



Published on [ONJava.com](http://www.onjava.com/) (<http://www.onjava.com/>)
<http://www.onjava.com/pub/a/onjava/2003/01/15/jmeter.html>
[See this](#) if you're having trouble printing code examples

Using JMeter

by [Budi Kurniawan](#)

01/15/2003

JMeter is a Java-based tool for load testing client-server applications. Stefano Mazzocchi originally wrote it to test the performance of Apache JServ (the predecessor of Jakarta Tomcat). It has since become a subproject of Jakarta.

Installing and Running JMeter

The most recent release of JMeter is version 1.8. You can download the latest stable version from [JMeter's site](#). Downloads are available as either .gz or .zip files. JMeter 1.8 requires a working JDK 1.4 environment.

Once you extract the binary distribution file, JMeter is ready for you. On Linux/UNIX, run JMeter by invoking the `jmeter` shell script. On Windows, call the `jmeter.bat` file. Both files can be found in the `bin/` directory of the JMeter installation directory. Figure 1 shows JMeter's main window, which is a Swing application.

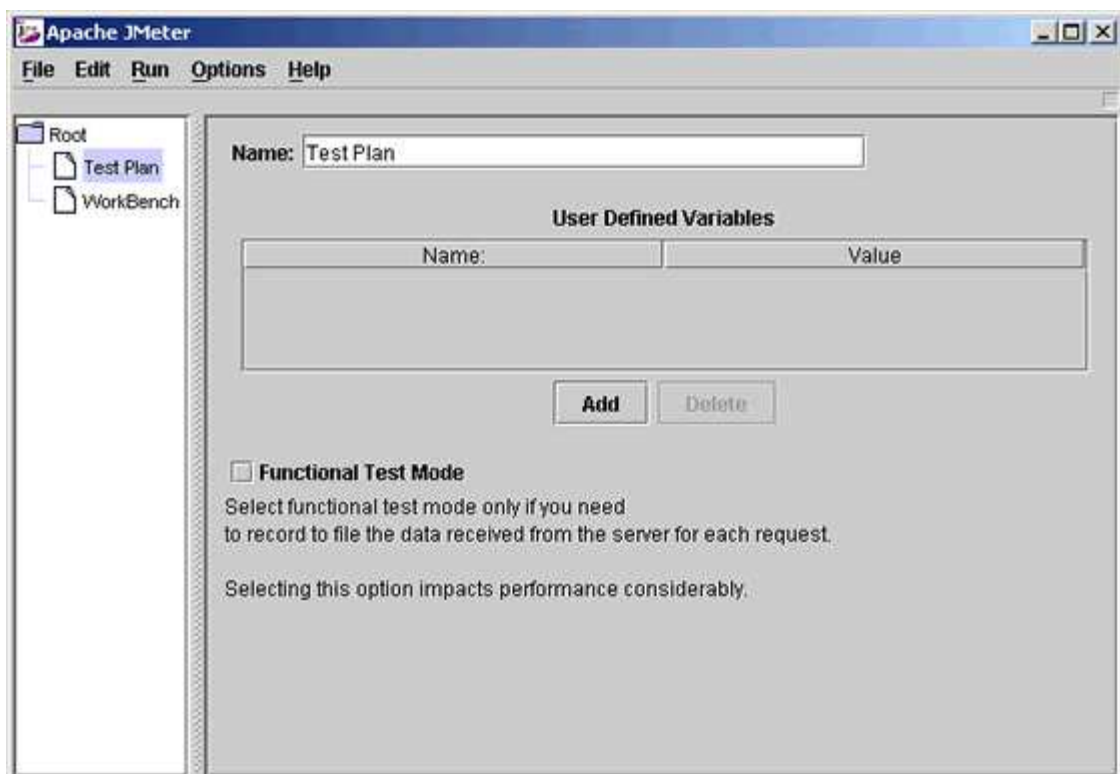


Figure 1: JMeter's main window.

The user interface has two panes. The left pane displays the elements used in our testing. Initially, there are the Root and two sub-elements, Test Plan and WorkBench. In this article we're only concerned with Test Plans. Add an element to a node by right-clicking it and selecting Add. To remove an element, select the element by clicking on it, then right-click on the element and choose the Remove option.

The right pane of the user interface displays the details of each element. You are now ready to use JMeter. There are two things to note:

1. You should not run JMeter on the same machine running the application to be tested. JMeter may use extensive resources that might affect the other application's performance if they are both run on the same machine.
2. Make sure that the testing is affected as little as possible by network traffic. The best thing to do is to ask your network administrator to set up an isolated sub-network for the machine running the Web application and the machine running JMeter.

Using JMeter for a Simple Test

Let's start with a very simple test. In this test, we will set up a test plan and stress test a Web application. You will be introduced with some common concepts in JMeter. After understanding this basic test, you should be able to use all of the capabilities of JMeter.

To conduct a test, you must have a test plan. A test plan describes the steps that JMeter will take to perform the testing. A test plan includes elements such as thread groups, logic controllers, sample generating controllers, listeners, timers, assertions, and configuration elements. Don't worry at this stage if you don't understand what these elements are.

A test plan must have at least one thread group. A thread group is the starting point of a test plan, and it can contain all other JMeter elements. A thread group controls the threads that will be created by JMeter to simulate simultaneous users.

Now, let's start by creating a thread group. Right-click the Test Plan element and select Add and then Thread Group. JMeter will create a thread group element under Test Plan element. Click the Thread Group element, and you will see a screen like the one in Figure 2.

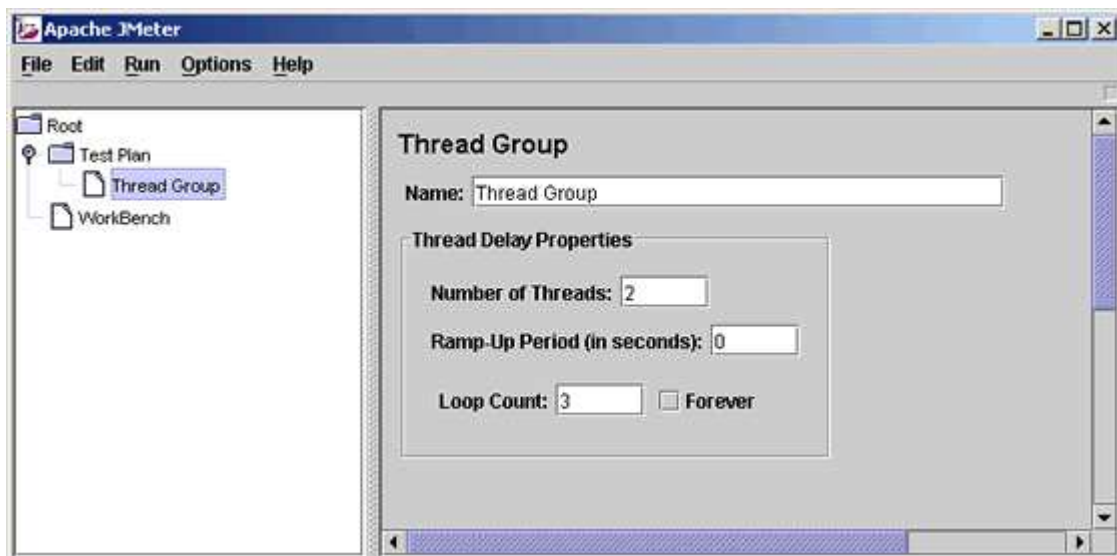


Figure 2: Configuring a thread group.

In this page, you can set the following properties:

- *Name* -- the name of this thread group. You can give a descriptive name to this property.
- *Number of Threads* -- the number of threads created. Each thread represents a single user. Therefore, if you want to simulate a load test with 10 concurrent users, enter 10 as the value for this property.
- *Ramp-Up Period* -- the number of seconds JMeter will take to accelerate to create all of the threads needed. If the number of threads used is 10 and the ramp-up period is 20 seconds, JMeter will take 20 seconds to create those 10 threads, creating one new thread every two seconds. If you want all threads to be created at once, put 0 in this box.
- *Forever* -- if clicked, this option tells JMeter to keep sending requests to the tested application indefinitely. If disabled, JMeter will repeat the test for the number of times entered in the Loop Count box.
- *Loop Count* --this property value only has an effect if the Forever check box is unchecked. It tells JMeter the number of times it has to repeat the test.

For our simple test, fill the properties with the values found in Figure 2. We will use two users and each test will be performed three times. We use small numbers here so that we can easily explain the results later; in real load testing, you might want to use higher numbers for these properties.

Next, you need to add the element that represents HTTP requests. To do so, right-click the Thread Group element, and select Add, Sampler, and then HTTP Request. An HTTP Request element will be added to the Thread Group element. Click the HTTP Request element to select it, and you should see a screen similar to Figure 3.

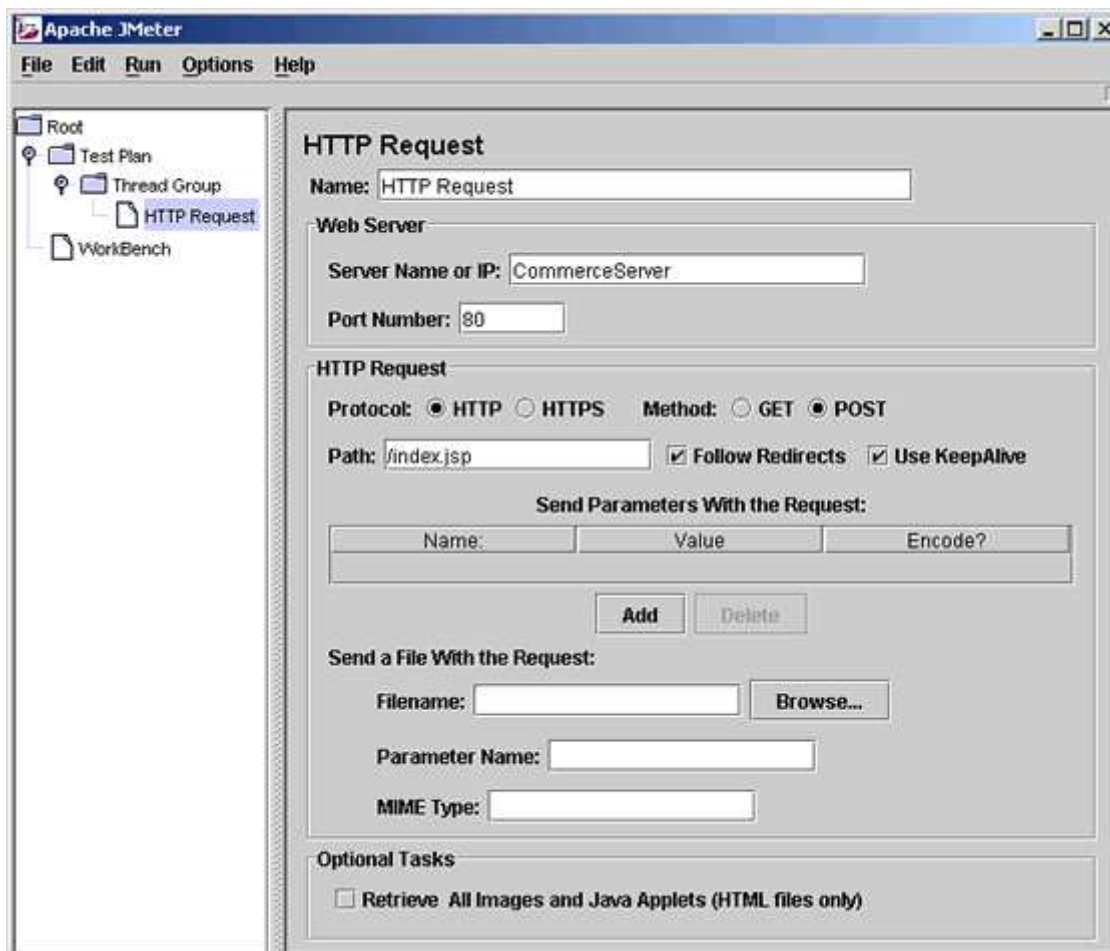


Figure 3: Configuring an HTTP Request element.

On the HTTP Request screen, you configure the HTTP requests that will be used to "hit" your application. Here, you can set the following properties.

- *Name* -- the name of this HTTP request. The name should be descriptive; remember that it is common to have multiple HTTP Request elements in a thread group.
- *Server Name or IP* -- the server name or the IP address of the machine running the application being tested.
- *Port Number* -- the port number used by the application. Normally, a Web application runs on port 80.
- *Protocol* -- the protocol used, either HTTP or HTTPS.
- *Method* -- the request method, either GET or POST.
- *Path* -- the path to the resource that will handle this request.
- *Follow Redirects* -- follows redirections sent by the Web application, if any.
- *Use KeepAlive* -- if checked, sends the `Connection = Keep-Alive` request header. By default, an HTTP 1.1 browser uses `Keep-Alive` as the value of the `Connection` header. Therefore, this checkbox should be checked.
- *Parameters* -- the list of parameters sent with this request. Use the Add and Delete buttons to add and remove parameters.
- *Send a file with a request* -- simulate a file upload to the Web application.
- *Retrieve all images and Java Applets* -- download embedded content.

Now you should be able to figure out the values for the properties in the HTTP Request page. The last element that we need to add to our test plan is a listener, which in JMeter is the same as a report. JMeter comes with various reports to choose from. A report can be a table or a graph. For this testing, use the easiest report available: a table.

To add a listener, right-click the Thread Group element, select Add, and then Listener and View Results in Table. Now you are ready to run the test plan. Before you run your test plan, however, you are advised to save the test plan just in case JMeter crashes the system (an occasional occurrence with higher numbers of threads and loop counts). Afterwards, select Start from the Run menu to execute the test plan.

While the test plan executes, the small box on the bar right below the menu bar will turn green. For a test that does not run indefinitely, JMeter will automatically stop the test plan after it's finished. For a test that goes on indefinitely, you must intervene to stop the test. Do this by selecting Stop from the Run menu.

When I ran my test plan, I got the results like those shown in Figure 4.

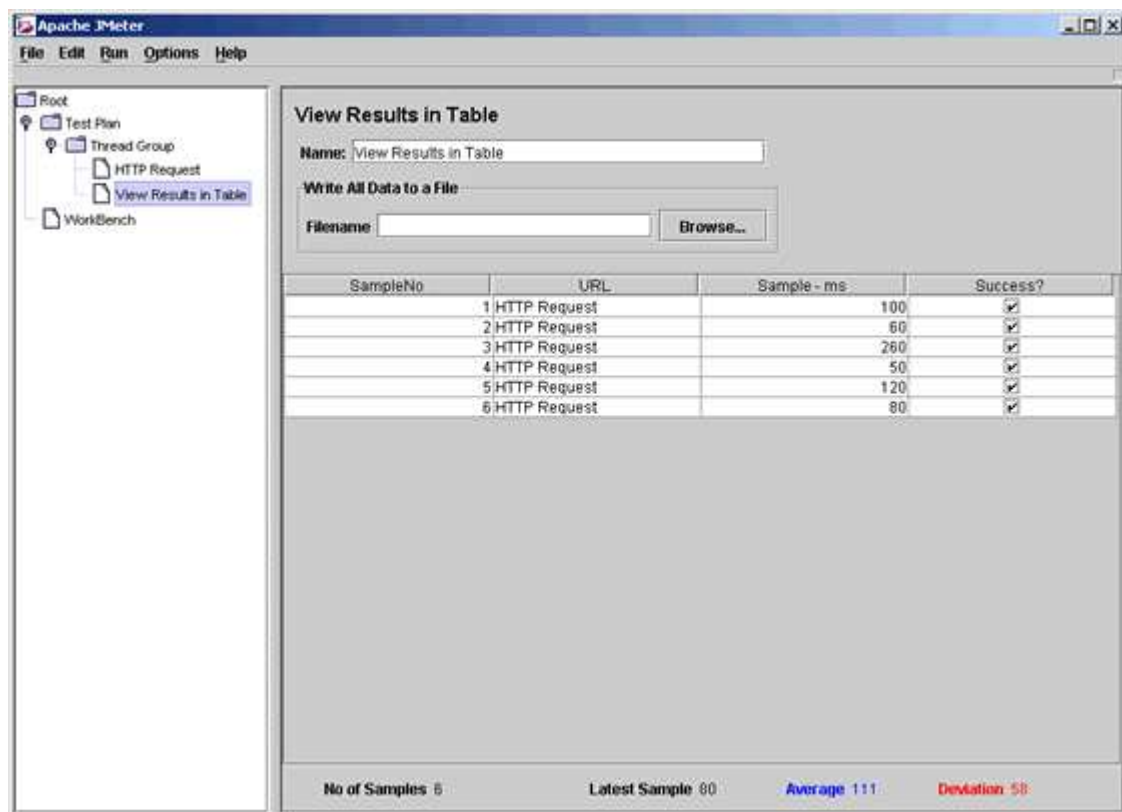


Figure 4 shows the 'View Results in Table' window in Apache JMeter. The window displays a table of test results for a 'View Results in Table' listener. The table has four columns: SampleNo, URL, Sample - ms, and Success?. It contains six rows of data, all of which are successful HTTP requests. Summary statistics at the bottom indicate 6 samples, a latest sample of 80, an average of 111, and a deviation of 58.

SampleNo	URL	Sample - ms	Success?
1	HTTP Request	100	✓
2	HTTP Request	60	✓
3	HTTP Request	260	✓
4	HTTP Request	50	✓
5	HTTP Request	120	✓
6	HTTP Request	80	✓

Summary statistics:
No of Samples: 6
Latest Sample: 80
Average: 111
Deviation: 58

Figure 4: A table report.

It is very easy to understand the figures in the table. There are six samples taken (two threads and three loop counts, thus $2 \times 3 = 6$). The response time from each sample is given in the third column. They are 100, 60, 260, 50, 120, and 80 ms. All samples are taken successfully, as described by the fourth column. On average, each sample has a response time of 111 ms $((100 + 60 + 260 + 50 + 120 + 80)/6)$.

Another important figure is the standard deviation, defined as the square root of the total of the deviation of each sample from the average. This figure indicates how stable your Web application is. If the standard deviation is high, some users will experience very good responses while some other users will wait for a longer time. The smaller this value, the better.

After conducting a simple test, it is very easy to do more complex tests. For load testing Web applications, increase the number of threads and the loop counts gradually and see how your applications cope with the loads. The following sections of this article tackle some other important aspects of load-testing Web applications with JMeter.

Listeners

JMeter comes with a number of listeners or reports. In the previous test, we used a table to display the test results. If this is not suitable for you, you can choose one or more of the other listeners for a thread group. A popular listener is the Graph Results, as shown in Figure 5.

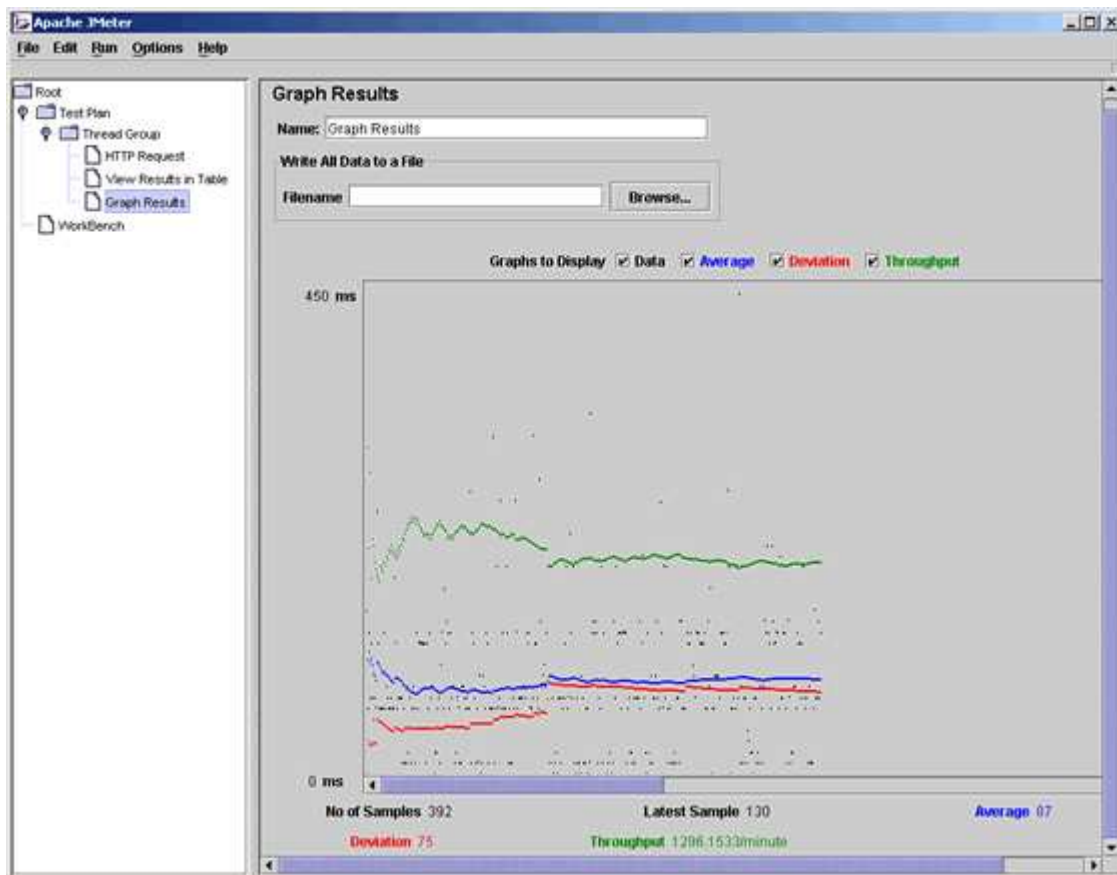


Figure 5: Graph results.

Multiple HTTP Requests

A real application has multiple resources, both static and dynamic. Chances are, you want to see the performance of these resources. JMeter makes it easy to employ multiple HTTP requests. Just add any number of HTTP Request elements and configure them, as in the previous test. If you have multiple HTTP Request elements, you might want to use a HTTP Request Defaults element, described in the following section.

HTTP Request Defaults

The HTTP Request Defaults element specifies the default values of existing HTTP Request elements within the same thread group. The HTTP Request Defaults element is especially useful because most, if not all, HTTP Request elements normally have the same server and port. Figure 6 shows the detail page of a HTTP Request Defaults element.

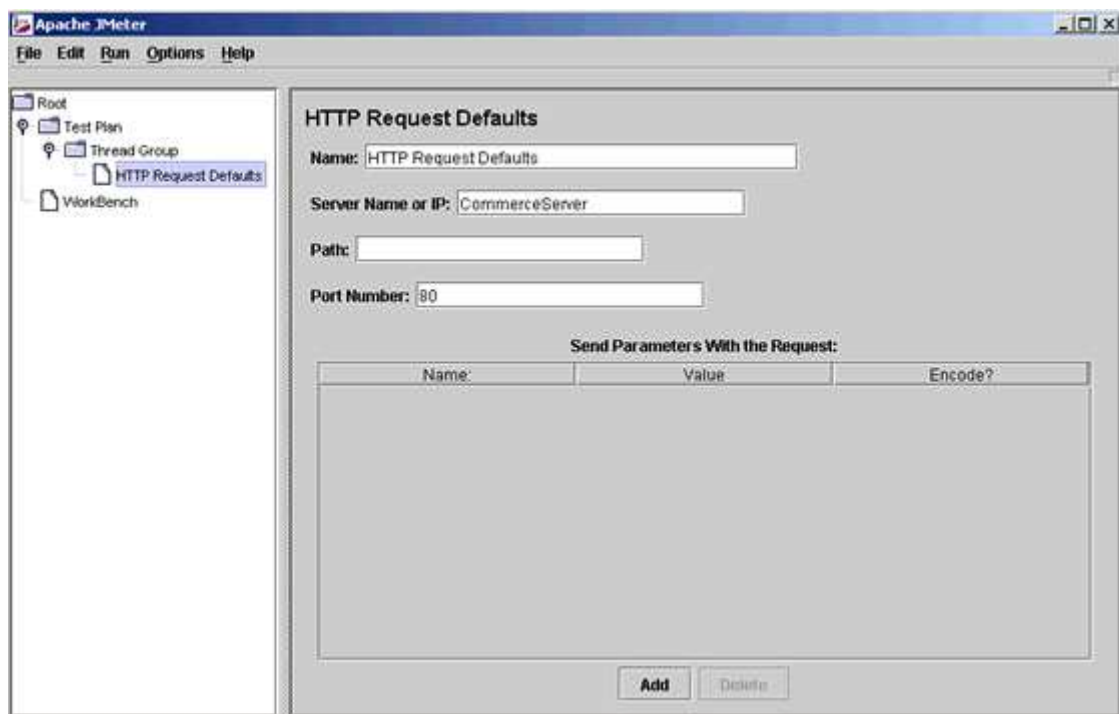


Figure 6: HTTP request defaults.

Add an HTTP Request Default element by right-clicking a Thread Group element and then selecting Add, Config Element, and HTTP Request Default.

Cookie Manager

Many Web applications use cookies. JMeter provides cookie capabilities through a Cookie Manager. Adding this element to a thread group allows you to send cookies to the application being tested, just as Web browsers do. Figure 7 displays the details page of a Cookie Manager. Here you can add and delete a cookie.

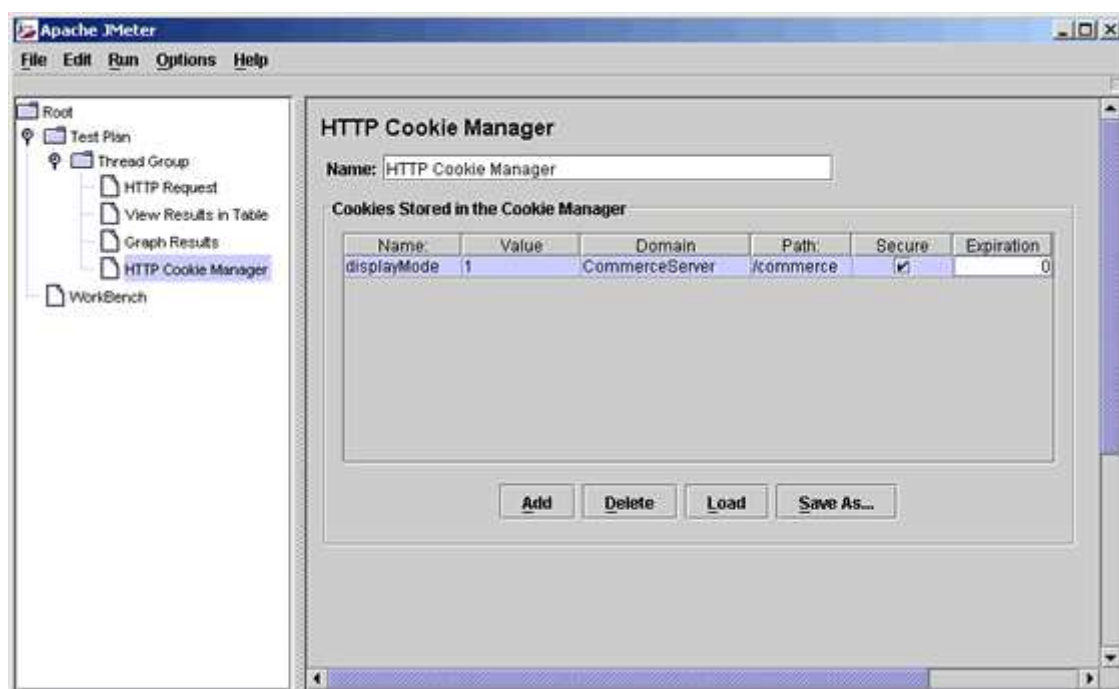


Figure 7; Cookie Manager.

You can add a Cookie Manager element by right-clicking a Thread Group element and then selecting Add,

Config Element, and Cookie Manager.

Summary

JMeter is capable of much more than our simple tests demonstrate. From these building blocks, it's possible to create extensive tests with highly detailed reports. Getting useful results is very easy, though.

For more information, see the [JMeter User Manual](#).

[*Budi Kurniawan*](#) is a senior J2EE architect and author.

Return to [ONJava.com](#).

Copyright © 2007 O'Reilly Media, Inc.