The following list consists of all the Logic Controllers JMeter provides −

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

### Test Fragments

A Test Fragment is a special type of element placed at the same level as the Thread Group element. It is distinguished from a Thread Group in that it is not executed unless it is referenced by either a Module Controller or an Include\_Controller. This element is purely for code re-use within Test Plans.

## Listeners

Listeners let you view the results of Samplers in the form of tables, graphs, trees, or simple text in some log files. They provide visual access to the data gathered by JMeter about the test cases as a Sampler component of JMeter is executed.

Listeners can be added anywhere in the test, including directly under the test plan. They will collect data only from elements at or below their level. The following list consists of all the Listeners JMeter provides −

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

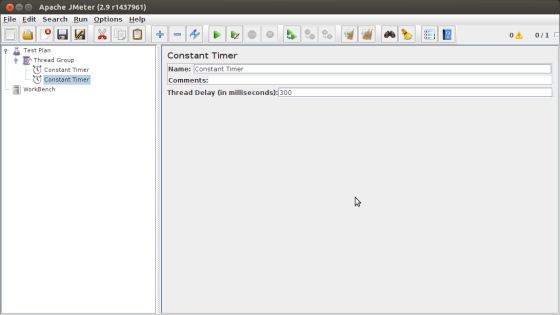
## Timers

By default, a JMeter thread sends requests without pausing between each sampler. This may not be what you want. You can add a timer element which allows you to define a period to wait between each request.

The following list shows all the timers that JMeter provides −

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

The following screenshot shows a Constant Timer Control Panel −



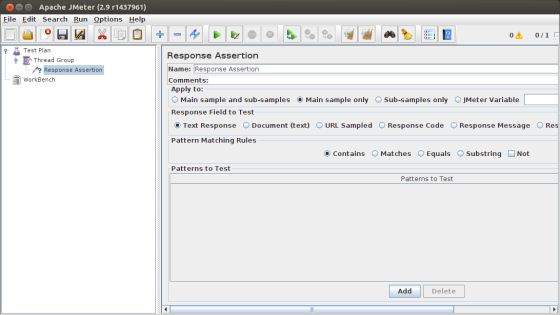
## Assertions

Assertions allow you to include some validation test on the response of your request made using a Sampler. Using assertions you can prove that your application is returning the correct data. JMeter highlights when an assertion fails.

The following list consists of all the assertions JMeter provides −

* Beanshell Assertion
* BSF Assertion
* Compare Assertion
* JSR223 Assertion
* Response Assertion
* Duration Assertion
* Size Assertion
* XML Assertion
* BeanShell Assertion
* MD5Hex Assertion
* HTML Assertion
* XPath Assertion
* XML Schema Assertion

The following screenshot shows a Response Assertion Control Panel −



## Configuration Elements

Configuration Elements allow you to create defaults and variables to be used by Samplers. They are used to add or modify requests made by Samplers.

They are executed at the start of the scope of which they are part, before any Samplers that are located in the same scope. Therefore, a Configuration Element is accessed only from inside the branch where it is placed.

The following list consists of all the Configuration Elements that JMeter provides −

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

## Pre-processor Elements

A pre-processor element is something that runs just before a sampler executes. They are often used to modify the settings of a Sample Request just before it runs, or to update variables that are not extracted from response text.

The following list consists of all the pre-processor elements that JMeter provides −

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

## Post-processor Elements

A post-processor executes after a sampler finishes its execution. This element is most often used to process the response data, for example, to retrieve a particular value for later use.

The following list consists of all the Post-Processor Elements JMeter provides −

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

## Execution Order of Test Elements

Following is 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)

-----------sorting and seraching