Getting Started with Apache JMeter™

Lab Guide

88BLZ20010

BlazeMeter

Table of Contents

Lab 1- Installing and Configuring

Goals	During this lab, you will learn how to install and configure Apache JMeter™.
Scenario	To do this, you will complete the following tasks: ▪ Download and install the Java SE Development Kit ▪ Download and install Apache JMeter™ ▪ Download and install the Apache JMeter™ Plug-in Manager
Time	20 minutes

Instructions

Click Lab Resources to download the lab resources required to run the Apache JMeter™ labs.

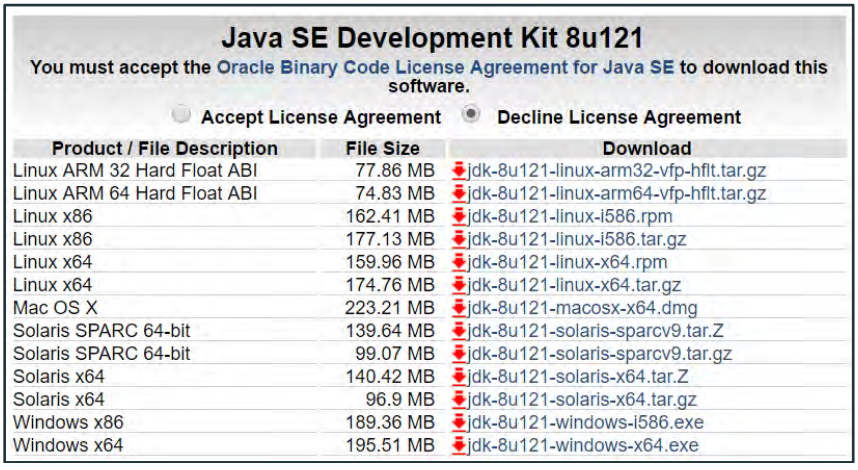
Part 1: Install the Java SE Development Kit

You must install the Java SE Development Kit (JDK) version 7 or later before installing JMeter. Complete these steps to install the JDK:

Note: *If you already have JDK version 7 or newer installed on your system, you can skip Part 1 of this lab.*

1. Go to http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.

2. Click the **Accept License Agreement** radio button in the JDK download section of the web page.



Java SE Development Kit 8u121

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ **Accept License Agreement** ◉ **Decline License Agreement**

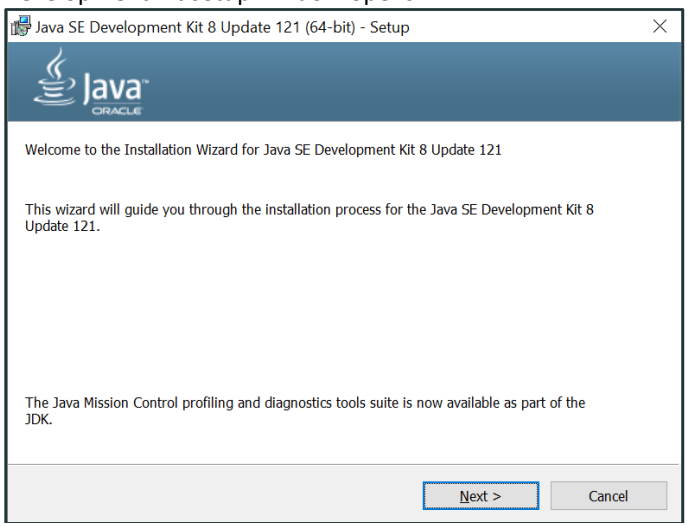
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.86 MB	jdk-8u121-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.83 MB	jdk-8u121-linux-arm64-vfp-hflt.tar.gz
Linux x86	162.41 MB	jdk-8u121-linux-i586.rpm
Linux x86	177.13 MB	jdk-8u121-linux-i586.tar.gz
Linux x64	159.96 MB	jdk-8u121-linux-x64.rpm
Linux x64	174.76 MB	jdk-8u121-linux-x64.tar.gz
Mac OS X	223.21 MB	jdk-8u121-macosx-x64.dmg
Solaris SPARC 64-bit	139.64 MB	jdk-8u121-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.07 MB	jdk-8u121-solaris-sparcv9.tar.gz
Solaris x64	140.42 MB	jdk-8u121-solaris-x64.tar.Z
Solaris x64	96.9 MB	jdk-8u121-solaris-x64.tar.gz
Windows x86	189.36 MB	jdk-8u121-windows-i586.exe
Windows x64	195.51 MB	jdk-8u121-windows-x64.exe

3. Click the correct version of JDK for your operating system to download the installation files.

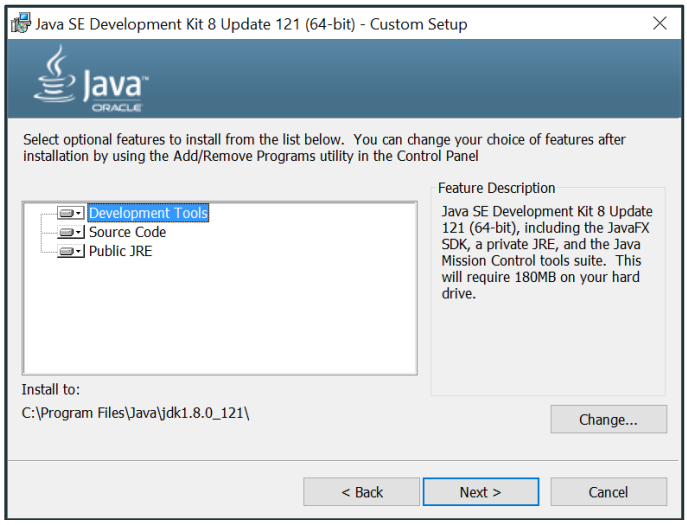
4. When the file is finished downloading, double-click it to begin installing JDK. The User Account Control window

opens.

5. Click **Yes** in the User Account Control window to allow this app to make changes to your device. The Java SE Development Kit setup window opens.



6. Click **Next** to start the installation wizard. The Custom Setup window opens.



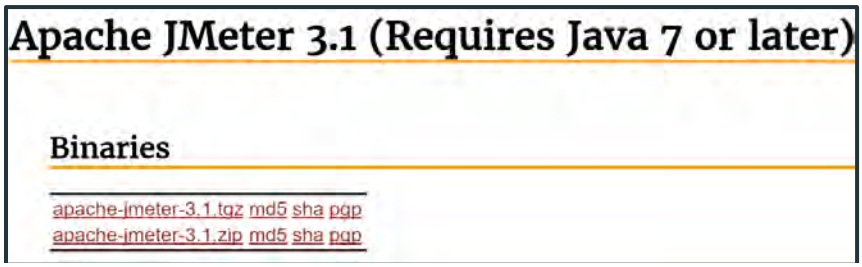
7. Click **Next**. The Progress window opens.

8. The Complete window opens when the installation finishes. Click **Close**.

Part 2: Install JMeter

Complete these steps to install JMeter from the Apache JMeter website:

1. Open http://jmeter.apache.org/download_jmeter.cgi to view the Download Apache JMeter web page.
2. Click **apache-jmeter-3.1.zip** to download the JMeter installation files.



3. When the download is complete, double-click **apache-jmeter-3.1_src** to open the zip file. Your application for managing zip files opens.

   Note: *If you do not have an application installed for opening zip files, you can download and install a trial version if the WinRAR archiver application at http://www.rarlab.com/.*

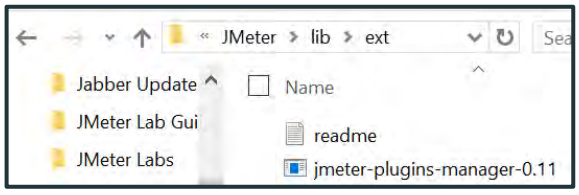
4. Create a new folder named JMeter on your desktop and extract the files to the folder.
5. Open your **JMeter** folder and then open the **bin** folder.
6. In the bin folder, locate the JMeter Windows batch file.
7. Right-click the **JMeter Windows batch file** and select **Send to ▶ Desktop** to create a shortcut to JMeter on your desktop.

Part 3: Install the JMeter Plugin Manager

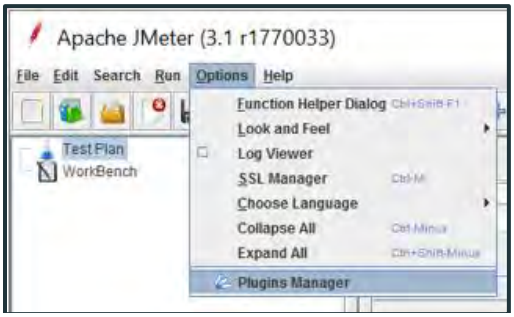
Complete these steps to install the JMeter Plugin Manager:

1. Go to https://jmeter-plugins.org/downloads/all/ to view the Plugins Manager Download page.
2. Click **plugins-manager.jar** in the middle of the page to download the installation files.

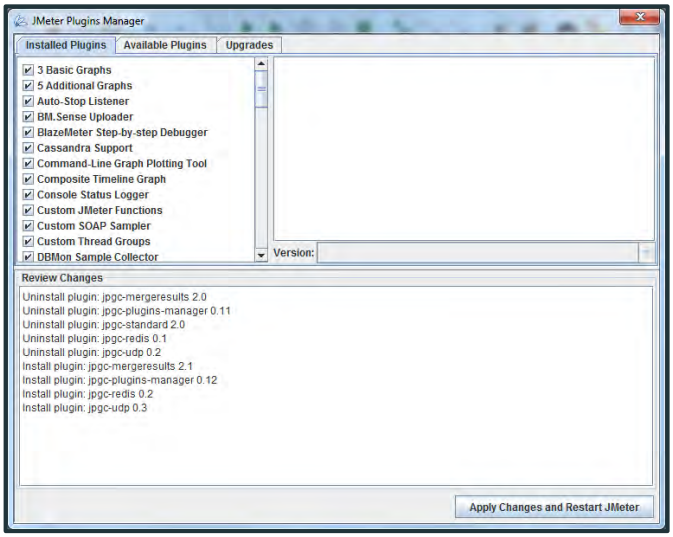
3. When the download is complete, move the file to the **lib/ext** directory of your JMeter folder.



4. Launch JMeter and select **Options** ▶ **Plugin Manager** to verify that the Plugin Manager launches properly.



5. You have finished this lab when the JMeter Plugins Manager opens properly and displays a list of already installed plugins in the Installed Plugins tab.



Lab 2: Working with Thread Groups

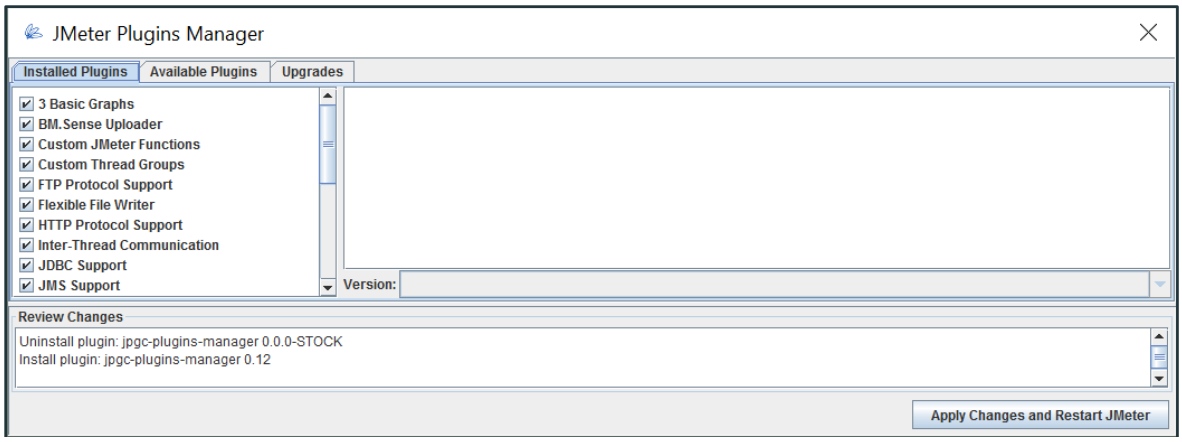
Goals	During this lab, you will learn how to configure Thread Groups and Ultimate Thread Groups.
Scenario	A Thread Group is one of the most basic elements of a JMeter script. You cannot run a JMeter script without a Thread Group. The Thread Group initiates the thread that runs the scenario that you specify in the script. In this lab, we examine two types of thread groups: standard and ultimate. The standard Thread Group is included in the JMeter installation by default. It is simple to configure and includes a ramp up time, loop count, and duration. The Ultimate Thread Group is more advanced and allows you to specify the ramp down time and the ramp up time. An Ultimate Thread Group can also be used to configure more complex thread scenarios. Complete these tasks to learn how to configure Thread Groups and Ultimate Thread Groups: ▪ Install the Dummy Sampler and the jpgc - Standard Set Plugins ▪ Create a new JMeter test ▪ Add a Thread Group ▪ Add a Dummy Sampler ▪ Configure the Thread Group ▪ Add an Ultimate Thread Group ▪ Add an Additional Dummy Sampler ▪ Add a View Results Tree listener and run the test
Time	20 minutes

Install the Dummy Sampler and jpgc - Standard Set Plugins

Complete these steps to install the Dummy Sampler and the jpgc – Standard Set plugins:

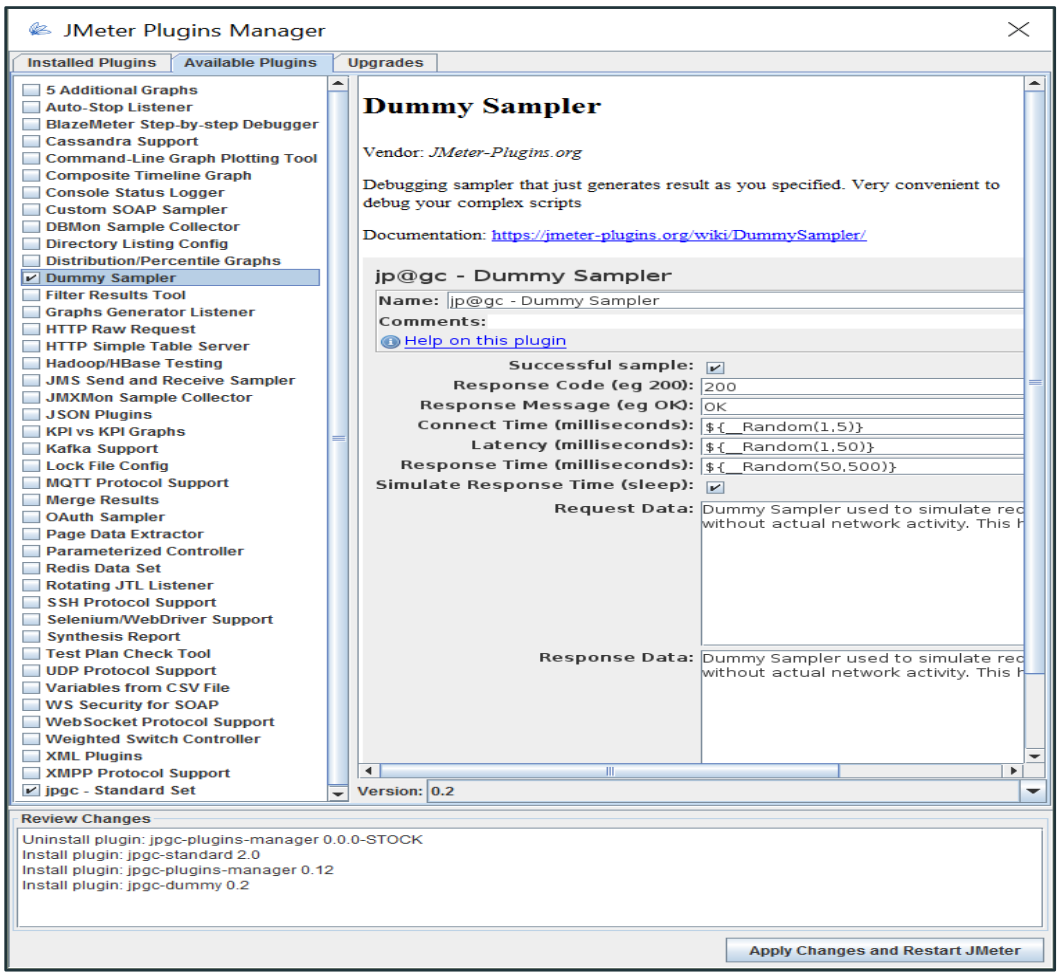
Note: *The jpgc – Standard Set plugin includes the Ultimate Thread Group that you will use in this lab.*

1. Launch **JMeter** from your desktop shortcut.
2. Select **Options** ▶ **Plugin Manager**. The JMeter Plugins Manager opens.



3. Click the **Available Plugins** tab.

4. Check the **Dummy Sampler** and the **jpgc - Standard Set** checkboxes in the list of available plugins.

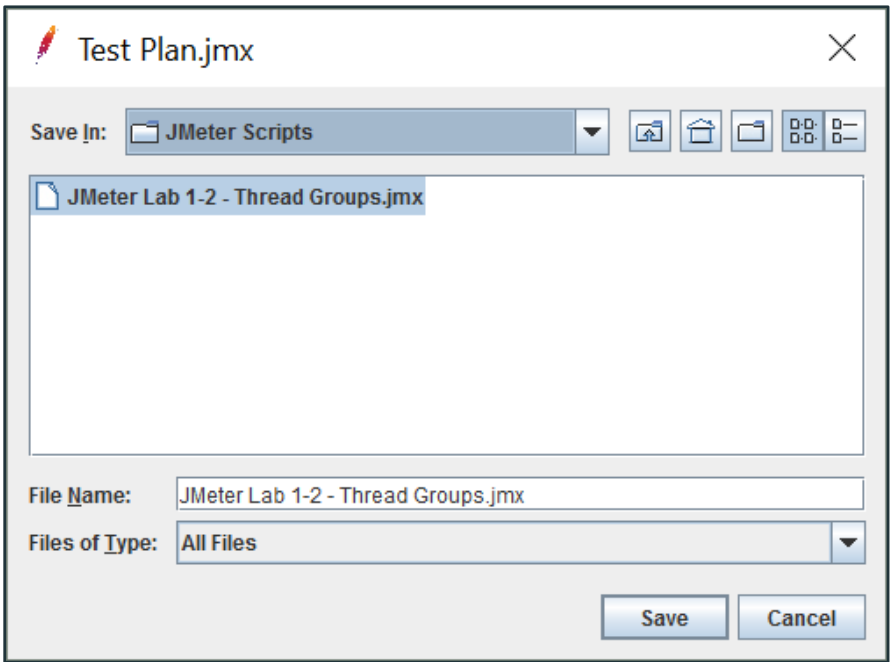


5. Click **Apply Changes and Restart JMeter** to install these plugins. JMeter installs the plugins, closes, and restarts automatically.

Create a New JMeter Test

Complete these steps to save your JMeter test and your new configuration:

6. Select **File ▶ Save Test Plan as**.
7. Select **Desktop** from the Save In dropdown menu.
8. Click **New Folder** to create a new folder on your desktop and name the folder **JMeter Scripts**.
9. Enter **JMeter 1-2 - Thread Groups.jmx** in the File Name field.

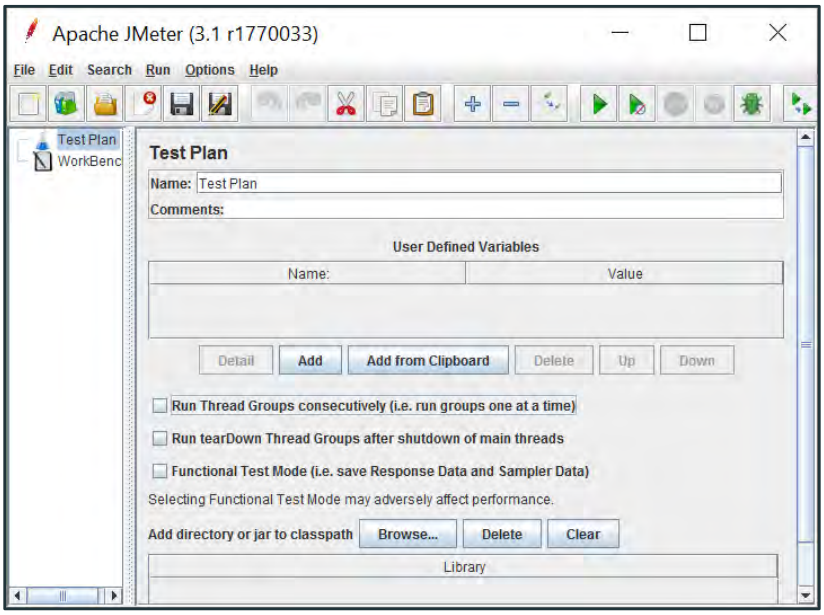


10. Click **Save** to save your test script.

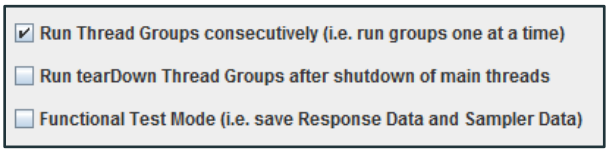
Add a Thread Group

Complete these steps to add a Thread Group to your JMeter test:

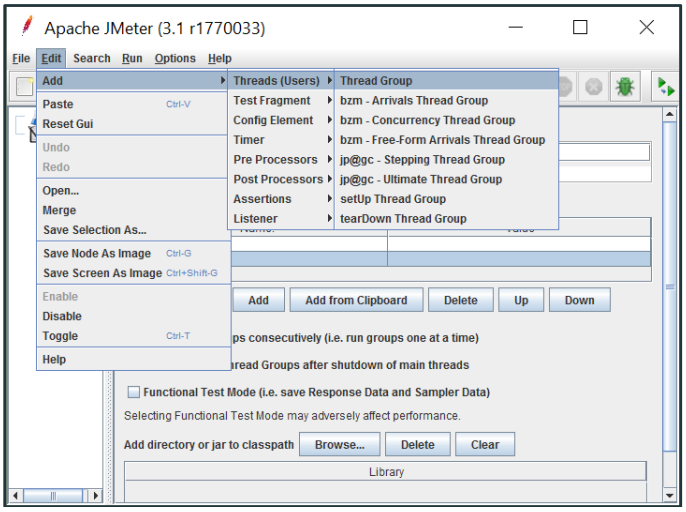
11. Click **Test Plan** to view the Test Plan pane.



12. Check the **Run Thread Groups consecutively** checkbox in the center of the screen.



13. Select **Edit** ▶ **Add** ▶ **Threads (Users)** ▶ **Thread Group** to create a new thread group. The Thread Group configuration pane opens.

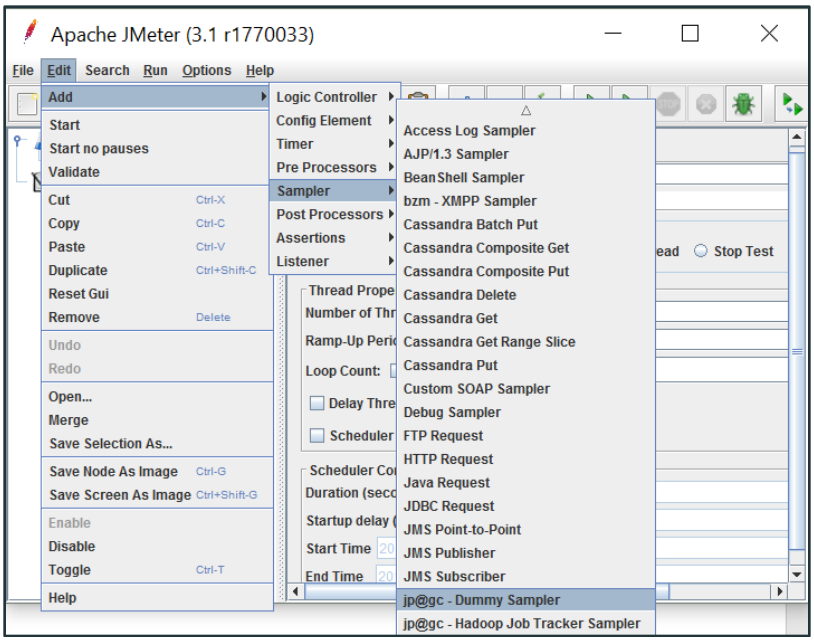


Note: *By default, a Thread Group is configured to have one thread with a ramp-up period of one second and a loop count of one.*

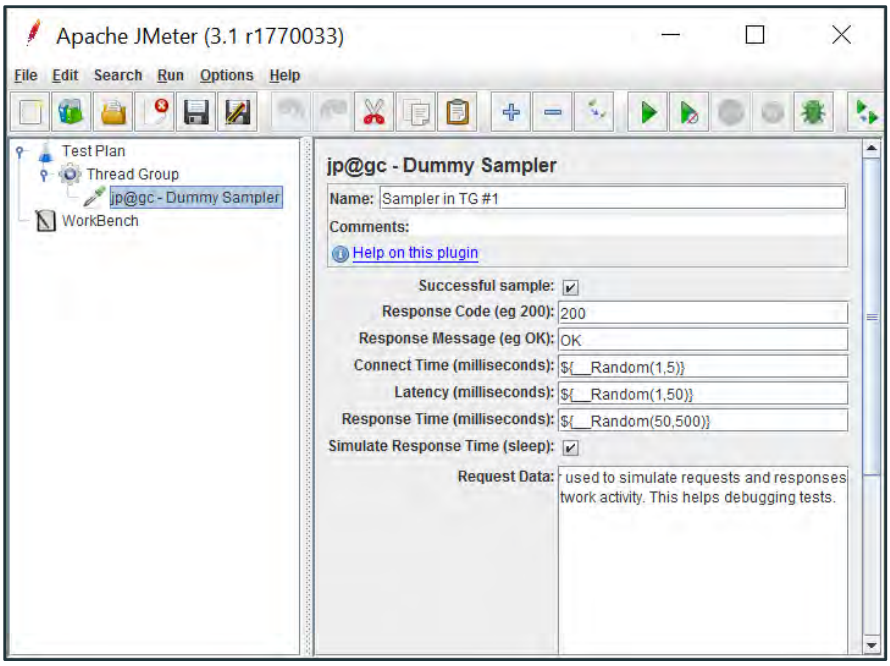
Add a Dummy Sampler

Complete these steps to add a Dummy Sampler to your JMeter test:

14. Select **Edit ▶ Add ▶ Sampler ▶ Dummy Sampler** to add a Dummy Sampler to your Thread Group. The **Dummy Sampler** configuration pane opens.



15. Change the name of the dummy sampler to **Sampler in TG #1** to indicate that this is the Dummy sampler for Thread Group one.



Configure the Thread Group

Complete these steps to configure the Thread Group:

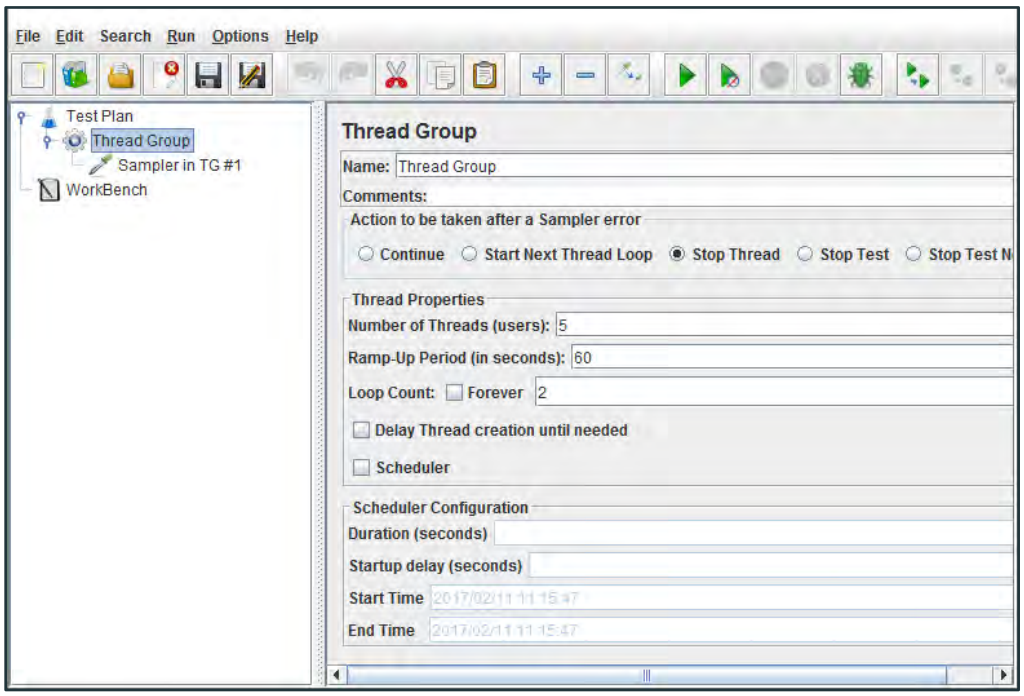
16. Click **Thread Group** in the Test Plan pane to view the thread group configuration screen.

17. From the Thread Group screen, select the **Stop Thread** radio button in the Action to be taken after a Sampler error section of the screen.

18. Configure the Thread Properties section of the screen as follows:

Field	Value
Number of Threads	**5**
Ramp-up Period	**60**
Loop Count	**2**

Your Thread Group configuration should look like this:

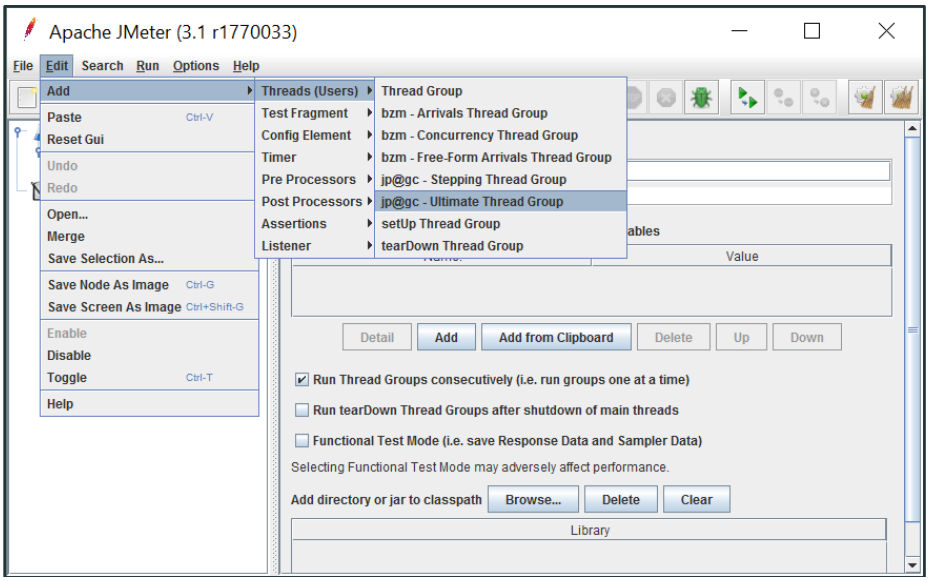


Add an Ultimate Thread Group

Complete these steps to add an Ultimate Thread Group to your test:

19. Click **Test Plan** to highlight it.

20. Right-click **Test Plan** and select **Add ▶ Threads (Users) ▶ Ultimate Thread Group**. The Ultimate Thread Group configuration pane opens.



21. Click **Add Row** three times to add three rows to your Ultimate Thread Group.

a. Configure row one as follows:

Field	Value
Start Threads Count	**3**
Initial Delay	**0**
Startup Time	**0**
Hold Load For	**900**
Shutdown Time	**300**

b. Configure row two as follows:

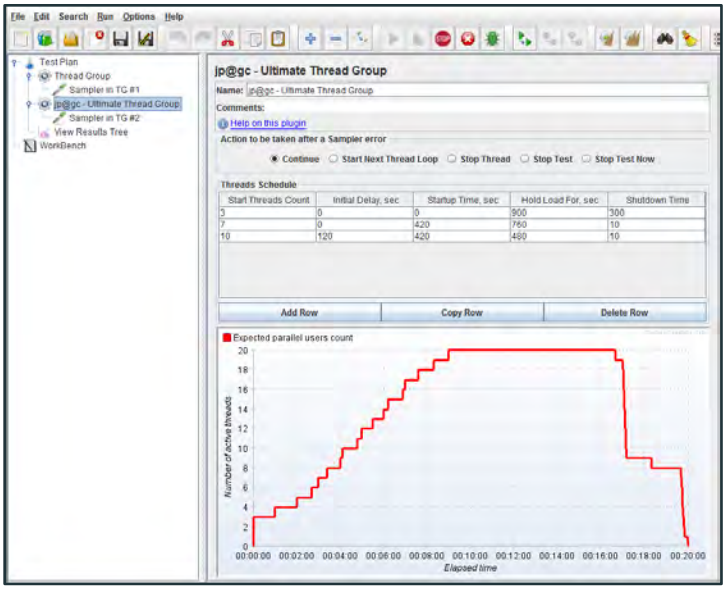
Field	Value
Start Threads Count	**7**
Initial Delay	**0**
Startup Time	**420**
Hold Load For	**760**
Shutdown Time	**10**

Note: *This test begins with three threads. The test will reach ten threads at two minutes.*

c. Configure row three as follows:

Field	Value
Start Threads Count	**10**
Initial Delay	**120**
Startup Time	**420**
Hold Load For	**480**
Shutdown Time	**10**

Your Ultimate Thread Group configuration should look like this:



Add an Additional Dummy Sampler

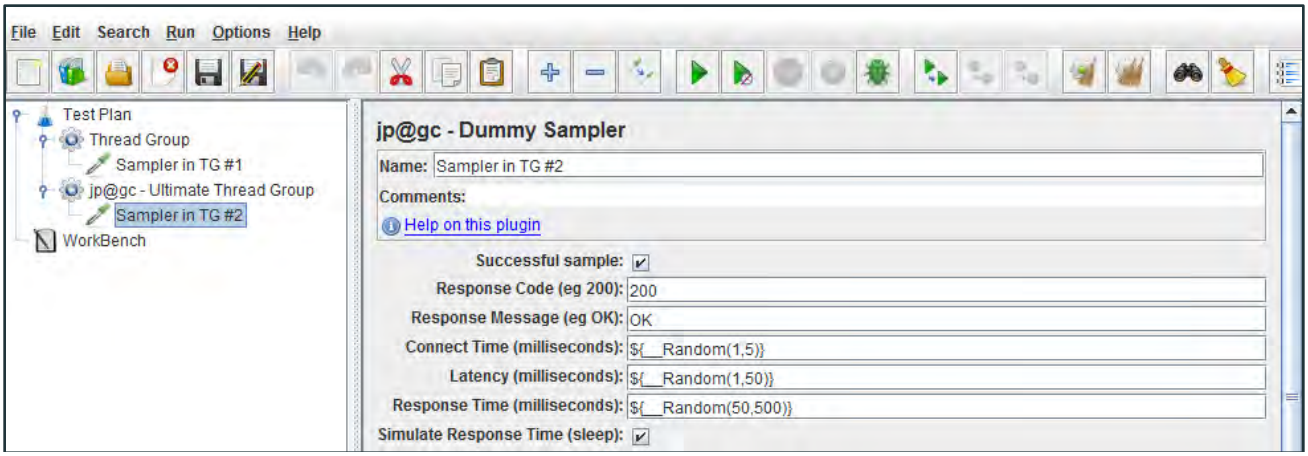
Complete these steps to add another Dummy Sampler and a View Results Tree listener to your test so that you can run the test and view the results:

22. Click **Ultimate Thread Group** to highlight it.

23. Right-click **Ultimate Thread Group**.

24. Select **Add ▶ Sampler ▶ Dummy Sampler**. The Dummy Sampler configuration pane opens.

25. Enter **Sampler in TG #2** in the name field to indicate that this is a Dummy sampler in Thread Group two.



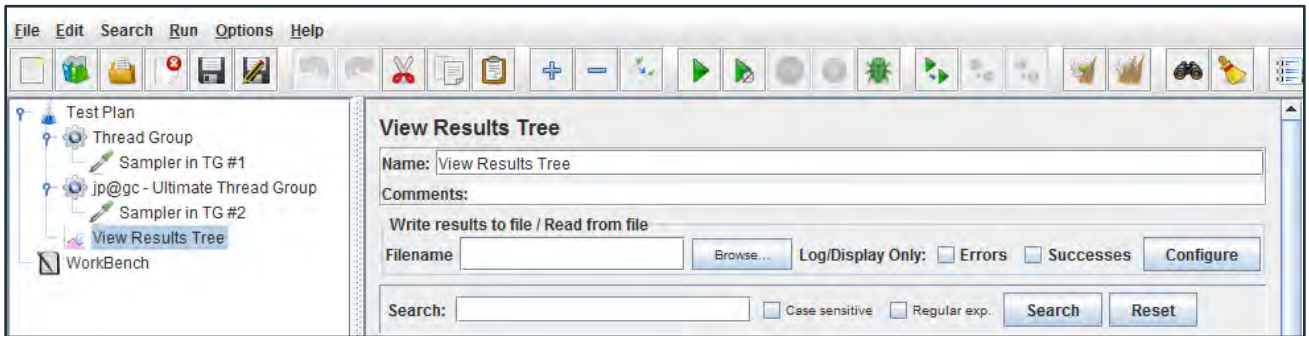
Add a View Results Tree Listener and Run the Test

Complete these steps to add a View Results Tree listener:

26. Click **Test Plan** to highlight it.

27. Right-click **Test Plan** in the left pane.

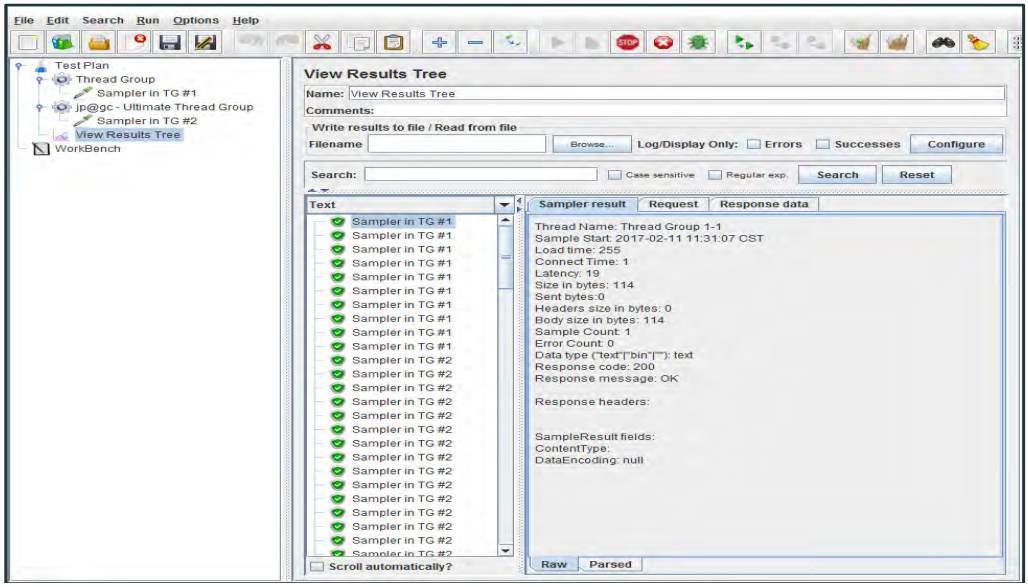
28. Select **Add** ▶ **Listener** ▶ **View Results Tree**. The View Results Tree configuration pane opens.



29. Click **Save**.

30. Click **Start** in the toolbar to run the test.

31. Click **View Results Tree** in the left pane. The View Results Tree pane opens and displays your test results for Thread Groups one and two as they are generated.



Lab 3 – Creating a Basic User Scenario

Goals

During this lab, you will learn how to simulate Internet browser behavior by using a cookie manager and a cache manager. You will then test your Internet browser response time.

Scenario

In this scenario, you will create a basic, realistic user scenario in which multiple virtual users enter a website and navigate the site. You will do this by sending HTTP requests to a web page. You will then send a second HTTP request to a second page.

Using the Cookie Manager and the Cache Manager, you will configure JMeter to behave like a real browser. You can clear your cache and cookies with each iteration to imitate the behavior of multiple users entering the website. At the end of the lab, you will run your test and view results for multiple users, noting the maximum response times.

Complete these tasks to configure and test this basic scenario:

- Create a new JMeter test
- Add a Thread Group
- Add an HTTP Cache Manager
- Add a Cookie Manager
- Add an HTTP Request Defaults element
- Add a View Results Tree listener
- Execute an HTTP Request and configure a Constant Timer
- Execute another HTTP Request and configure a Constant Timer
- Run the test and view the results
- Add a Response Time Listener and generate an Aggregate Report

Time

20 minutes

Create a New JMeter Test

Complete these steps to create a new JMeter test:

1. Launch **JMeter**.
2. Select **File ▶ Save Test Plan As**. The Test Plan.jmx window opens.

3. Enter **JMeter Lab 1-3 - Basic Scenario.jmx** in the File Name field and click **Save**.
   Note: *Save the lab periodically throughout this exercise to preserve your work.*

Add a Thread Group

Complete these steps to add a Thread Group to your test:

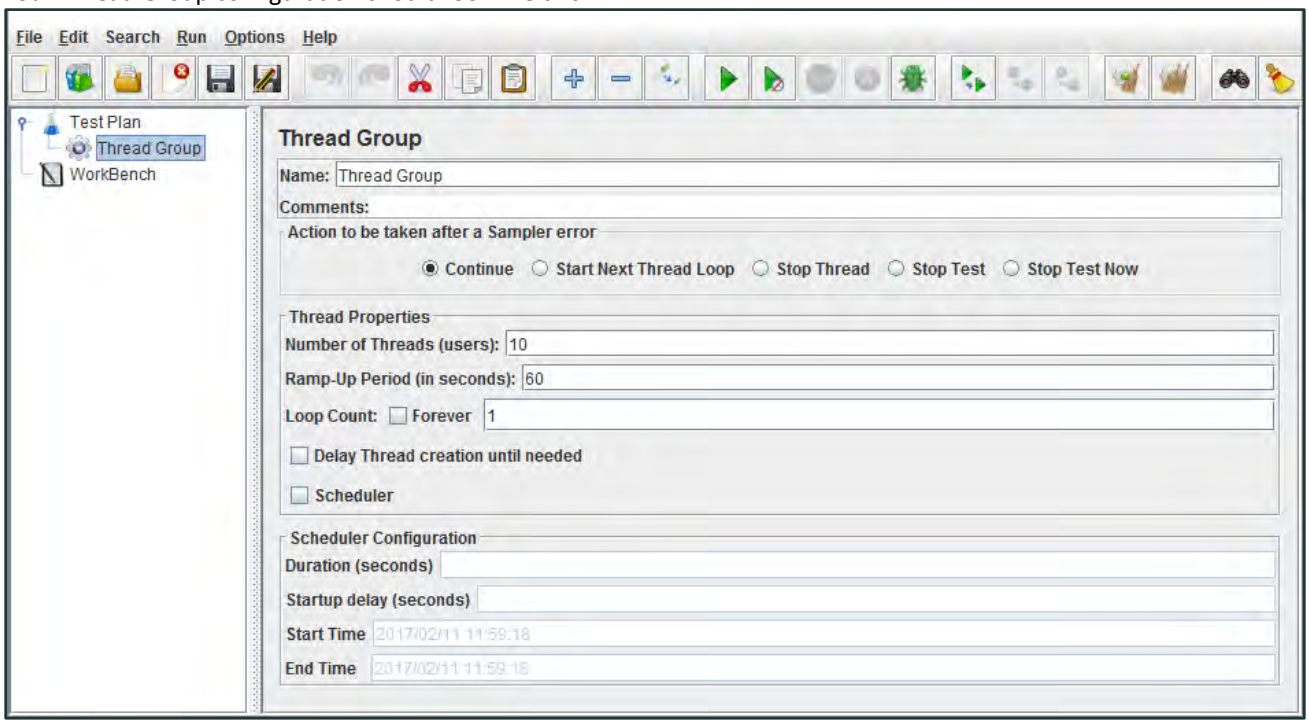
4. Click **Test Plan** to highlight it.

5. Right-click **Test Plan** and select **Add ▶ Threads (Users) ▶ Thread Group**. The Thread Group configuration pane opens.

6. Configure your Thread Group as follows:

Field	Value	Notes
Number of Threads	**10**	
Ramp-Up Period	**60**	This is how long it will take for all the threads to start running. The last thread will start at 60 seconds.

Your Thread Group configuration should look like this:



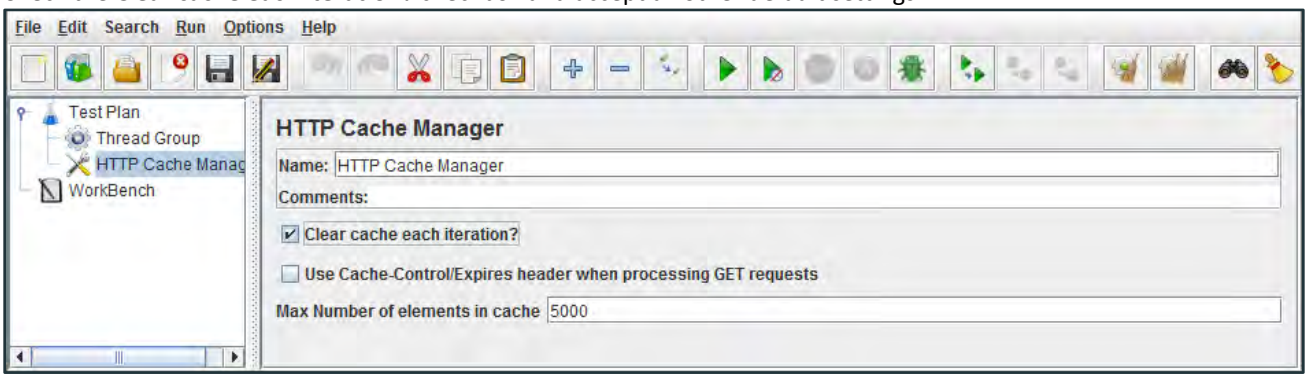
Add an HTTP Cache Manger

Complete these steps to add an HTTP Cache Manager to your Test Plan:

7. Click **Test Plan** to highlight it.

8. Right-click **Test Plan** and select **Add ▶ Config Element ▶ HTTP Cache Manager** to add a cache manager to the test.

9. Check the **Clear cache each iteration?** checkbox and accept all other default settings.



Add an HTTP Cookie Manger

Complete these steps to add an HTTP Cookie Manager to your Test Plan:

10. Click **Test Plan** to highlight it.

11. Right-click **Test Plan** and select **Add ▶ Config Element ▶ HTTP Cookie Manager** to add a Cookie Manager to the test.

12. Check the **Clear cache each iteration?** checkbox and accept all other default settings.

Note: *Adding a cache manager and a cookie manager to the test plan allows you to better simulate real Internet browser behavior.*

Add an HTTP Request Defaults Element

Complete these steps to add an HTTP Request Defaults element to your Test Plan:

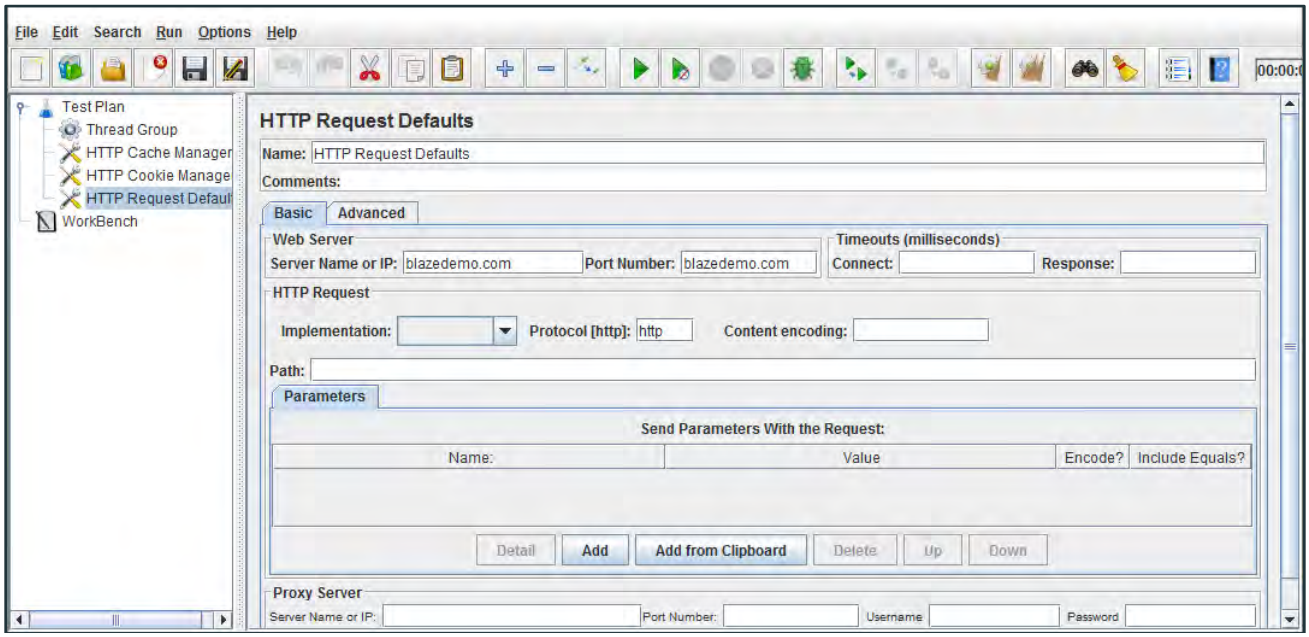
13. Click **Test Plan** to highlight it.

14. Right-click **Test Plan** and select **Add ▶ Config Element ▶ HTTP Request Defaults**.

15. Configure the HTTP Request Defaults element as follows:

Field	Value	Notes
Server Name or IP	**blazedemo.com**	
Port Number	**blazedemo.com**	
Protocol	**http**	Accept default value of http.

Your HTTP Request Defaults configuration should look like this:



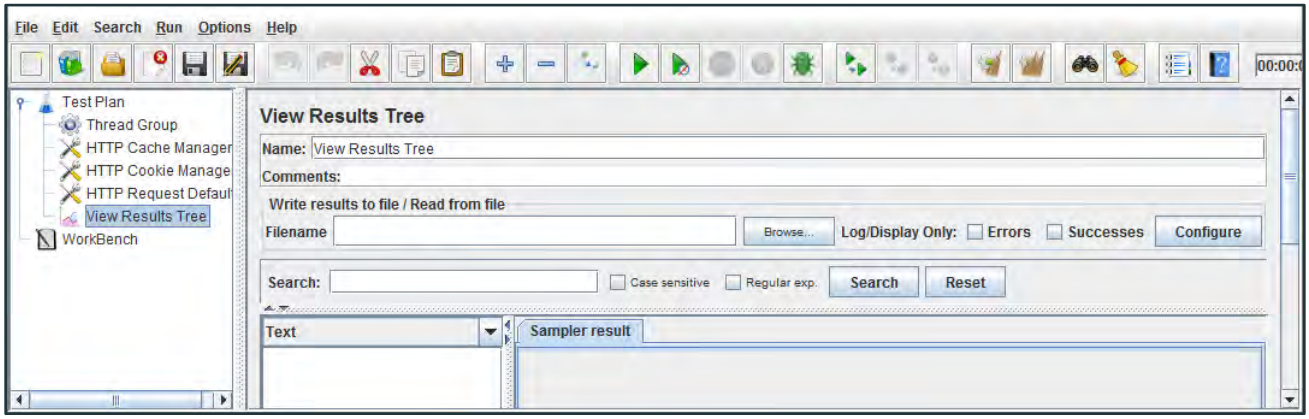
Note: *You can also enter the protocol in the Server Name or IP field and leave the Protocol field blank. For example, enter* ***http://blazedemo.com*** *in the Server Name or IP field.*

Add a View Results Tree Listener

Complete this steps to add a View Results Tree listener that will capture and display your test results:

16. Click **Test Plan** to highlight it.

17. Right-click **Test Plan** and select **Add ▶ Listener ▶ View Results Tree** to listen to traffic. The View Results Tree configuration pane opens.

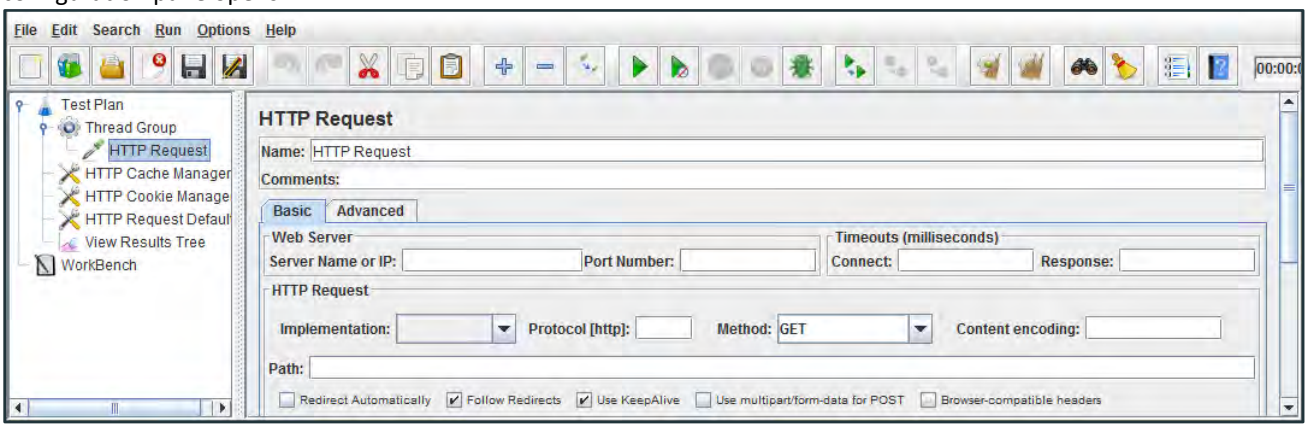


Execute an HTTP Request and Configure a Constant Timer

Complete these steps to execute an HTTP Request and configure a Constant Timer:

18. Click **Thread Group** to highlight it.

19. Right-click **Thread Group** in the left pane and select **Add ▶ Sampler ▶ HTTP Request**. The HTTP Request configuration pane opens.

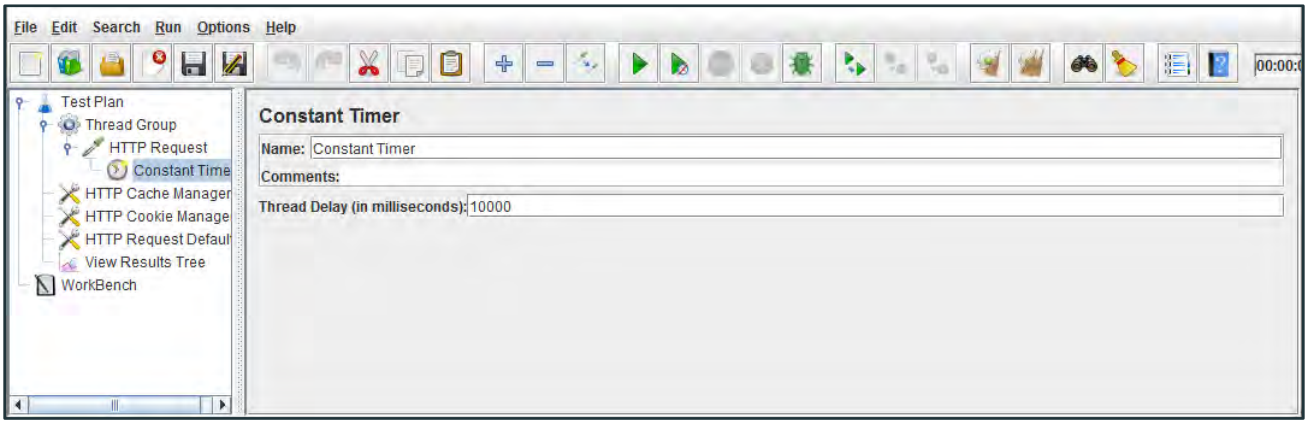


Note: *When the fields of the HTTP Request configuration pane are left blank, the HTTP Request Default values are automatically applied. For example, the HTTP Request will use the Server Name blazedemo.com provided in the Server Name field of the HTTP Request Defaults configuration pane.*

20. Right-click **HTTP Request** in the left pane under Thread Group.

21. Select **Add ▶ Timer ▶ Constant Timer** from the right-click menu. The Constant Timer configuration pane displays.

22. Enter **10000** in the Thread Delay (in milliseconds) field to run the timer for ten seconds.



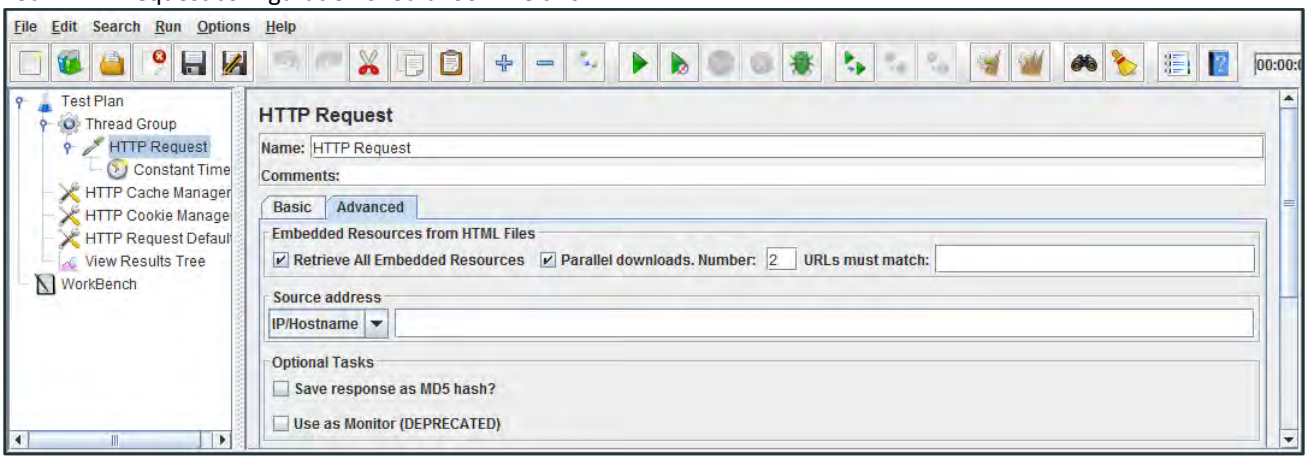
23. Click **HTTP Request** in the left pane. The HTTP Request configuration pane opens

24. Select the **Advanced** tab. The HTTP Request Advanced configuration pane opens.

25. Configure the Advanced tab of your HTTP Request as follows:

Field	Value
Retrieve All Embedded Resources	**Checked**
Parallel downloads	**Checked**
Parallel downloads, Number	**2**

Your HTTP Request configuration should look like this:



Execute Another HTTP Request and Configure a Constant Timer

In this section, you will execute another HTTP Request that will wait three seconds before executing. The delay will occur before the previous request.

Complete these steps to execute another HTTP Request and configure a Constant Timer:

26. Click **Thread Group** to highlight it.

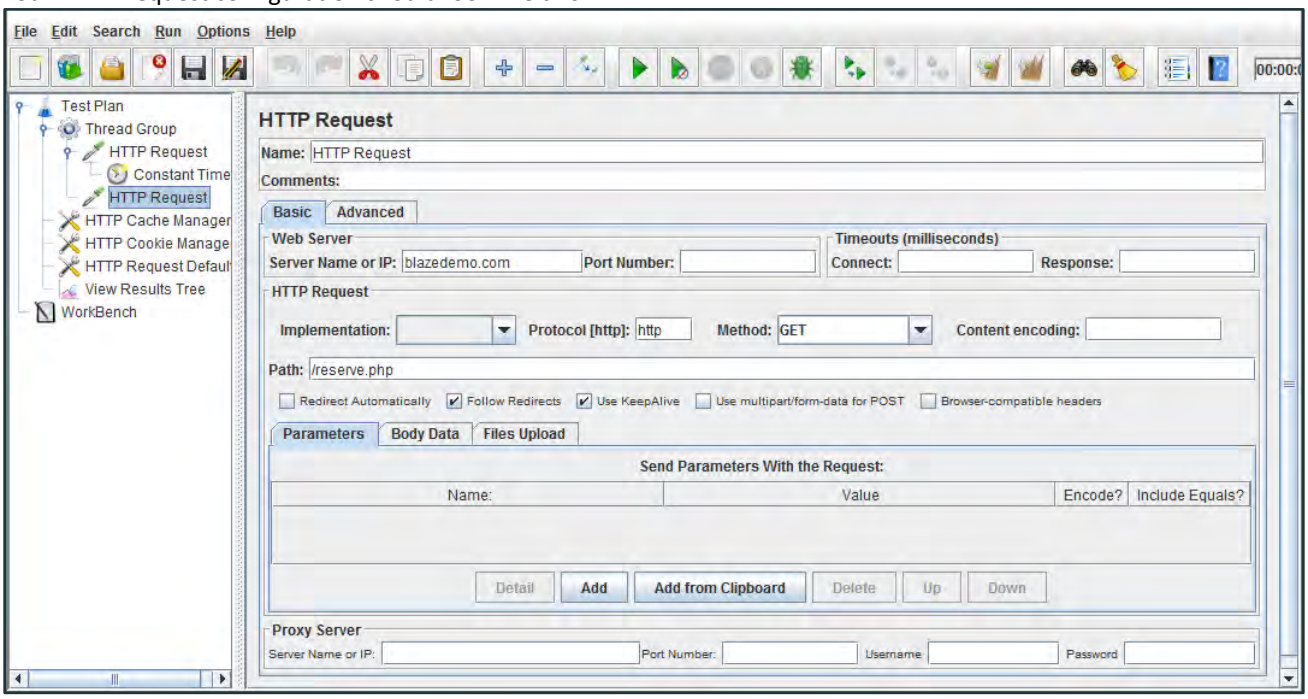
27. Right-click **Thread Group** in the left pane and select **Add ▶ Sampler ▶ HTTP Request**. The new HTTP Request configuration page opens.

28. Click the **Basic** tab.

29. Configure the Basic tab of your HTTP Request as follows:

Field	Value	Notes
Server or IP	**blazedemo.com**	
Method	**GET**	Accept the default value of Get.
Protocol	**http**	Accept the default value of http.
Path	**/reserve.php**	

Your HTTP Request configuration should look like this:

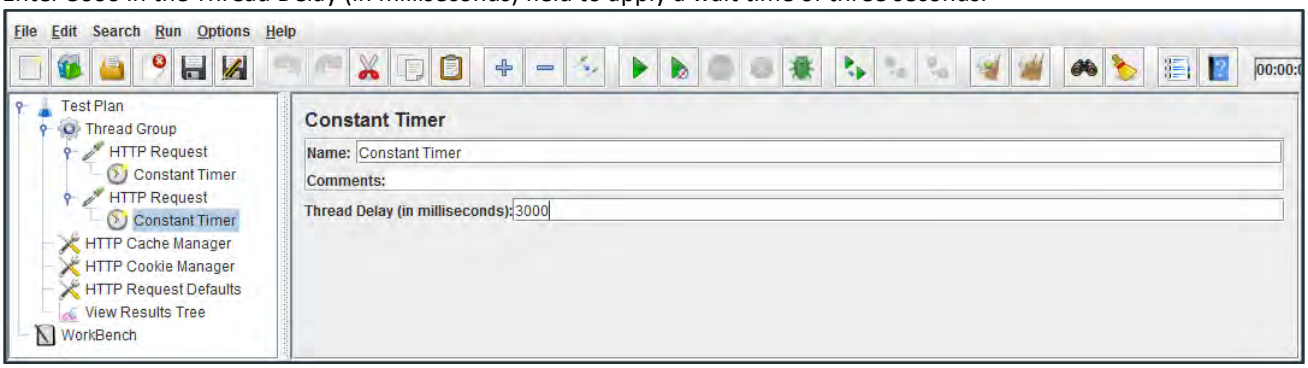


30. Click the **HTTP Request** to select it.

31. Right-click the new **HTTP Request**.

32. Select **Add ▶ Timer ▶ Constant Timer**. The Constant Timer configuration pane opens.

33. Enter **3000** in the Thread Delay (in milliseconds) field to apply a wait time of three seconds.



Run the Test and the View Results

Complete these steps to add a Response Time Listener, run your test, and note your test results:

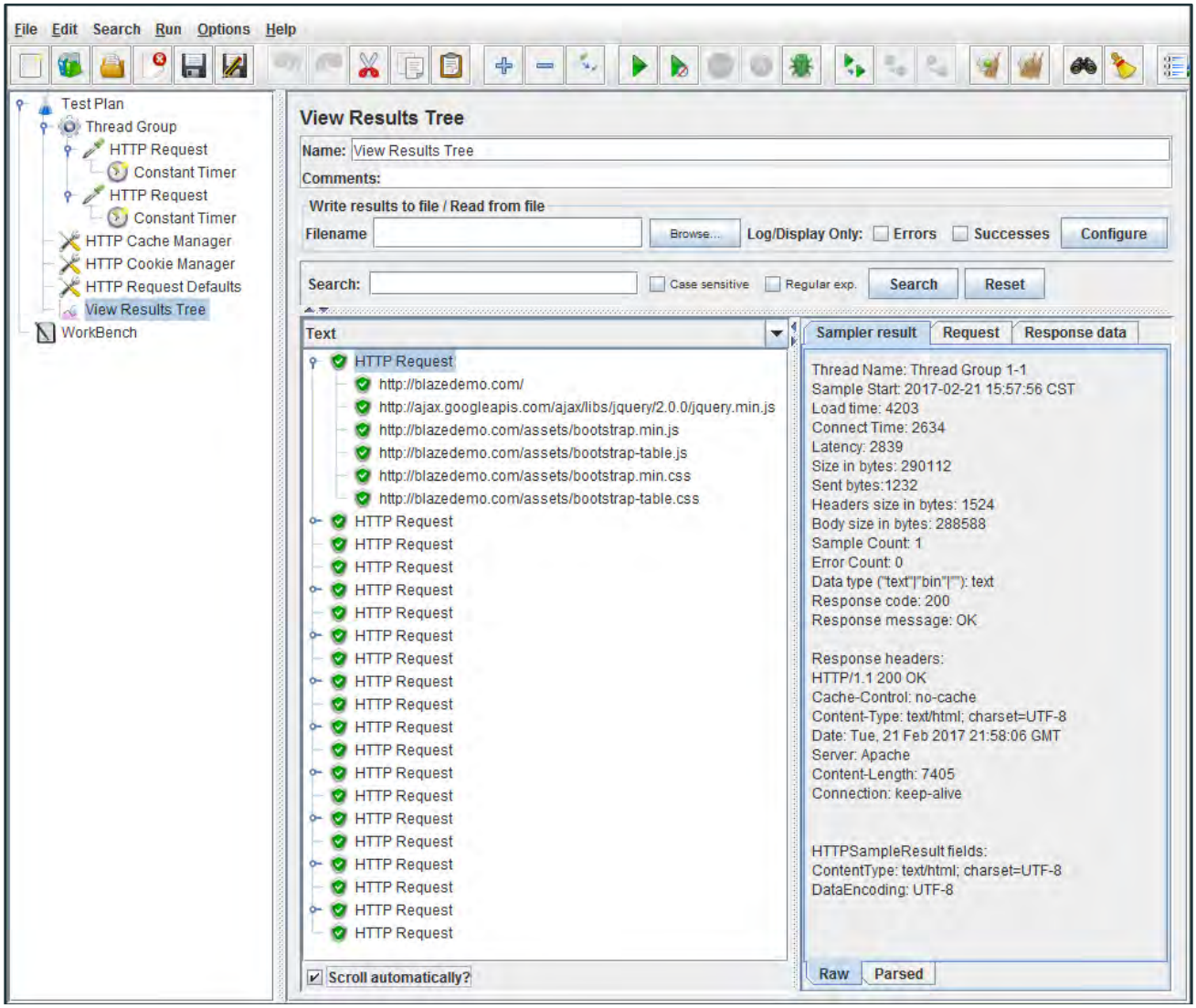
34. Click **Save**.

35. Click **View Results Tree** in the left pane. The View Results Tree configuration pane opens and the test results begin to display.

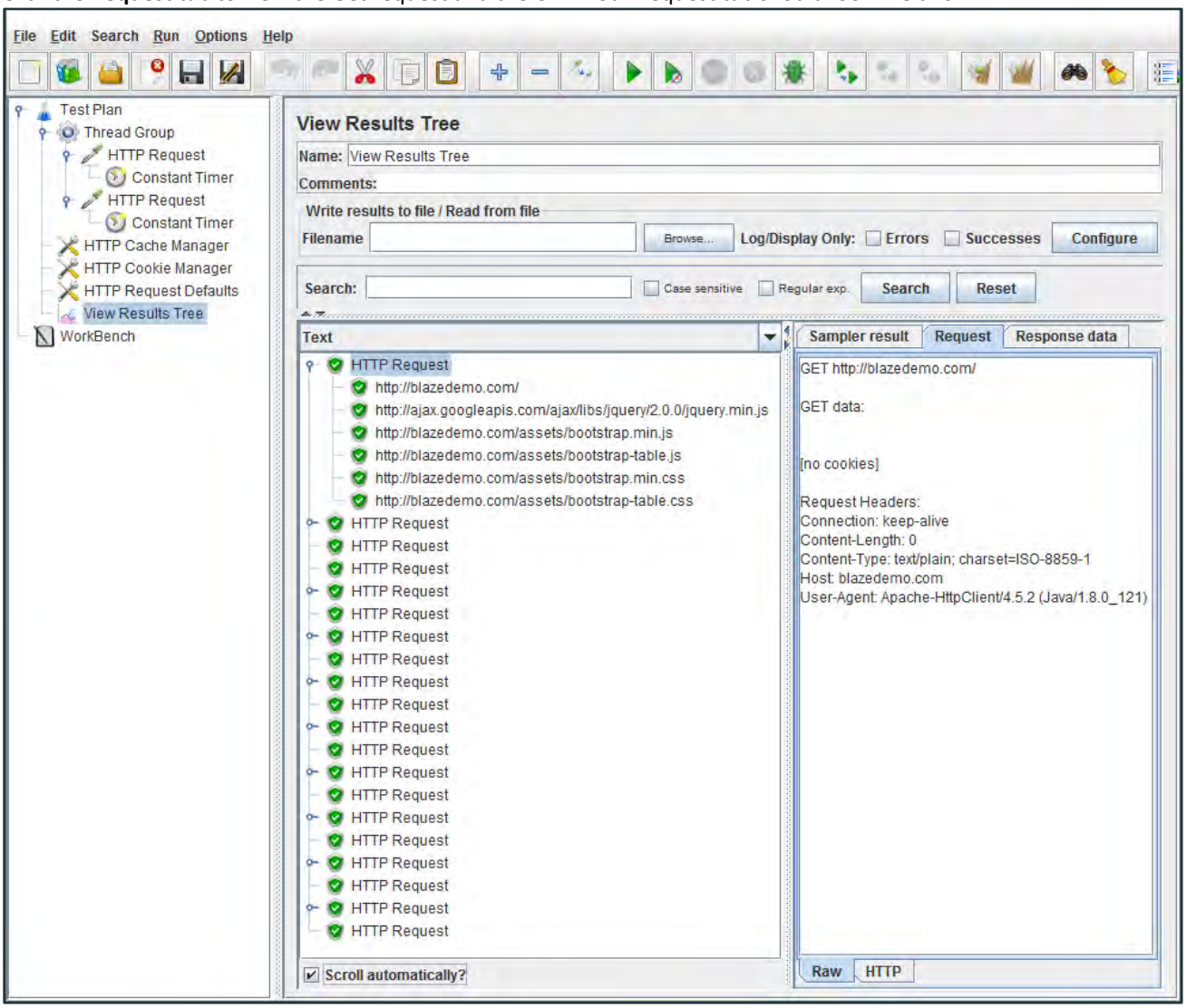
    Note: *Each thread executes two http requests. You should see the results for twenty total http requests.*

36. Click **Start** in the toolbar.

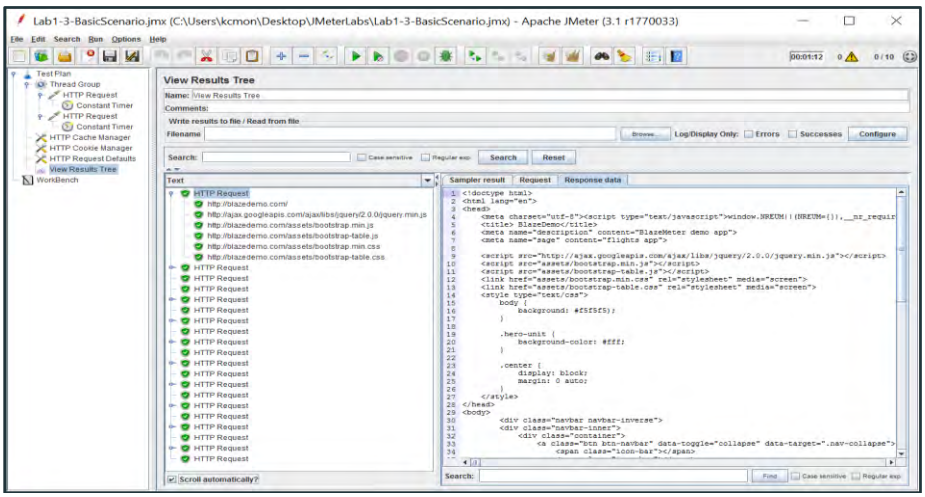
    Your test results should look like this:



37. Click the **Request** tab to view the Get request and the URL. Your Request tab should look like this:



38. Click the **Response data** tab to view the response data. What was the maximum response time in your test? The response time is also known as the load time. You can manually check the load time for each http request to discover the maximum load time. You can also add an additional listener to capture this information.



Add a Response Time Listener and Generate an Aggregate Report

Complete these steps to add a Response Time Listener, run your test again, and note your test results:

39. Click **Test Plan** to highlight it.

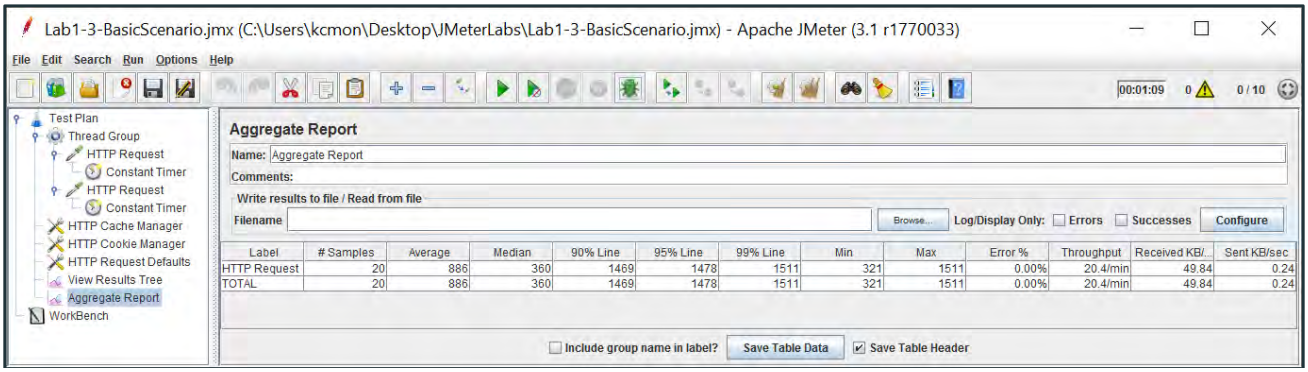
40. Right-click **Test Plan** in the left pane.

41. Select **Add ▶ Listener ▶ Aggregate Report**. The Aggregate Report configuration pane opens.

42. Click **Save**.

43. Click **Start** in the tool bar to run the test again.

44. As data appears in the Aggregate Report pane, note the maximum response time in milliseconds in the Max column of the report.



Lab 4 – Utilizing Assertions

Goals	During this lab, you will learn how to use a JSON Assertion, a Response Assertion, and a Duration Assertion.
Scenario	Assertions are used to verify that you accessed the web page that you wanted to access and that you received the response and the value that you wanted to receive. For example, when you login to a website, you see welcome text with your username on the home page. If you want to verify that the sign in completed properly, you may verify that the word “welcome” is included in the sign in request response. During this lab, you will complete these tasks: ▪ Use a JSON Assertion to verify information ▪ Use a JSON Assertion to verify information after modifying the expected value ▪ Apply a Response Assertion to verify text in a response ▪ Utilize a Response Assertion to verify receipt of a code ▪ Verify an outcome with a Duration Assertion
Time	20 minutes

Part 1: Use a JSON Assertion to Verify Information

In Part 1 of this lab, you will use a JSON Assertion to verify information. A JSON Assertion is a dedicated and more convenient method for handling JSON responses that are common and popular in web apps.

Create a New JMeter Test

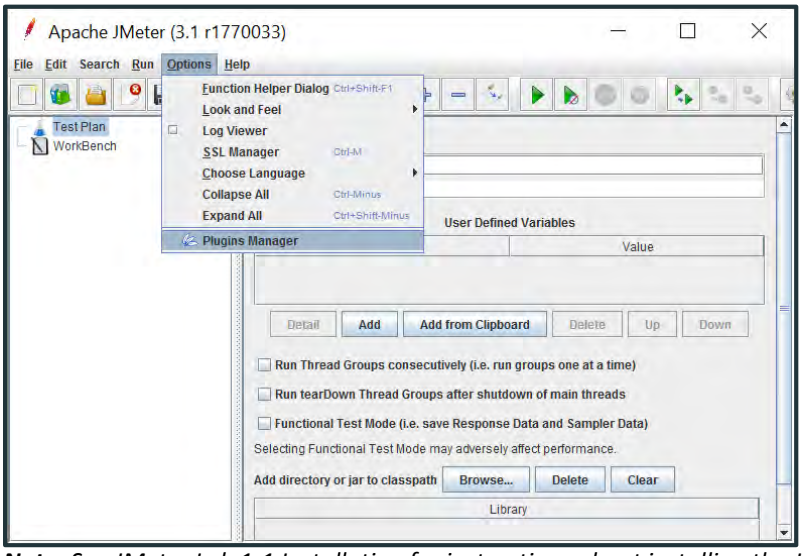
Complete these steps to create a new JMeter test:

1. Launch **JMeter** from your desktop shortcut.
2. Select **File ▶ Save Test Plan As**. The Test Plan.jmx window opens.
3. Enter **Lab1-4-Assertions.jmx** in the File Name field and click **Save**.
   Note: *Save the lab periodically throughout this exercise to preserve your work.*

Install the JSON Plugin

Complete these steps to install the JSON plugin in JMeter:

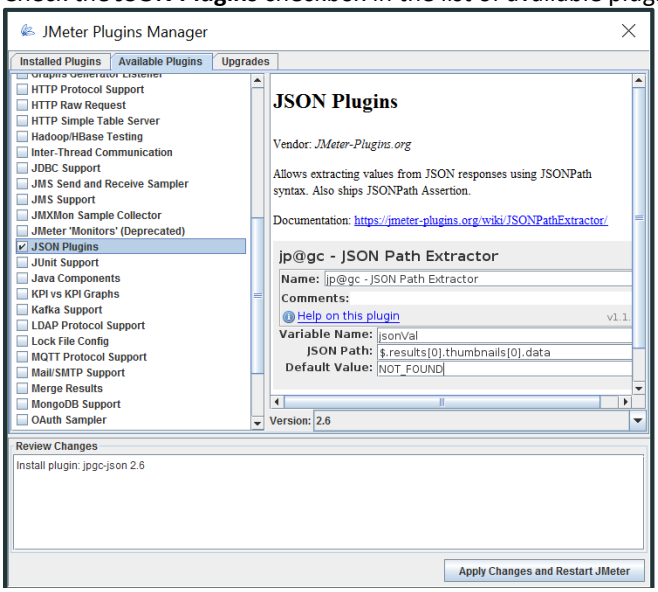
4. Select **Options** ▶ **Plugins Manager**.



Note: *See JMeter Lab 1-1 Installation for instructions about installing the JMeter Plugins Manager if it is not available in your Options menu.*

5. Click the **Available Plugins** tab of the JMeter Plugins Manager window.

   Check the **JSON Plugins** checkbox in the list of available plugins.



6. Click **Apply Changes and Restart JMeter**. JMeter closes and reopens automatically with the plugin that you selected installed.

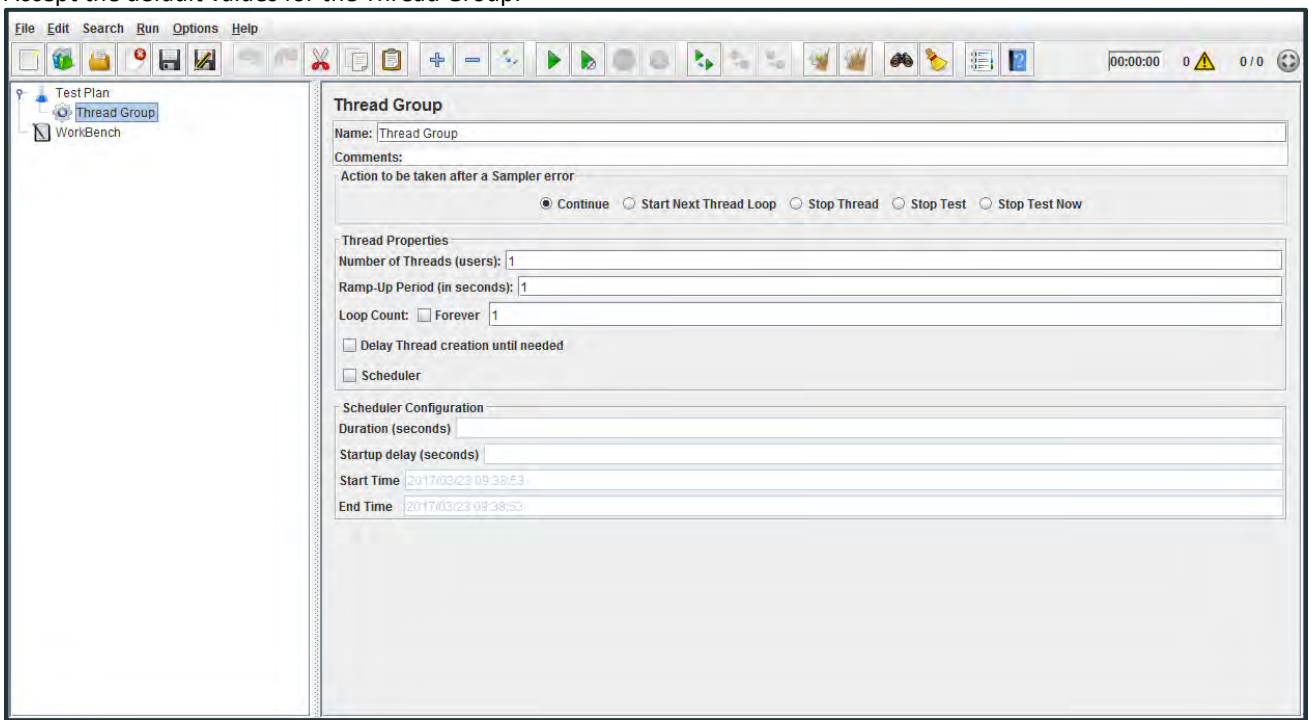
Add a Thread Group

Complete these steps to add a Thread Group:

7. Click **Test Plan** to select it.

8. Right-click **Test Plan** and select **Add ▶ Threads (Users) ▶ Thread Group.** The Thread Group configuration pane opens.

9. Accept the default values for the Thread Group.



Add an HTTP Request

Complete these steps to create an HTTP GET Request to ip-ranges.amazonaws.com/ip-ranges.json:

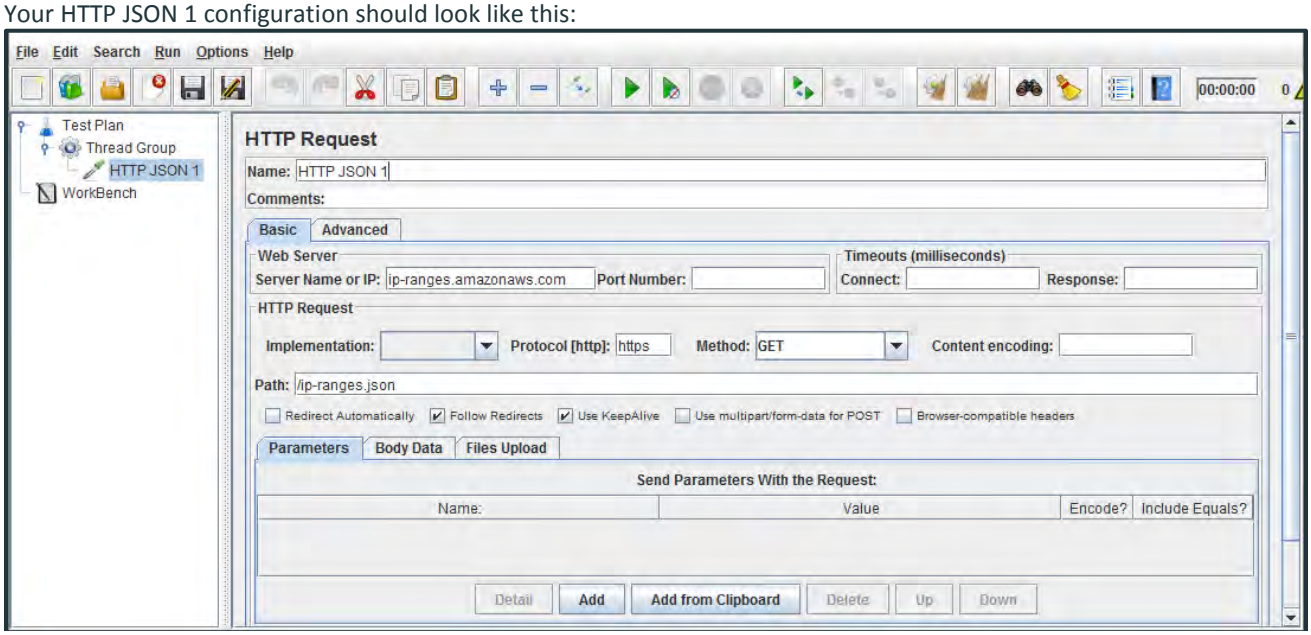
10. Right-click **Thread Group**.

11. Select **Add ▶ Sampler ▶ HTTP Request**. The HTTP Request configuration pane opens.

12. Configure the HTTP Request as follows:

Field	Value
Name	**HTTP JSON 1**
Server Name or IP	**ip-ranges.amazonaws.com**
Protocol	**https**
Path	**/ip-ranges.json**
Method	**GET**

Your HTTP JSON 1 configuration should look like this:



Add a JSON Path Assertion

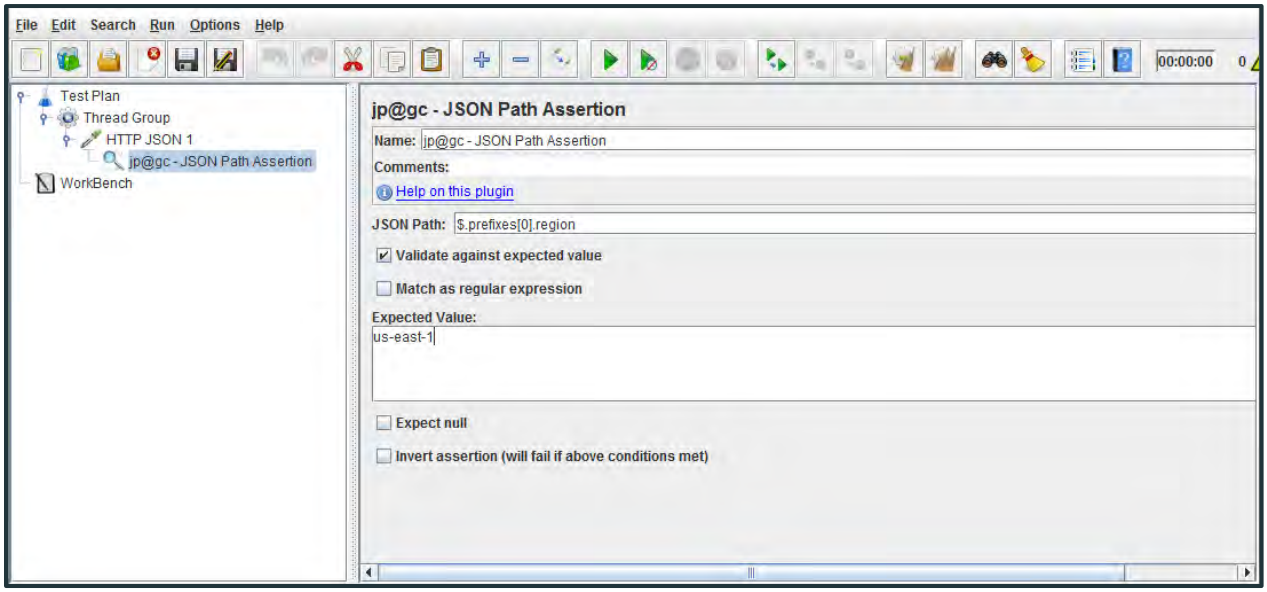
Complete these steps to add a JSON Path Assertion to HTTP JSON 1:

13. Right-click **HTTP JSON 1** and select **Add ▶ Assertions ▶ JSON Path Assertion**. The JSON Path Assertion configuration pane opens.

14. Configure JSON Path Assertion for HTTP JSON 1 as follows:

Field	Value
JSON Path	**$.prefixes[0].region**
Validate against expected value	**Checked**
Match as regular expression	**Unchecked**
Expected Value	**us-east-1**

Your JSON Path Assertion configuration should look like this:



Add a View Results Tree Listener

Complete these steps to add a View Results Tree listener:

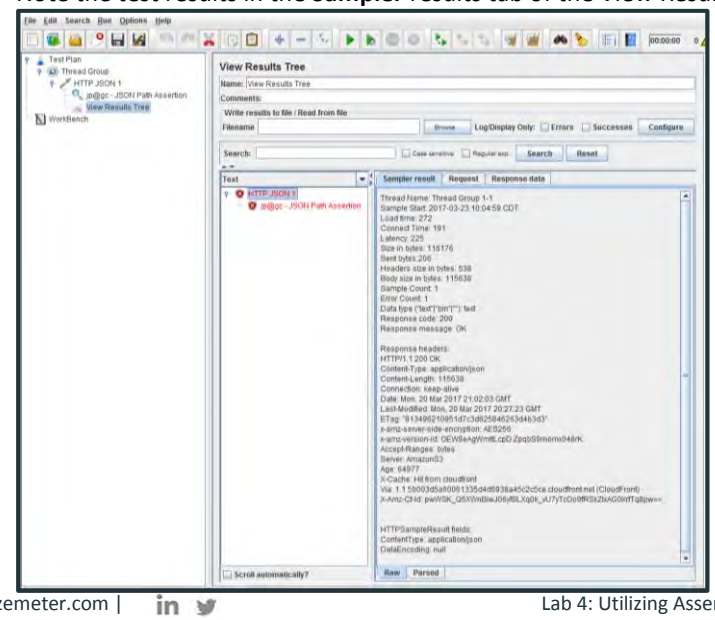
15. Click **HTTP JSON 1** to select it.

16. Right-click **HTTP JSON 1** and select **Add ▶ Listener ▶ View Results Tree**. The View Results Tree configuration pane opens.

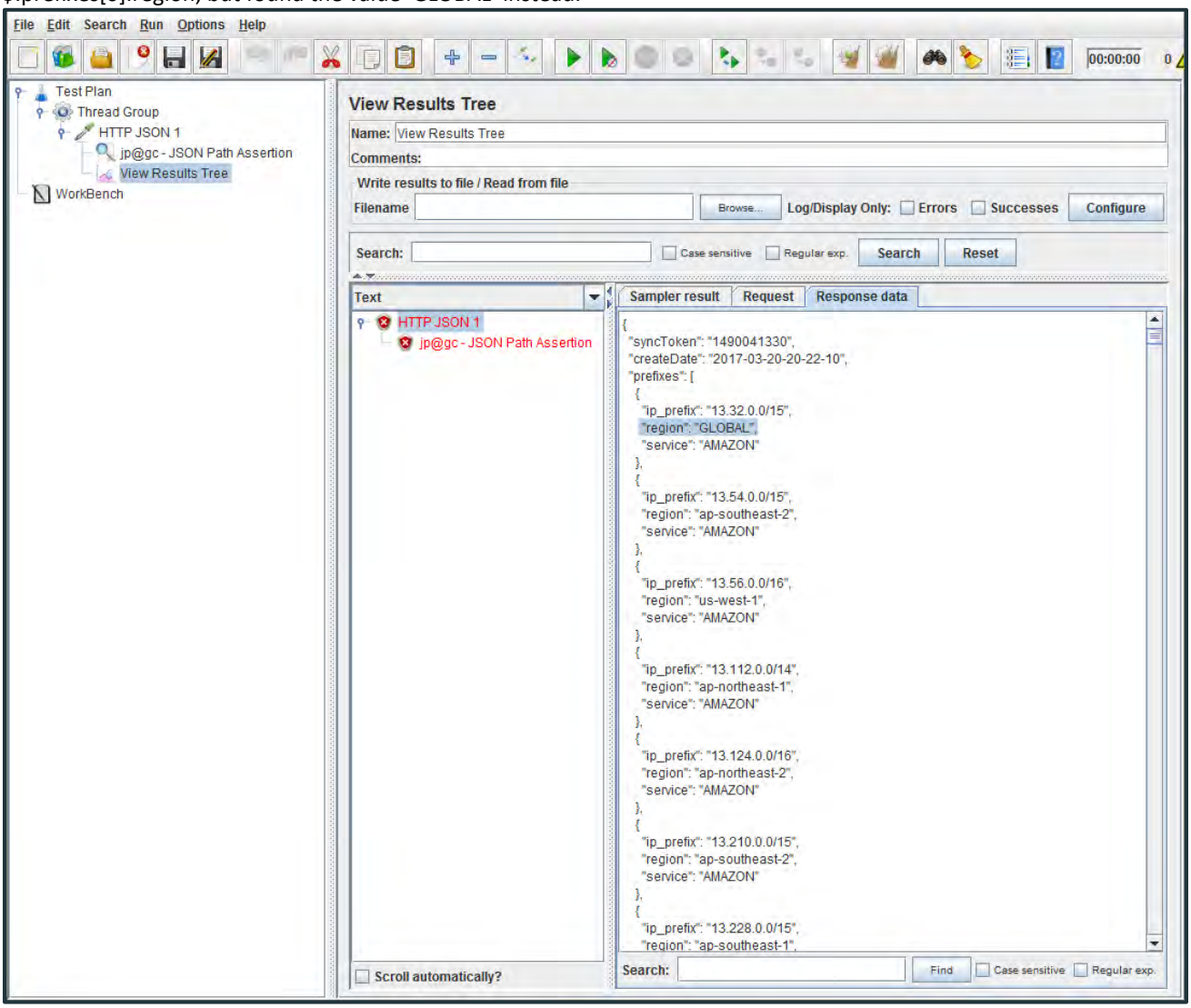
17. Click **Start** to run the test and view the results as they appear in the View Results Tree listener pane.

    Note: *Because the default thread group configuration was applied to this test, only one thread group will run one time.*

18. Note the test results in the **Sampler** results tab of the View Results Tree listener. Note that the test failed.



19. Click the **Response Data** tab. The test failed because the assertion expected to find 'us-east-1' in the $.prefixes[0].region, but found the value 'GLOBAL' instead.



Part 2: Verify a New Expected Value with a JSON Assertion

In Part 2 of this lab, we will recreate the test in Part 1, change the expected value, run the test, and note the new results.

Create an HTTP Request

Complete these steps to create an HTTP GET Request to ip-ranges.amazonaws.com/ip-ranges.json:

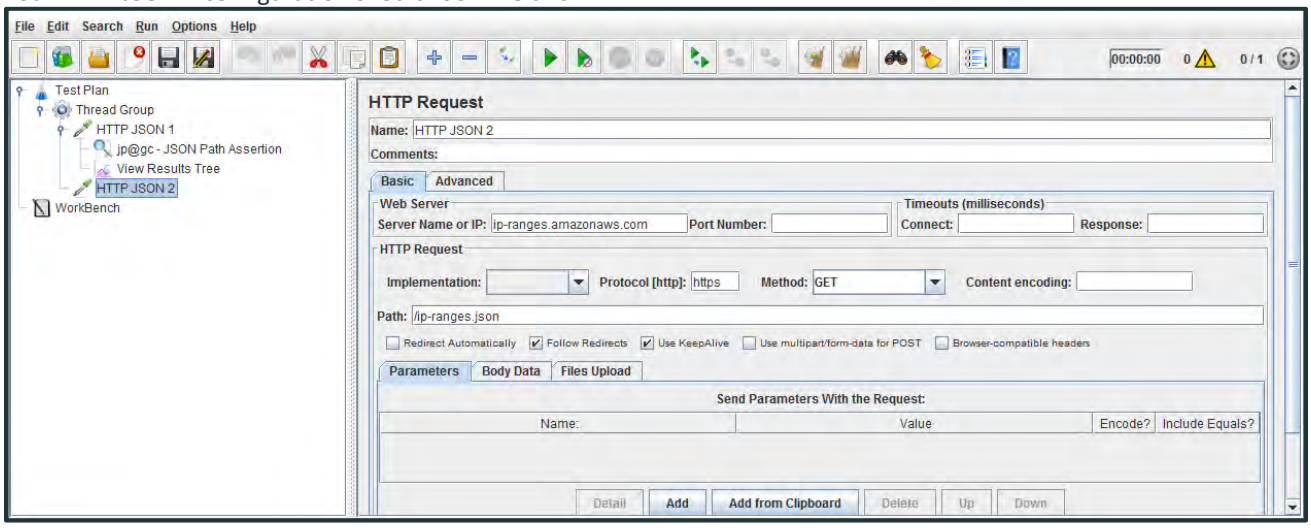
20. Click **Thread Group** to select it.

21. Right-click **Thread Group** and select **Add ▶ Sampler ▶ HTTP Request**. The HTTP Request configuration pane opens.

22. Configure the HTTP Request as follows:

Field	Value
Name	**HTTP JSON 2**
Server Name or IP	**ip-ranges.amazonaws.com**
Protocol	**https**
Path	**/ip-ranges.json**
Method	**GET**

Your HTTP JSON 2 configuration should look like this:



Add a JSON Path Assertion

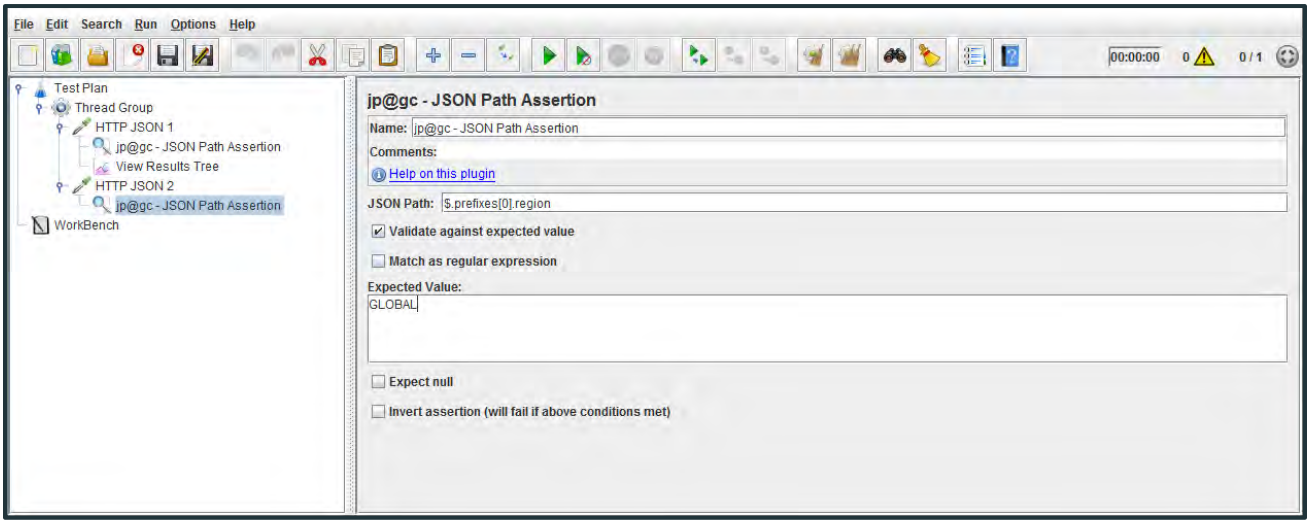
Complete these steps to add a JSON Path Assertion:

23. Right-click **HTTP JSON 2** and select **Add ▶ Assertions ▶ JSON Path Assertion**. The JSON Path Assertion configuration pane opens.

24. Configure JSON Path Assertion for HTTP JSON 2 as follows:

Field	Value
JSON Path	**$.prefixes[0].region**
Validate against expected value	**Checked**
Match as regular expression	**Unchecked**
Expected Value	**GLOBAL**

Your JSON Path Assertion configuration for HTTP JSON 2 should this:



Add a View Results Tree Listener

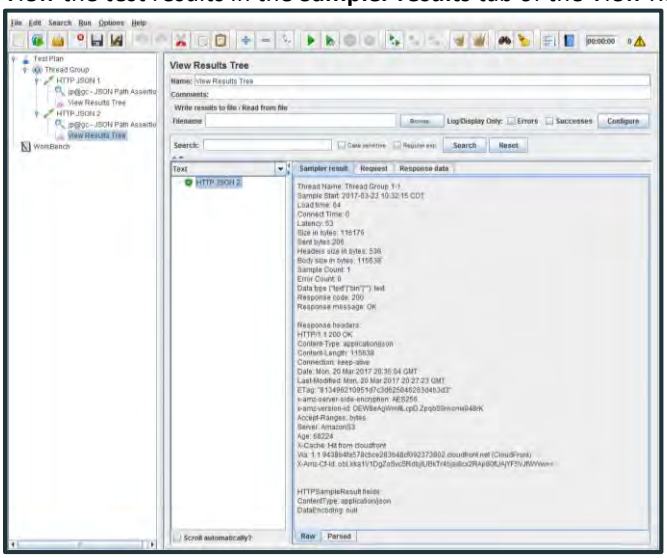
Complete these steps to add a View Results Tree listener:

25. Click **HTTP JSON 2** to select it.

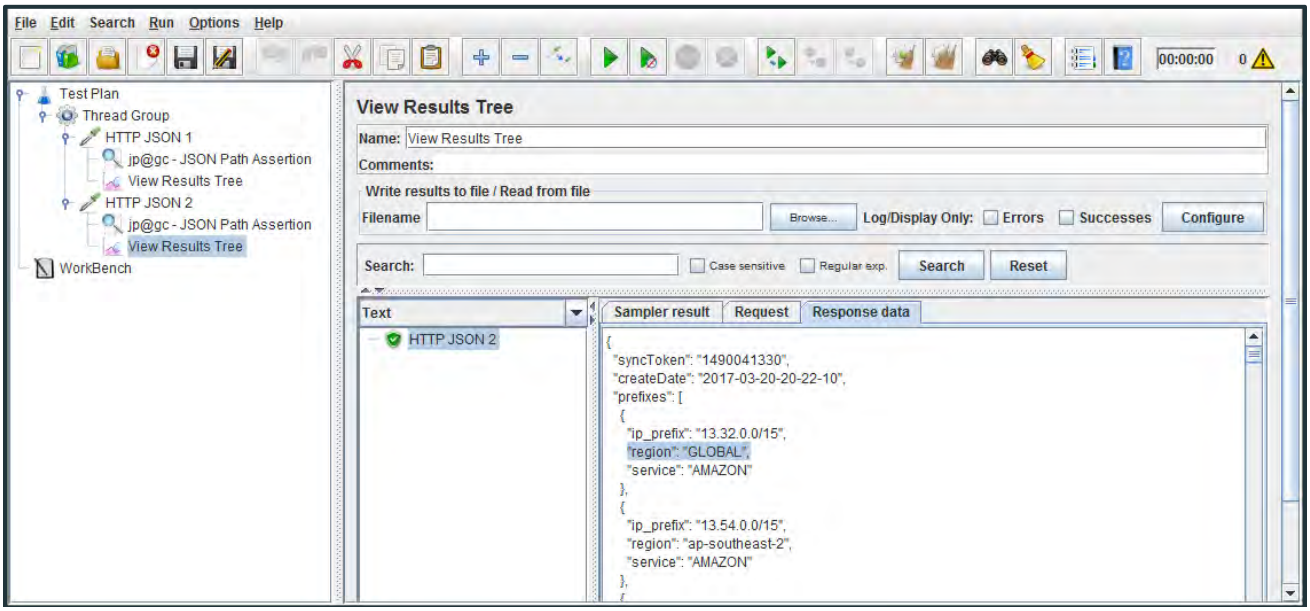
26. Right-click **HTTP JSON 2** and select **Add ▶ Listener ▶ View Results Tree**. The View Results Tree configuration pane opens.

27. Click **Start** to run the test again.

28. View the test results in the **Sampler results** tab of the View Results Tree listener. Note that the test passed.



Click the Response data tab to view the Response data and note that the region in the first prefix matches the expected value field of GLOBAL.



Part 3: Apply a Response Assertion to Verify Text in a Response

In Part 3 of this lab, we will apply a Response Assertion to verify whether specific text is contained in the response or not.

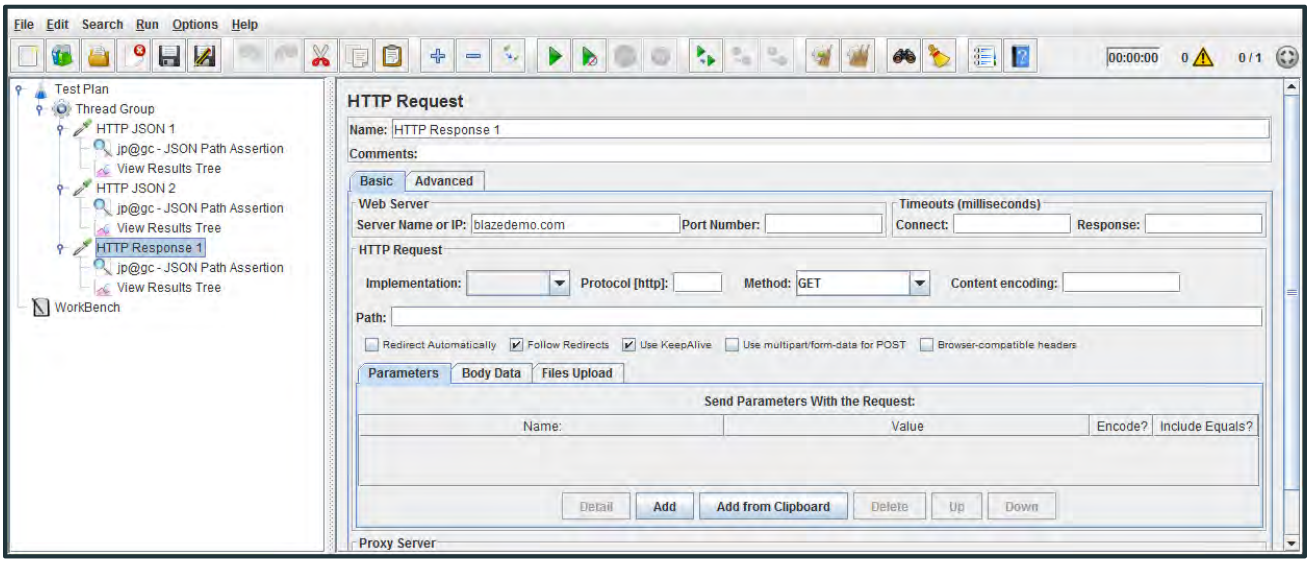
Create a Duplicate and Configure an HTTP Get Request

Complete these steps to create a duplicate HTTP GET request to the BlazeMeter website:

29. Right-click **HTTP JSON 2** and select **Duplicate** to create a duplicate of this HTTP Request.

30. Configure the HTTP Request as follows:

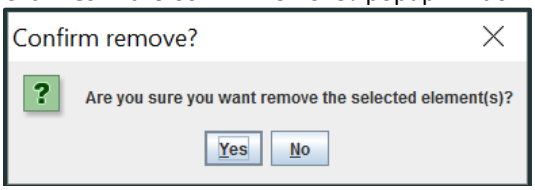
Field	Value
Name	**HTTP Response 1**
Server Name or IP	**blazedemo.com**
Method	**GET**



31. While pressing, and holding the **Shift** key, click **JSON Path Assertion** and **View Results Tree** under HTTP Response 1 to highlight both items.

32. Right-click and select **Remove**.

33. Click **Yes** in the Confirm remove? popup window.



Add a Response Assertion

Complete these steps to add a Response Assertion to HTTP Response 1:

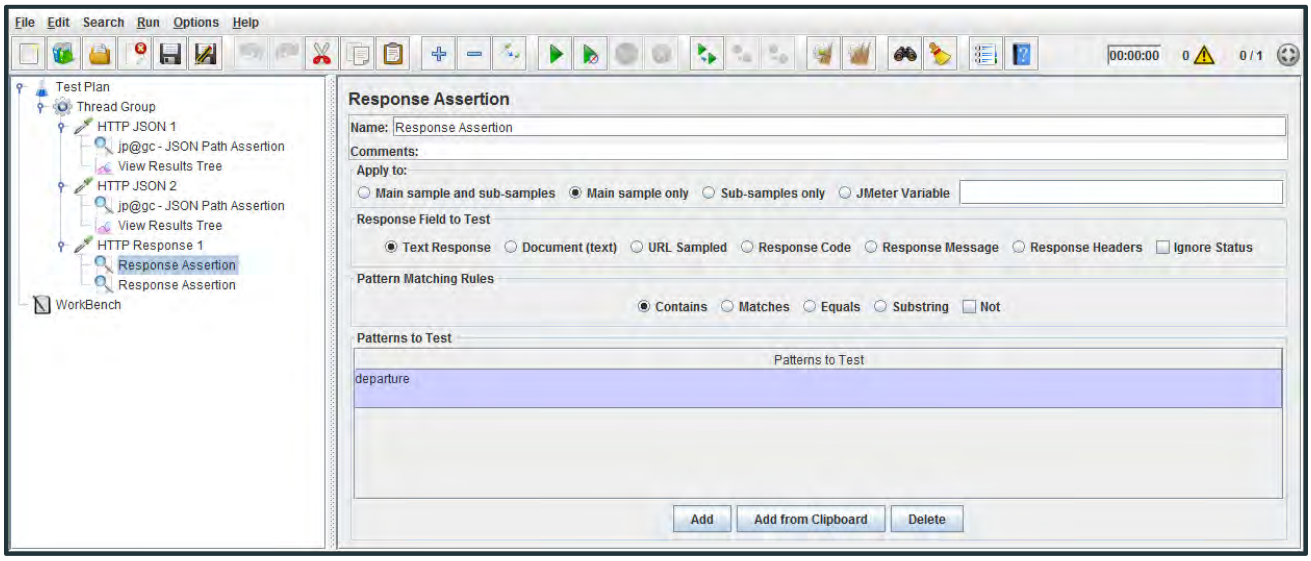
34. Right-click **HTTP Response 1** and select **Add ▶ Assertions ▶ Response Assertion**.

35. Configure this Response Assertion as follows:

Field	Value
Apply to:	**Main sample only**
Response Field to Test	**Text Response**
Pattern Matching Rules	**Contains**

36. Click **Add** to add a row to the Patterns to Test section of the Response Assertion pane.

37. Enter **departure** in the row and accept all other default settings.



Add a Second Response Assertion

Complete these steps to add a second Response Assertion to HTTP Response 1:

38. Click **HTTP Response 1** to select it.

39. Right-click **HTTP Response 1** and select **Add ▶ Assertions ▶ Response Assertion**.

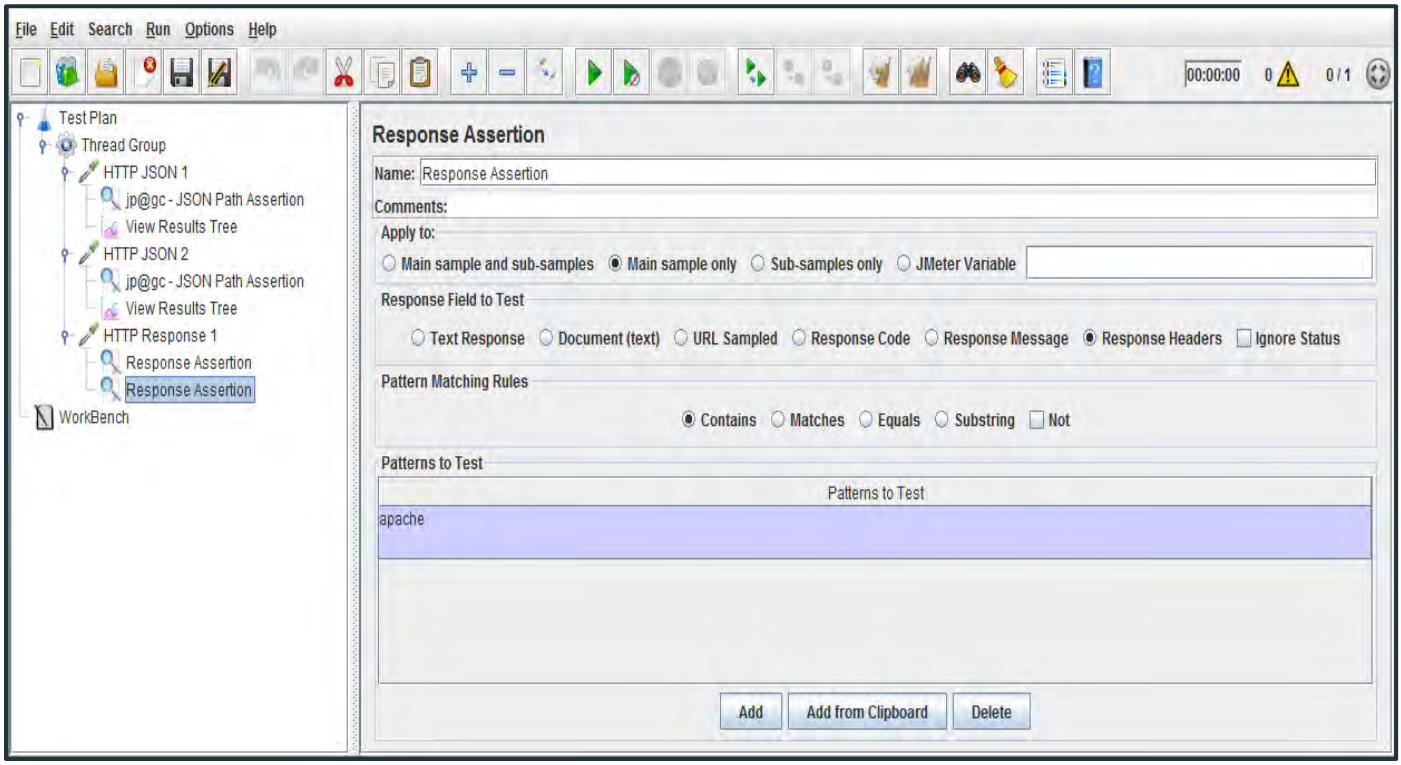
40. Configure this new Response Assertion as follows:

Field	Value
Apply to:	**Main sample only**
Response Field to Test	**Response Headers**
Pattern Matching Rules	**Contains**

41. Click **Add** to add a row to the Patterns to Test section of the Response Assertion pane.

42. Enter **apache** in the row and accept all other default settings.

Your second Response Assertion configuration for HTTP Response 1 should look like this:



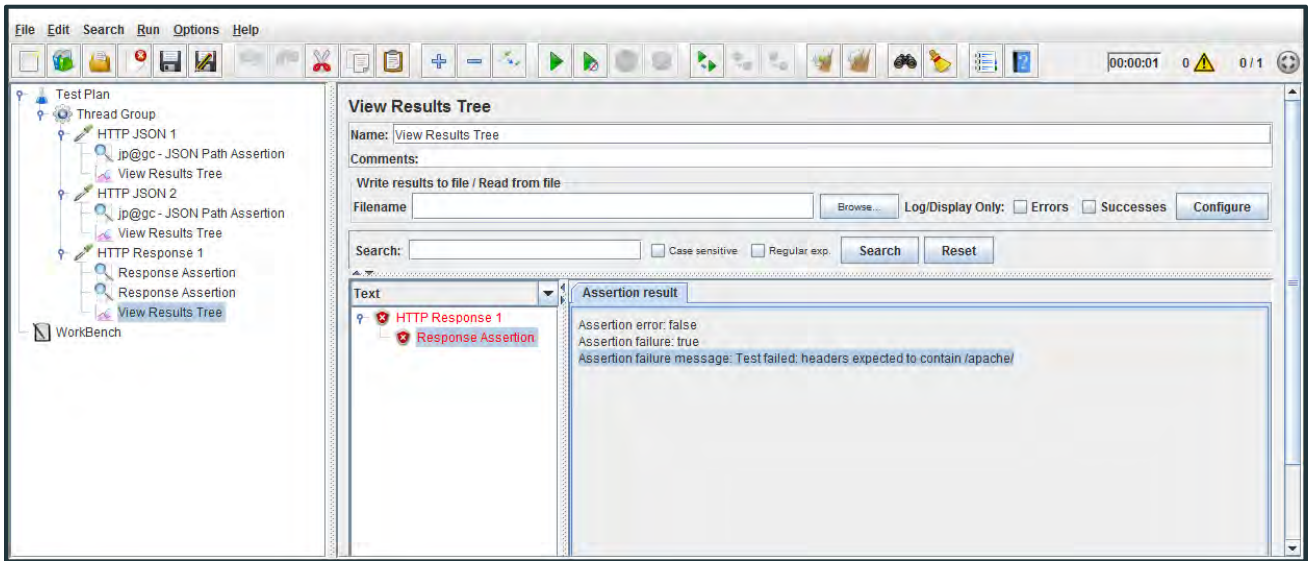
Add a View Results Tree Listener

Complete these steps to add a View Results Tree listener:

43. Click **HTTP Response 1** to select it.

44. Right-click **HTTP Response 1** and select **Add ▶ Listener ▶ View Results Tree**. The View Results Tree configuration pane opens.

45. Click **Start** to run the test and view the results as they appear in the View Results Tree listener pane. Note that the test fails.



Part 4: Utilize a Response Assertion to Verify Receipt of a Code

In Part 4 of this lab, we will utilize a Response Assertion to verify receipt of a 404 code.

Create a Duplicate HTTP Get Request

Complete these steps to create a duplicate HTTP GET Request to the BlazeMeter website:

46. Click **HTTP Response 1** to select it.

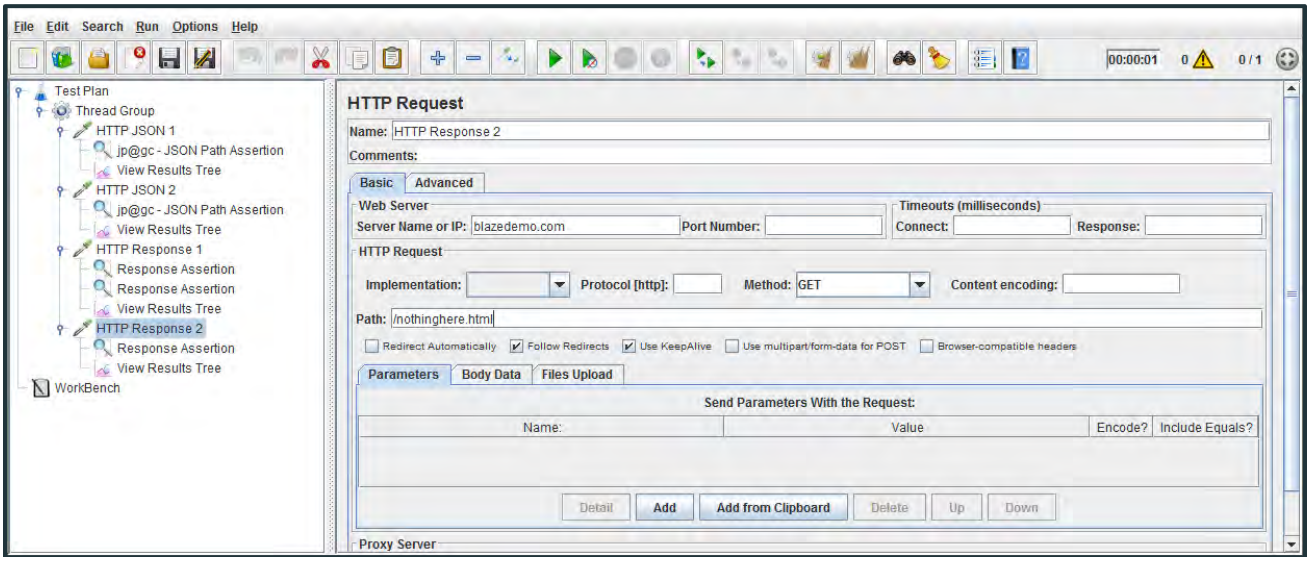
47. Right-click **HTTP Response 1** and select **Duplicate** to create a duplicate of this HTTP Request.

48. Remove one of the Response Assertions for your new HTTP Request.

49. Configure the new HTTP Request as follows:

Field	Value
Name	**HTTP Response 2**
Server Name or IP	**blazedemo.com**
Method	**GET**
Path	**/nothinghere.html**

Your HTTP Response 2 configuration should look like this:



Configure the Response Assertion for HTTP Response 2

Complete these steps to configure the Response Assertion for HTTP Response 2:

50. Click the **Response Assertion** for HTTP Response 2 to select it.

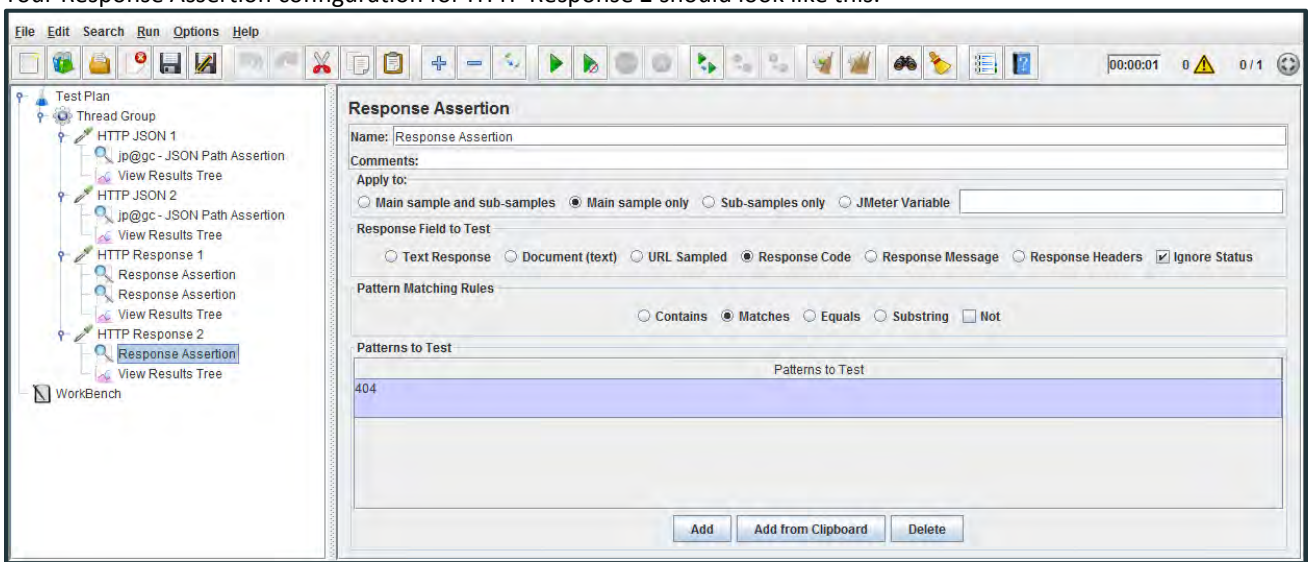
51. Configure this Response Assertion as follows:

Field	Value
Apply to:	**Main sample only**
Response Field to Test	**Response Code**
Response Field to Test	**Ignore Status Selected**
Pattern Matching Rules	**Matches**

52. Click **Add** to add a row to the Patterns to Test section of the Response Assertion pane.

53. Enter **404** in the row and accept all other default settings.

Your Response Assertion configuration for HTTP Response 2 should look like this:

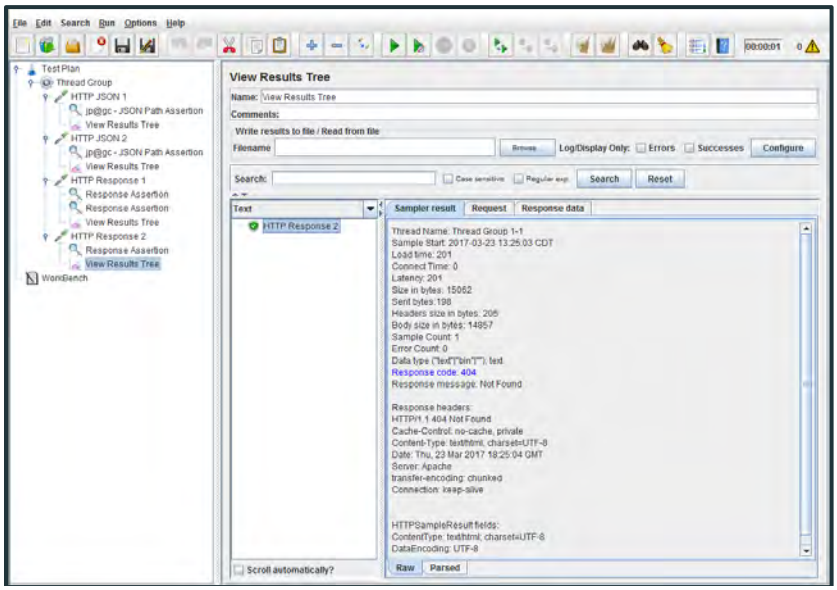


Execute Test and Note Results

Complete these steps to execute your new test and view the results:

54. Click **View Results Tree** for HTTP Response 2 to select it.

55. Click **Start** to run your test. Note that the test passes and the Response Code of 404 is found.



Part 5: Verify an Outcome with a Duration Assertion

In Part 5 of this lab, we will use a Duration Assertion to verify that the outcome matched the expectation of the test.

Add a Dummy Sampler

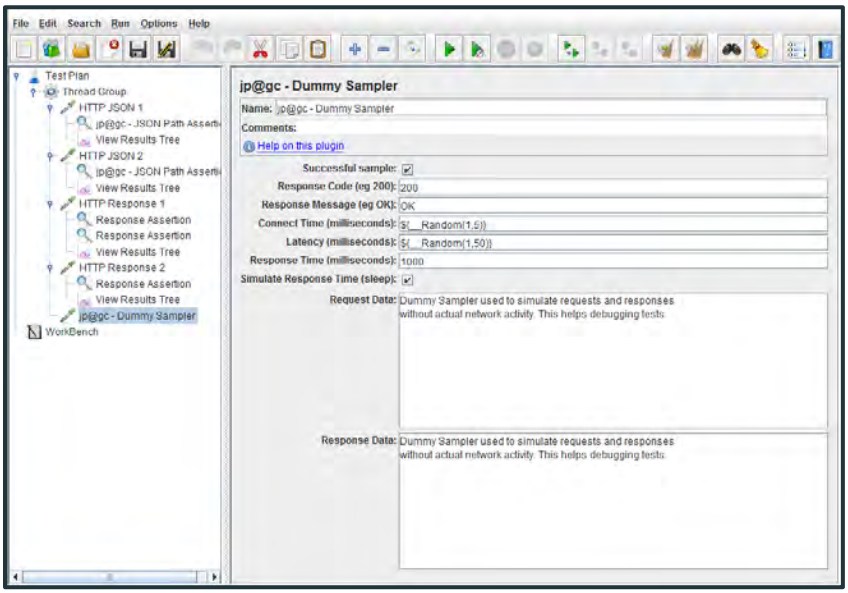
Complete these steps to add a Dummy Sampler to your test:

56. Click **Thread Group** to select it.

57. Right-click **Thread Group** and select **Add ▶ Sampler ▶ Dummy Sampler**.

58. Enter **1000** in the Response Time field and accept all other default settings.

    Your Dummy Sampler configuration should look like this:



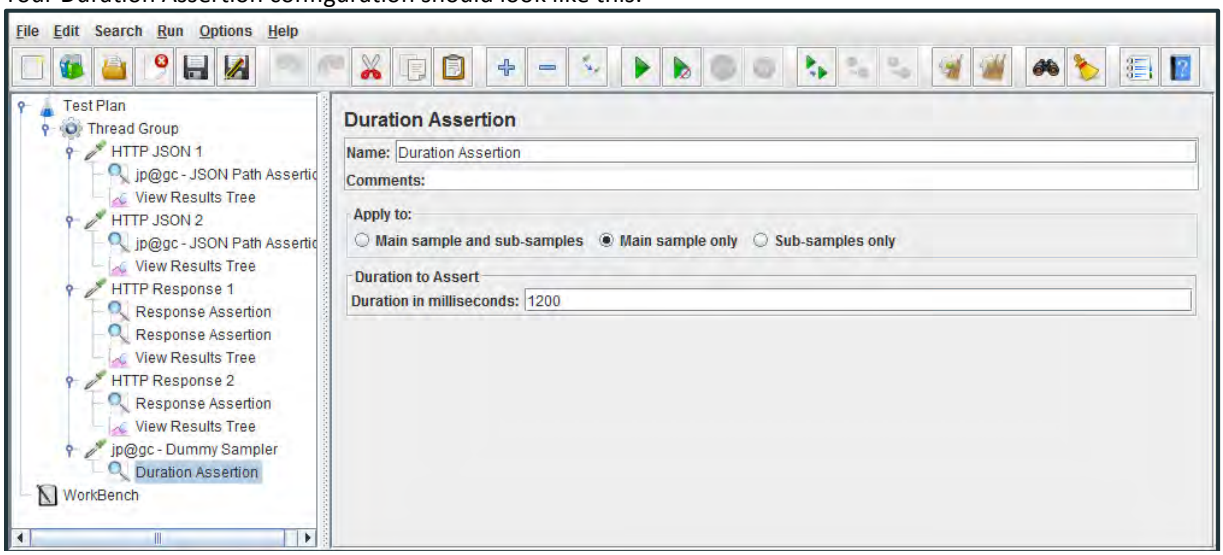
Add a Duration Assertion

Complete these steps to add and configure a Duration Assertion:

59. Right-click **Dummy Sampler** and select **Add ▶ Assertions ▶ Duration Assertion**.

60. Enter 1200 in the Duration in milliseconds field and accept all other default settings.

Your Duration Assertion configuration should look like this:



Add a View Results Tree Listener

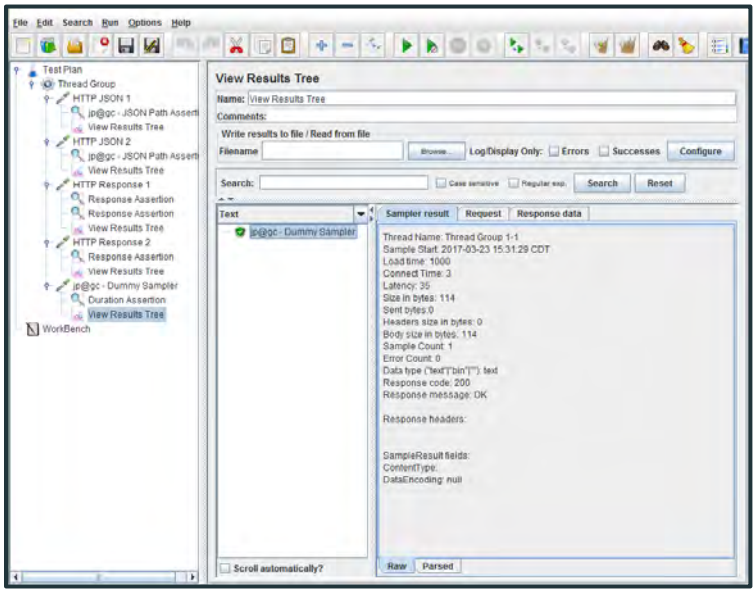
Complete these steps to add a View Results Tree listener:

61. Click **Dummy Sampler** to select it.

62. Right-click **Dummy Sampler** and select **Add ▶ Listener ▶ View Results Tree**. The View Results Tree configuration pane opens.

63. Click Start to run the test and view the results as they appear in the View Results Tree listener pane

    .Your results should look like this:



Lab 5 - Understanding Correlations

Goals During this lab, you will learn how to extract a CSRF token from a response header, the body of a response, and so on. You will then learn how to use the extracted token in another HTTP request.

Scenario This is a more advanced lab that builds on everything that you learned in the previous labs. In this lab, you will use a regular expression extractor to extract a token while using timers, login, logout, and a data file with random delays between requests to mimic real world scenarios.

You will accomplish this by completing these tasks:

- Create a new JMeter test
- Add a Thread Group
- Add and configure an HTTP Request
- Extract the CSRF token
- Register a new account
- Add an HTTP Cookie Manager
- Add a View Results Tree listener
- Add an HTTP Request Sampler to login
- Add an HTTP Request Sampler to logout
- Add a Uniform Random Timer
- Add a Debug Sampler
- Review tips and tricks

Time 20 minutes

Create a New JMeter Test

Complete these steps to create a new JMeter test:

1. Launch **JMeter** from your desktop shortcut.
2. Select **File ▶ Save Test Plan As**. The Test Plan.jmx window opens.
3. Enter **Lab1-5-Correlations.jmx** in the File Name field and click **Save**.
   Note: *Save the lab periodically throughout this exercise to preserve your work.*

Add a Thread Group

Complete these steps to add and configure a Thread Group:

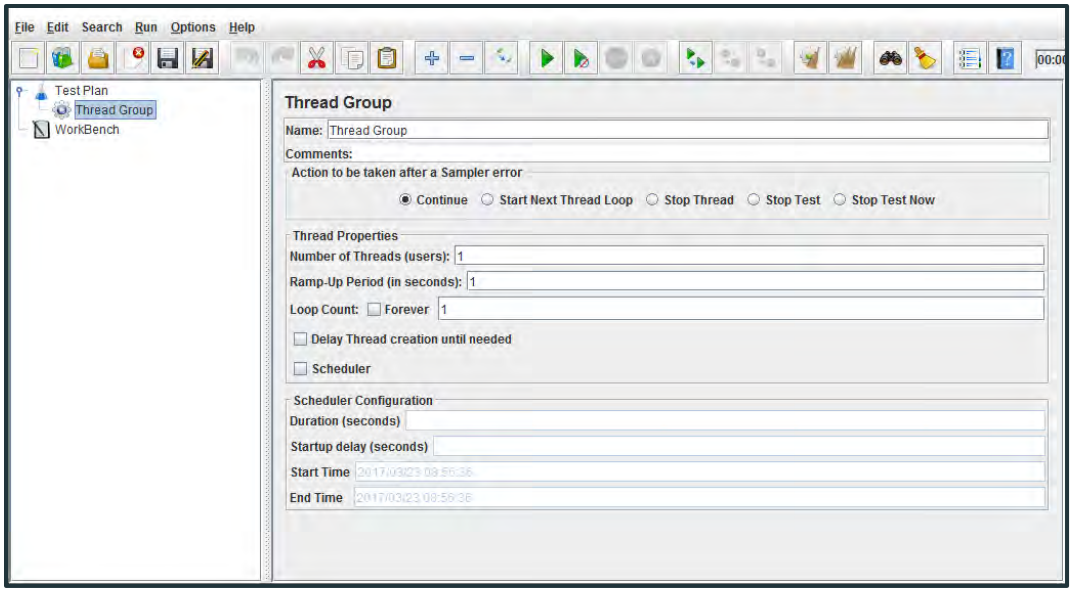
4. Click **Test Plan** to select it.

5. Right-click **Test Plan** and select **Add ▶ Thread Groups ▶ Thread Group**. The Thread Group configuration pane opens.

6. Configure your thread group as follows by accepting the default values:

Field	Value
Number of Threads	**1**
Ramp-Up Period	**1**
Loop Count	**1**

Your Thread Group configuration should look like this:



Add and Configure an HTTP Request

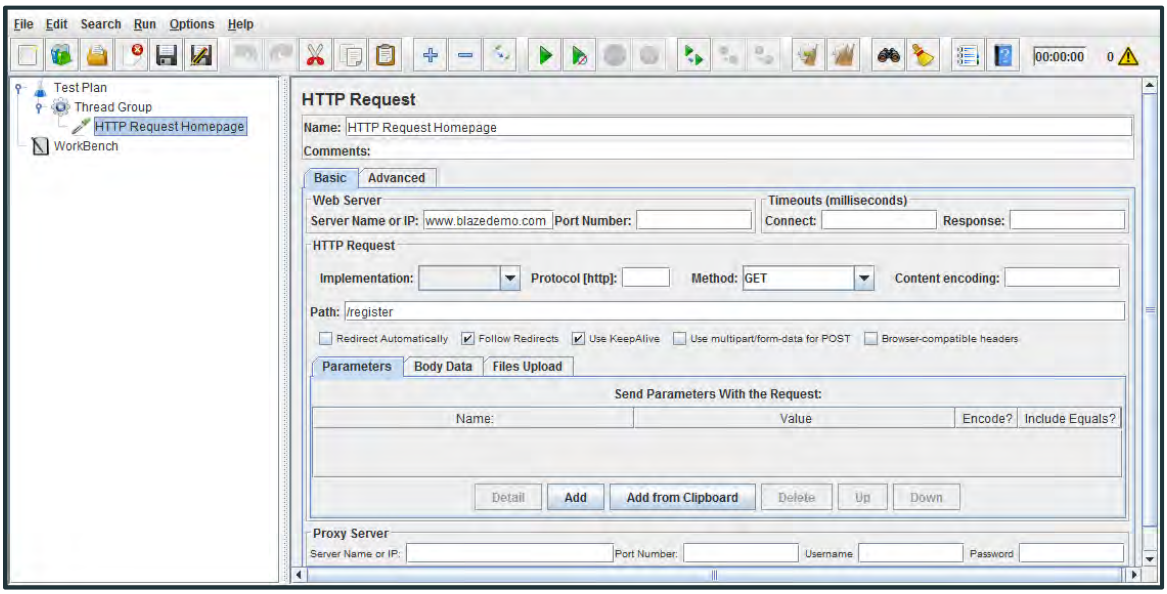
Complete these steps to add and configure an HTTP Request that will be used to extract the CSRF token:

7. Right-click **Thread Group** and select **Add ▶ Sampler ▶ HTTP Request**. The HTTP Request configuration pane opens.

8. Configure the HTTP Request as follows:

Field	Value
Name	**HTTP Request Homepage**
Server or IP	**www.blazedemo.com**
Path	**/register**
Method	**GET**

Your HTTP Request Homepage configuration should look like this:



Extract the CSRF Token

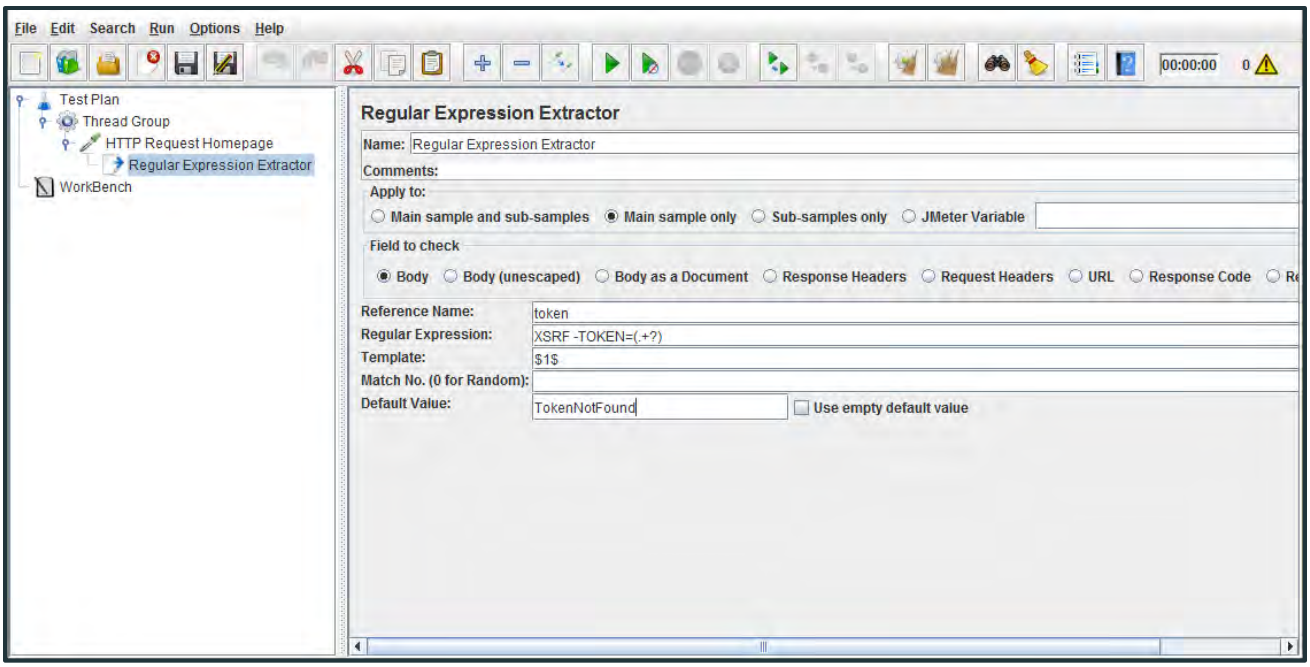
Complete these steps to extract the CSRF token from the GET request response received from the www.blazemeter.com/register web page:

9. Right-click **HTTP Request Homepage** and select **Add ▶ Post Processors ▶ Regular Expression Extractor**. The Regular Expression Extractor configuration pane opens.

10. Configure the Regular Expression Extractor as follows:

Field	Value	Notes
Main sample only	**Selected**	
Response Headers	**Selected**	
Reference Name	**token**	This is the name that will be used later in the script for extracting the CSRF token.
Regular Expression	**XSRF -TOKEN=(.+?)**	This is the expression that will be used to extract the CSRF token.
Template	**1**	
Default Value	**TokenNotFound**	This message will appear in your test results if no token was found.

Your Regular Expression Extractor configuration should look like this:



Register a New Account

Complete these steps to register a new account and create a new HTTP Request:

11. Click **Thread Group** to select it.

12. Right-click **Thread Group** and select **Add ▶ Sampler ▶ HTTP Request**. The HTTP Request configuration pane opens.

13. Configure your HTTP Request sampler as follows:

Field	Value
Name	**HTTP Request Register**
Method	**Post**
Server Name or IP	**www.blazedemo.com**
Path	**/register**

    a. Click **Add** five times at the bottom of the Send Parameters With Request section to add five rows to this section.

    b. Configure row one as follows:

Field	Value
Name	**name**
Value	**Jane Smith**

    c. Configure row two as follows:

Field	Value
Name	**company**
Value	**BlazeMeter**

d. Configure row three as follows:

Field	Value
Name	**email**
Value	**jsmith@blazemeter.com**

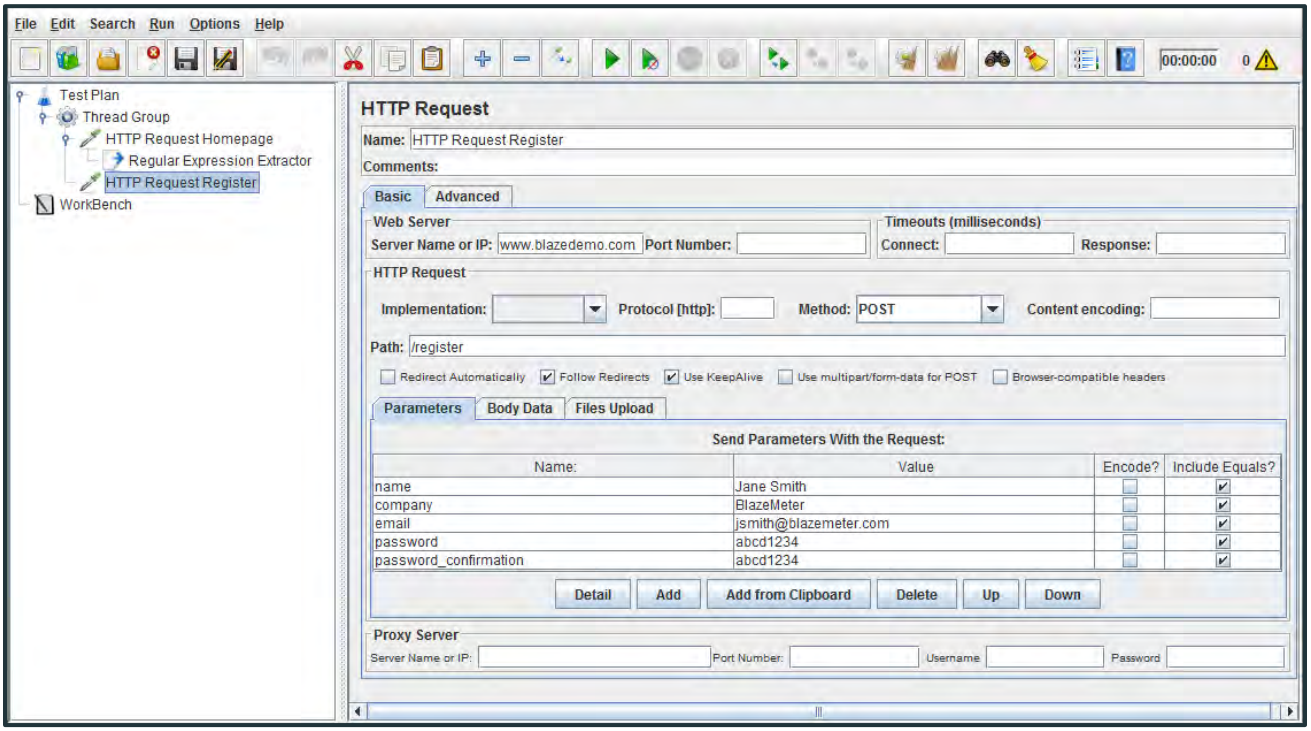
e. Configure row four as follows:

Field	Value
Name	**password**
Value	**abcd1234**

f. Configure row five as follows:

Field	Value
Name	**password_confirmation**
Value	**abcd1234**

Your HTTP Request sampler configuration should look like this:



Add an HTTP Cookie Manager

Complete these steps to add and configure an HTTP Cookie Manager:

14. Right-click **HTTP Request Register** and select **Add ▶ Config Element ▶ HTTP Cookie Manager**. The HTTP Cookie Manager configuration pane opens.

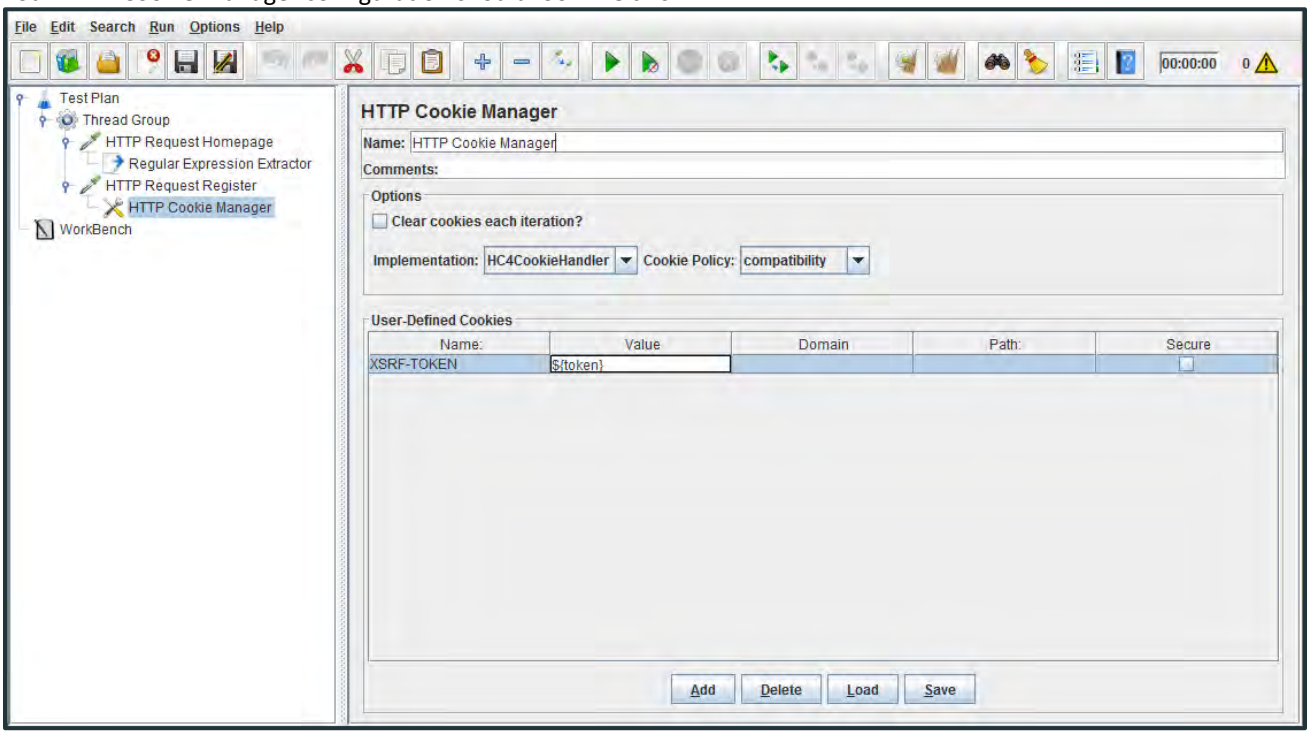
15. Select **compatibility** from the Cookie Policy dropdown menu.

16. Click **Add** to add a row to the User-Defined Cookies section of the pane.

17. Configure the HTTP Cookie Manager as follows:

Field	Value	Notes
Name	**XSRF-TOKEN**	
Value	**${token}**	This allows the HTTP Cookie Manager to use the token that was extracted in the first sequence.

Your HTTP Cookie Manager configuration should look like this:



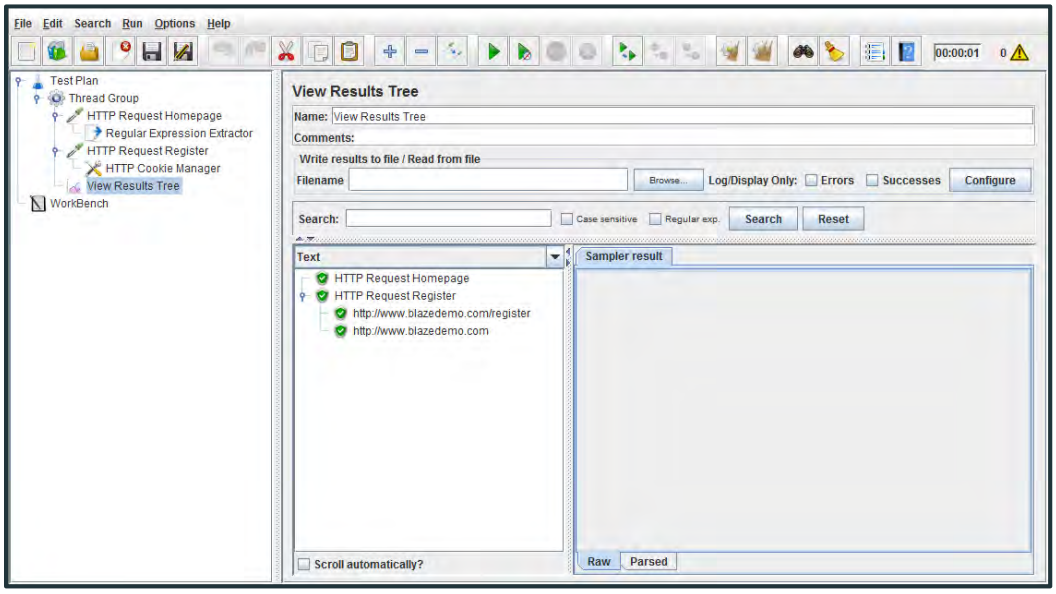
Add a View Results Tree Listener

Complete these steps to add a View Results Tree listener that will capture and display the results of your test:

18. Click **Thread Group** to select it.

19. Right-click **Thread Group** and select **Add ▶ Listener ▶ View Results Tree**. The View Results Tree configuration pane opens.

Click **Start** to run the script. The test results appear in the Sampler result tab of the View Results Tree pane. The test successfully extracts the token from the first request and posts a cookie to the second request.



Add an HTTP Request Sampler to Login

Complete these steps to add an HTTP Request sampler to login to the website:

20. Click **Thread Group** to select it.

21. Right-click **Thread Group** and select **Add ▶ Sampler ▶ HTTP Request**. The HTTP Request configuration pane opens.

22. Configure the HTTP Request Login as follows:

Field	Value
Name	**HTTP Request Login**
Server Name or IP	**www.blazedemo.com**
Path	**/login**
Method	**POST**

a. Click **Add** two times to add two rows to the Send Parameters With Request section of the pane.

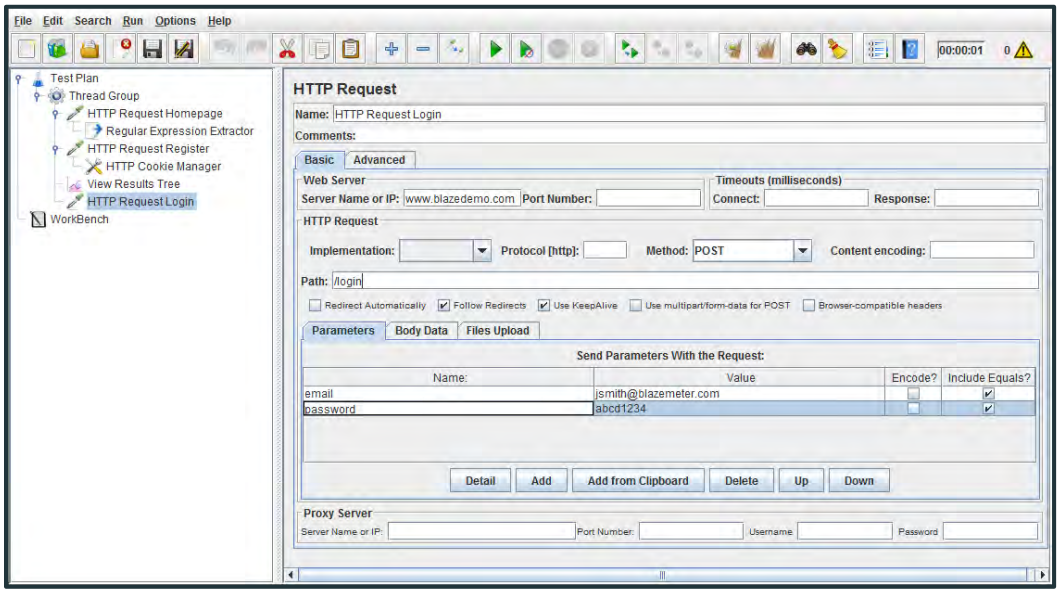
b. Configure row one as follows:

Field	Value
Name	**email**
Value	**jsmith@blazemeter.com**

c. Configure row two as follows:

Field	Value
Name	**password**
Value	**abcd1234**

Your HTTP Request Login configuration should look like this:



Add an HTTP Request Sampler to Logout

Complete these steps to add an HTTP Request Sampler to logout of the website and pass the token as a parameter:

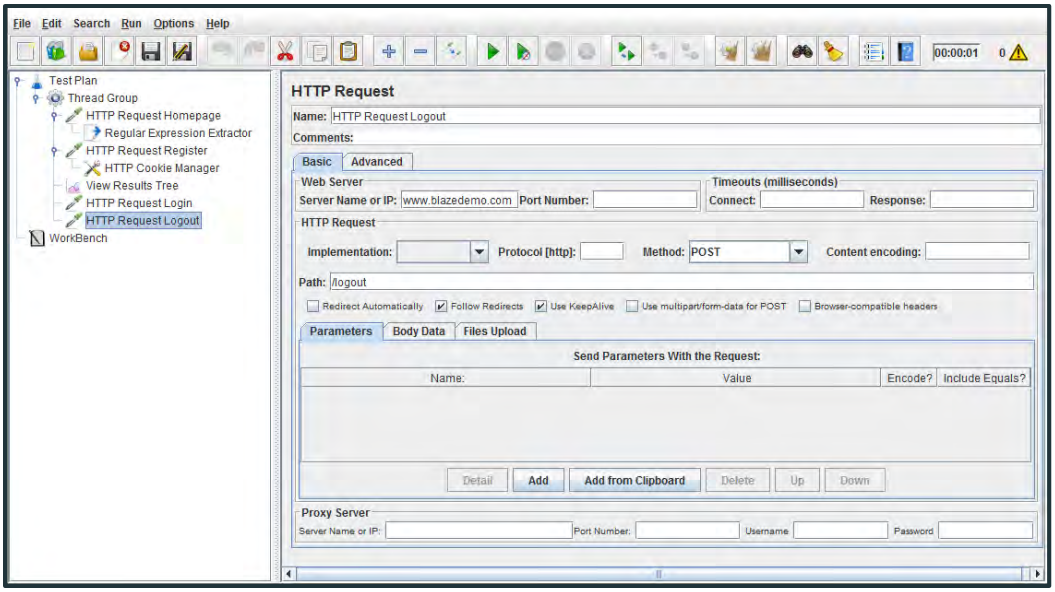
23. Click **Thread Group** to select it.

24. Right-click **Thread Group** and select **Add ▶ Sampler ▶ HTTP Request**. The HTTP Request configuration pane opens.

25. Configure the HTTP Request Logout as follows:

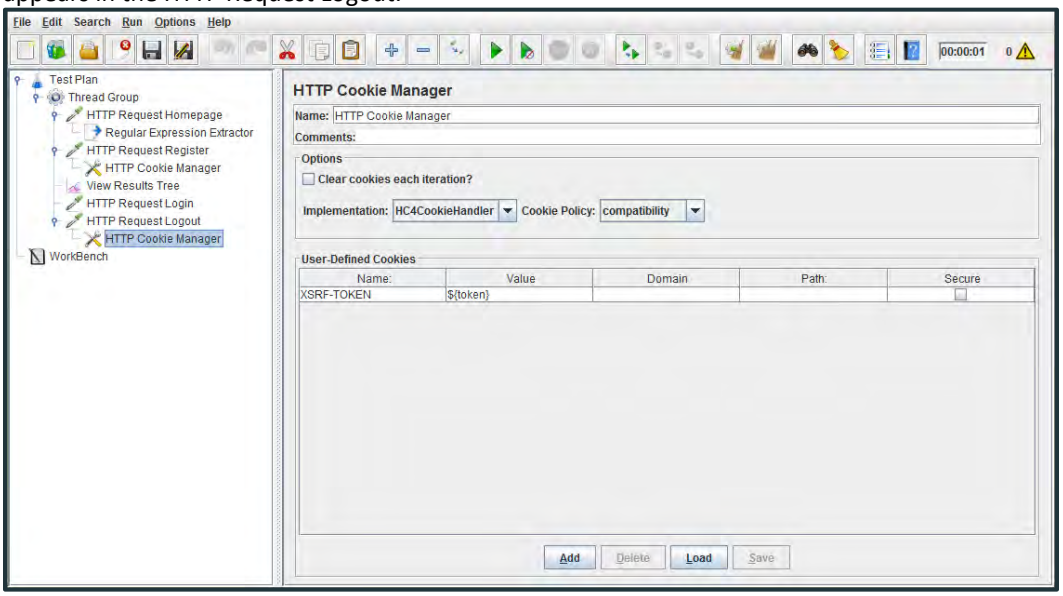
Field	Value
Name	**HTTP Request Logout**
Server Name or IP	**www.blazedemo.com**
Method	**POST**
Path	**/logout**
Follow Redirects	**Checked**
Use KeepAlive	**Checked**

Your HTTP Request Logout configuration should look like this:



26. Right-click the **HTTP Cookie Manager** in the Test Plan pane on the left and select **Copy**.

27. Right-click **HTTP Request Logout** in the Test Plan pane and select **Paste**. A copy of the HTTP Cookie Manager appears in the HTTP Request Logout.



Add a Uniform Random Timer

Complete these steps to add a Uniform Random Timer that gives a random delay of approximately 2 seconds between requests:

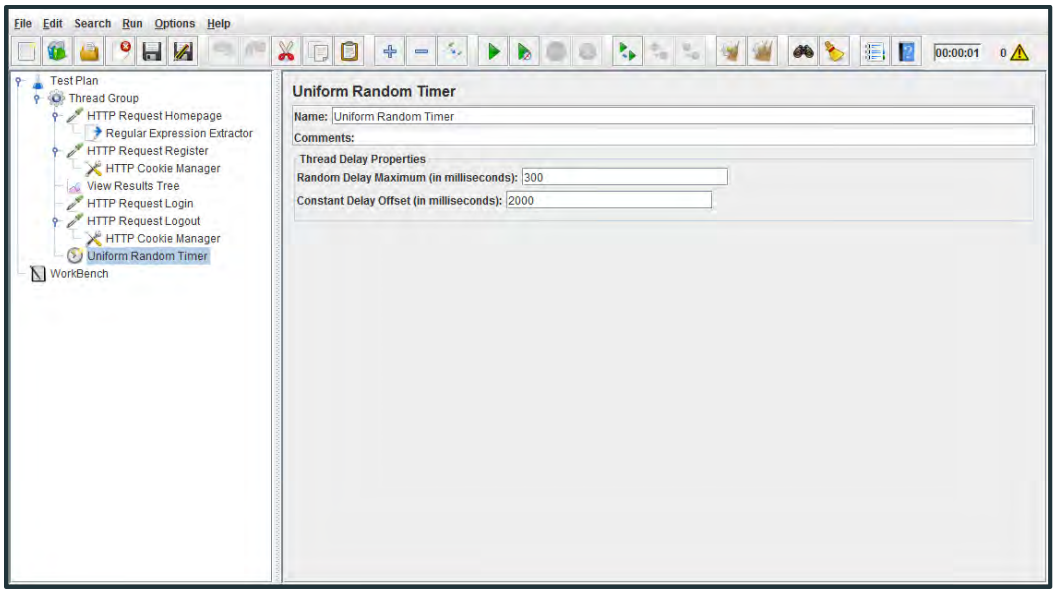
28. Click **Thread Group** to select it.

29. Right-click **Thread Group** and select **Add ▶ Timer ▶ Uniform Random Timer**. The Uniform Random Timer configuration pane opens.

30. Configure the Uniform Random Timer as follows:

Field	Value
Random Delay Maximum	**300**
Constant Delay Offset	**2000**

Your Uniform Random Timer configuration should look like this:

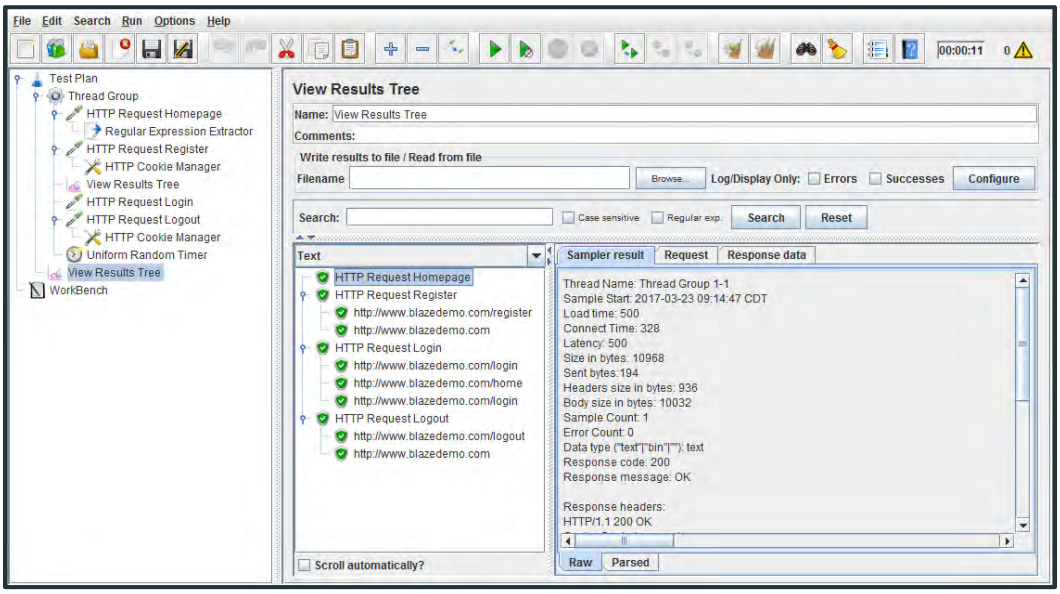


31. Click **Test Plan** to select it.

32. Right-click **Test Plan** and select **Add ▶ Listener ▶ View Results Tree**. The View Results Tree configuration pane opens.

33. Click **Start** to run your test.

Your test results should look like this:



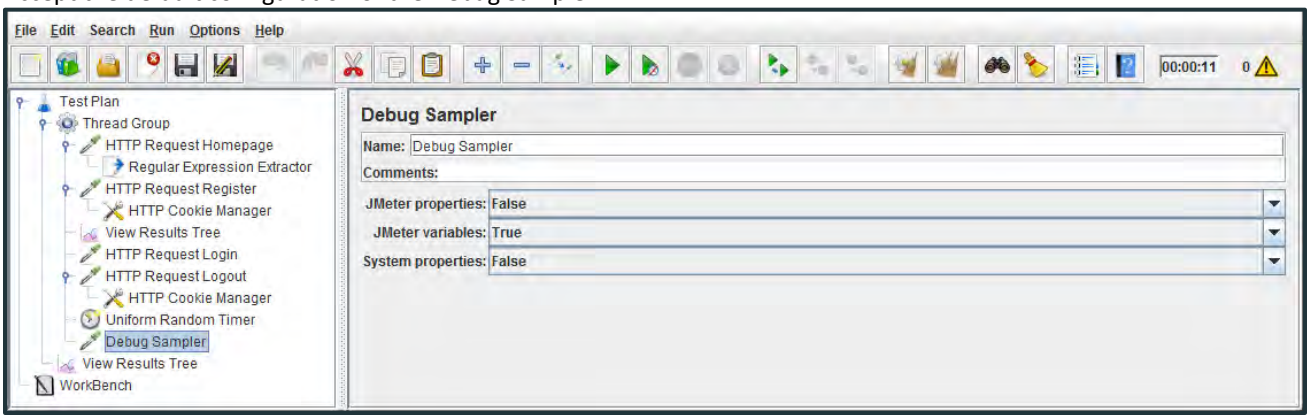
Add a Debug Sampler

You must add a Debug Sampler to view what was extracted during your test. Complete these steps to add a Debug Sampler:

34. Click **Thread Group** to select it.

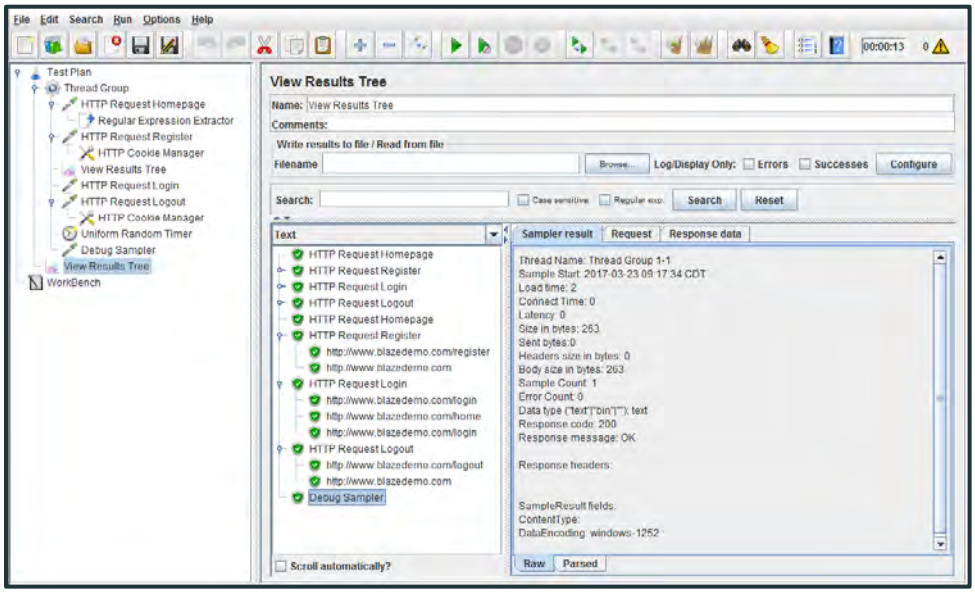
35. Right-click **Thread Group** and select **Add ▶ Sampler ▶ Debug Sampler**. The Debug Sampler configuration pane opens.

36. Accept the default configuration of the Debug Sampler.

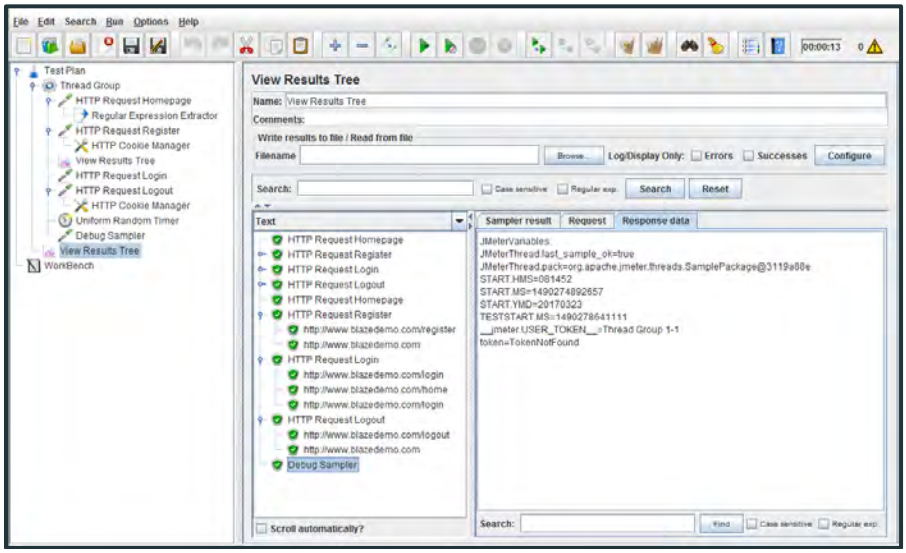


37. Click the last **View Results Tree** listener in the Test Plan pane to open the View Results Tree pane.

38. Click **Start** to run the test again. Your test results display in the View Results Tree pane.



39. Click **Response data** to view the variables that were used in the test and note the value captured for *token* in the Response data tab.



Tips and Tricks

- The best way to create an initial script before adding parameters is to record it using the BlazeMeter chrome extension recorder. See the knowledge base article Chrome Extension for more information.

- For more information about how to create login scenarios, see How To Use JMeter For Login Authentication. There are at least two good options to generate unique login credentials for multiple sign ups and logins. For this exercise, we used a comma separated value (CSV) file to create a data set which holds the username, email address, and password. For more information about the JMeter CSV Data Set Config option, see Using CSV Data Set Config in the BlazeMeter.com online knowledge base.

Lab 6 - Configuring Functions

Goals During this lab, you will learn about and apply these JMeter functions: log, time, counter & ThreadNum, UUID, Property, and instanceId.

Scenario Create a standard test in JMeter using these JMeter functions: log, time, counter & ThreadNum, UUID, Property, and instanceId.

You will accomplish this by completing the following tasks:

- Configure and Run a JMeter Test
- Configure a JMeter Test Using the Log Function
- Configure a JMeter Test Using the Time Function
- Configure a JMeter Test Using the Counter & ThreadNum Function
- Configure a JMeter Test Using the UUID Function
- Configure a JMeter Test Using the Property Function
- Configure a JMeter Test Using the Property Function with an instanceId

Time 20 minutes

Part 1: Configure and Run a JMeter Test

Create a New JMeter Test

Complete these steps to create a new JMeter test for this lab:

1. Launch **JMeter** from your desktop shortcut.
2. Select **File > Save Test Plan As**. The Test Plan.jmx window opens.
3. Enter **Lab1-6-Functions.jmx** in the File Name field and click **Save**.
   Note: *Save the lab periodically throughout this exercise to preserve your work.*

Add a Thread Group

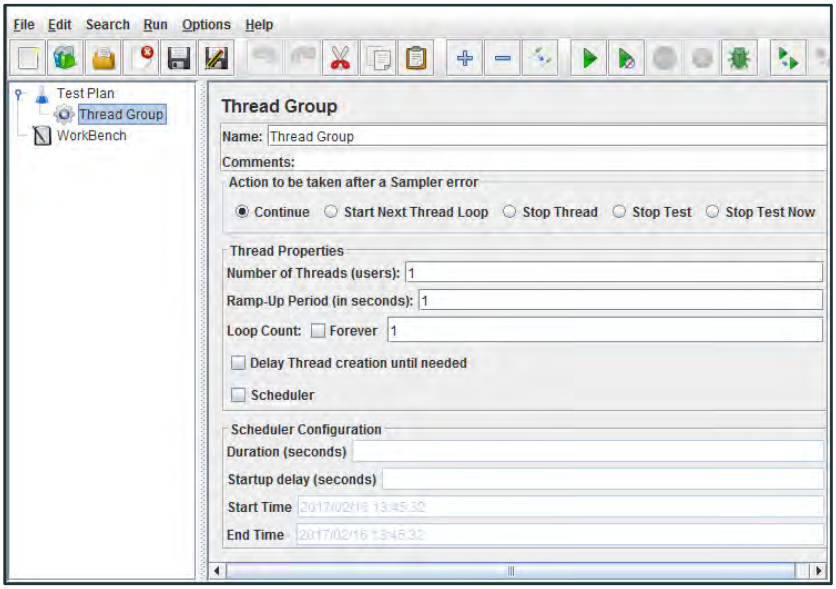
Complete these steps to add a Thread Group to your test:

4. Click **Test Plan** to select it.
5. Right-click **Test Plan** and select **Add ▶ Thread Groups ▶ Thread Group**. The Thread Group configuration pane opens.

6. Accept the default configuration for a Thread Group:

Field	Value
Number of Threads	**1**
Ramp-Up Period	**1**
Loop Count	**1**

Your Thread Group configuration should look like this:



Add a Dummy Sampler

Complete these steps to add a Dummy Sampler to your test:

7. Click **Thread Group** to select it.

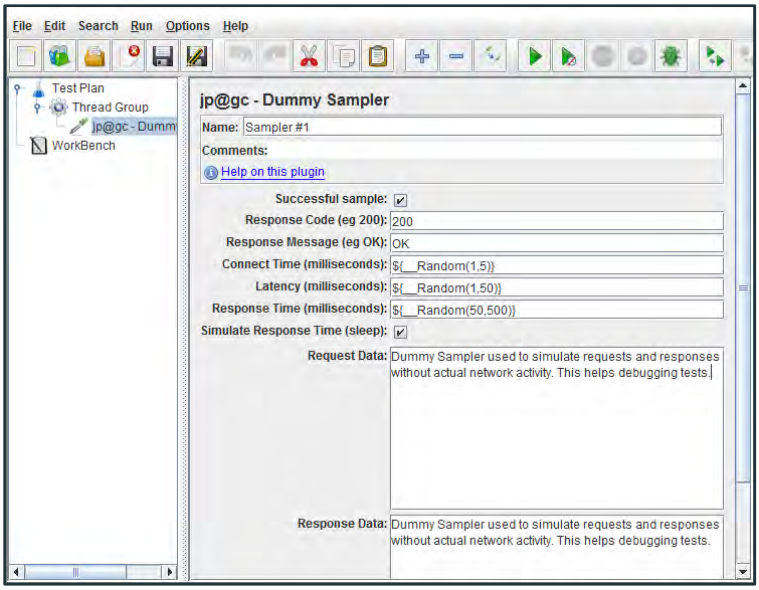
8. Right-click **Thread Group** and select **Add ▶ Sampler ▶ Dummy Sampler**. The Dummy Sampler configuration pane opens.

9. Verify that the Dummy Sampler is configured as follows with these default values:

Field	Value
Name	**Sampler #1**
Successful sample	**Checked**
Response Code	**200**
Response Message	**OK**
Connect Time	**${_Random(1,5)}**

Latency	**${_Random(1,50)}**
Response Time	**${_Random(0,500)}**
Simulate Response Time (sleep)	**Checked**
Request Data	**Dummy Sampler used to simulate requests and responses without actual network activity. This help debugging tests**
Response Data	**Dummy Sampler used to simulate requests and responses without actual network activity. This helps debugging test**

Your Dummy Sampler configuration should look like this:

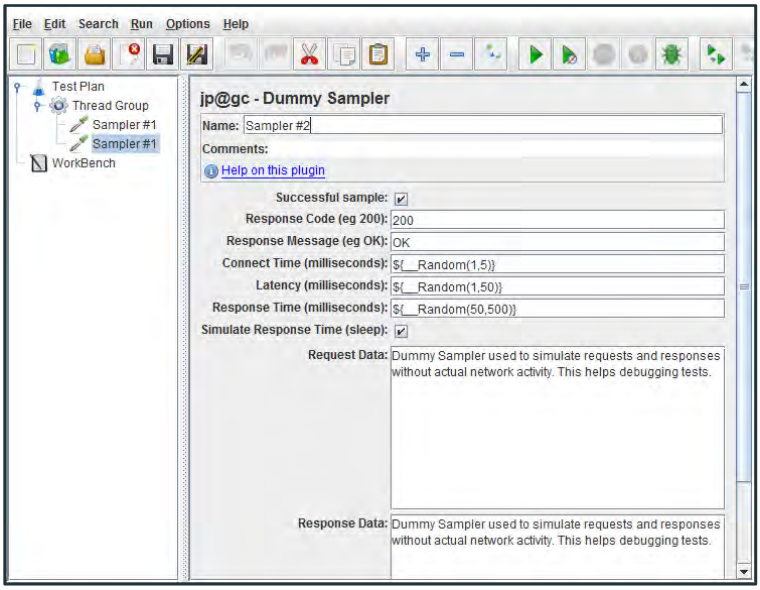


Duplicate the Dummy Sampler

Complete these steps to duplicate the Dummy Sampler that you just created:

10. Right-click Sampler #1 select **Duplicate**. A duplicate of Dummy Sampler 1 appears in the Thread Group.

11. Enter **Sampler #2** in the name field of the new Dummy Sampler. Your second Dummy Sampler configuration should look like this:

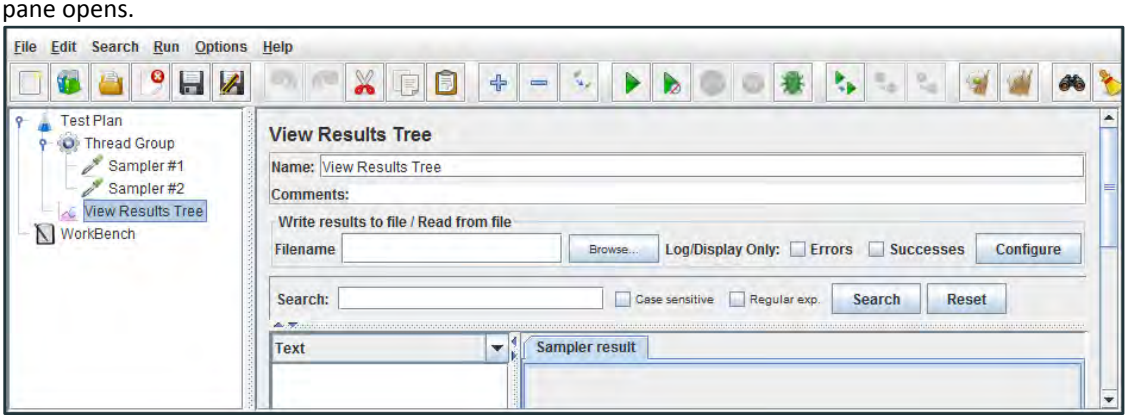


Add a View Results Tree Listener

Complete these steps to add a View Results Tree listener to your test that will capture and display your test results:

12. Click **Test Plan** to select it.

13. Right-click **Test Plan** and select **Add ▶ Listener ▶ View Results Tree**. The View Results Tree listener configuration pane opens.

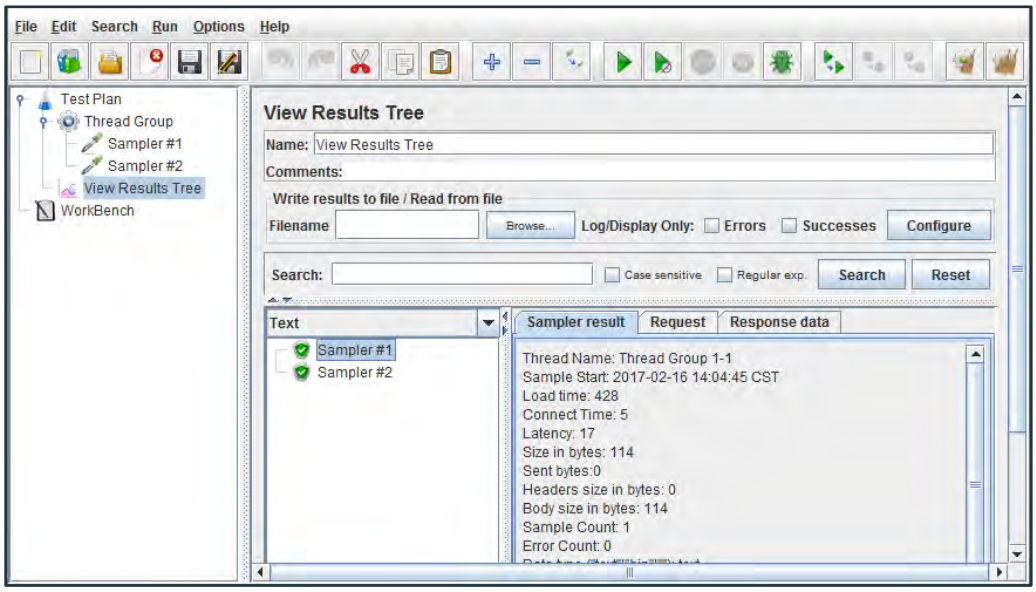


Run the Test and Observe the Test Results

Complete these steps to run your test and view the test results in the View Results Tree listener pane:

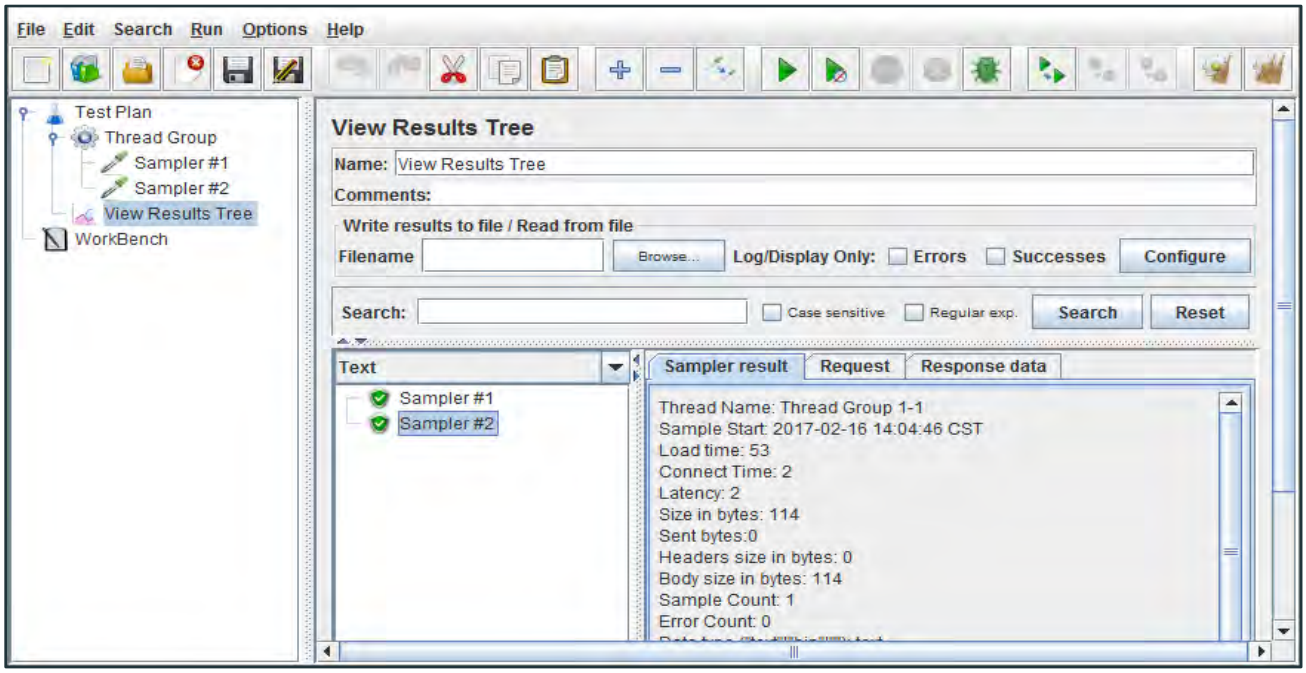
14. Click **View Results Tree** to view the test results pane.

15. Click **Start** to begin your test.



16. Click Sampler #1 and note the load time and connect time in the Sampler result tab.

17. Click **Sampler #2** and note the load time and connect time in the Sampler result tab.



Note: *Load time is given in milliseconds and connect time is given in seconds.*

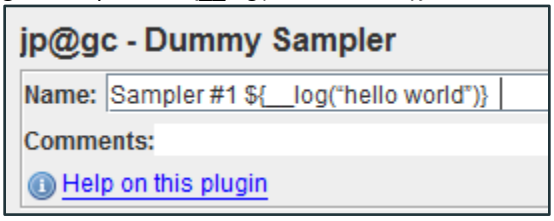
Part 2: Configure a JMeter Test Using the Log Function

Configure and Run the Log Function

Complete these steps to configure and run a Log function:

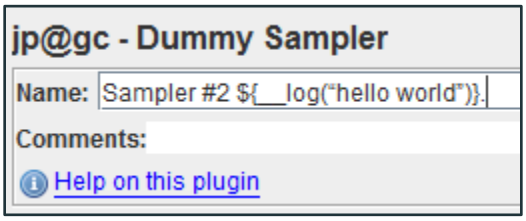
1. Click **Sampler #1** to view its configuration page. The Dummy Sampler configuration page opens.

2. Enter the log function **${__log("hello world")}** after Sampler #1 in the Name field. Your sampler name should now give *Sampler #1 ${__log("hello world")}*.



3. Click **Sampler #2** to view its configuration page. The Dummy Sampler configuration page opens.

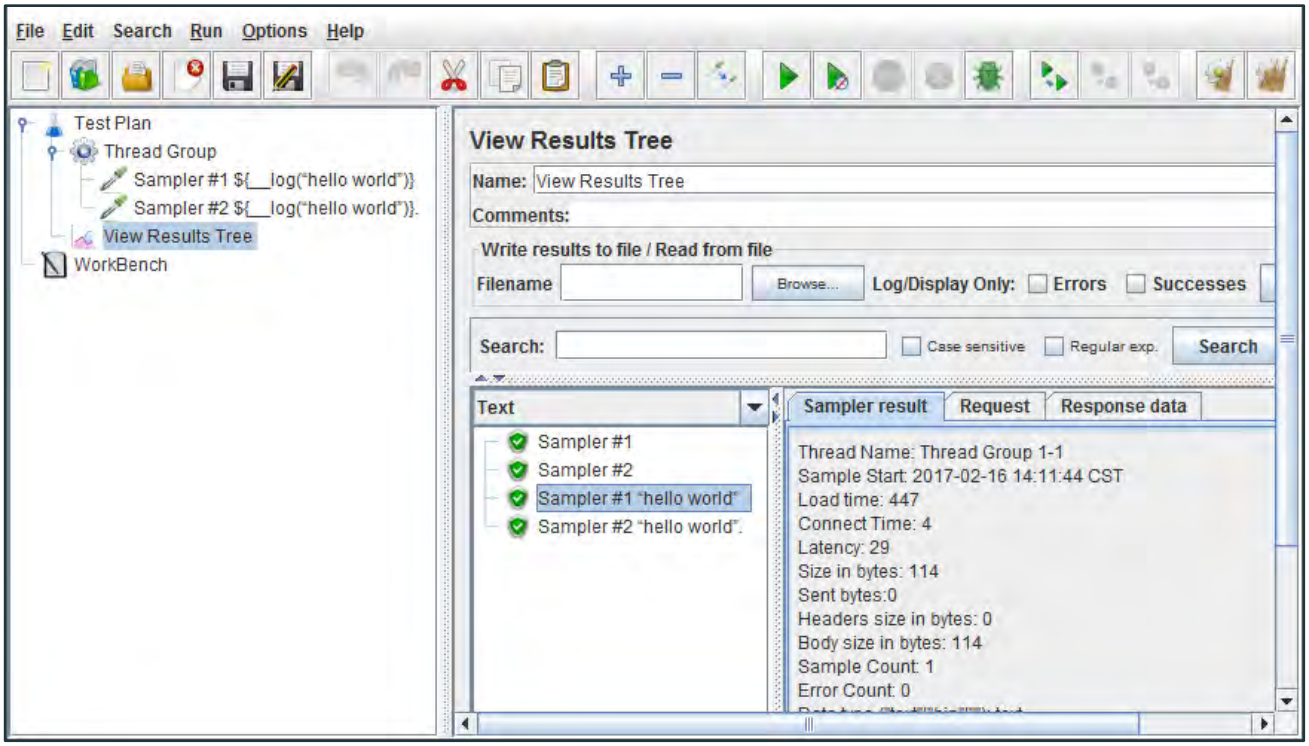
4. Enter the log function **${__log("hello world")}** after Sampler #2 in the Name field. Your sampler name should now give *Sampler #2 ${__log("hello world")}*.



5. Click **View Results Tree** to display the pane that will show your test results.

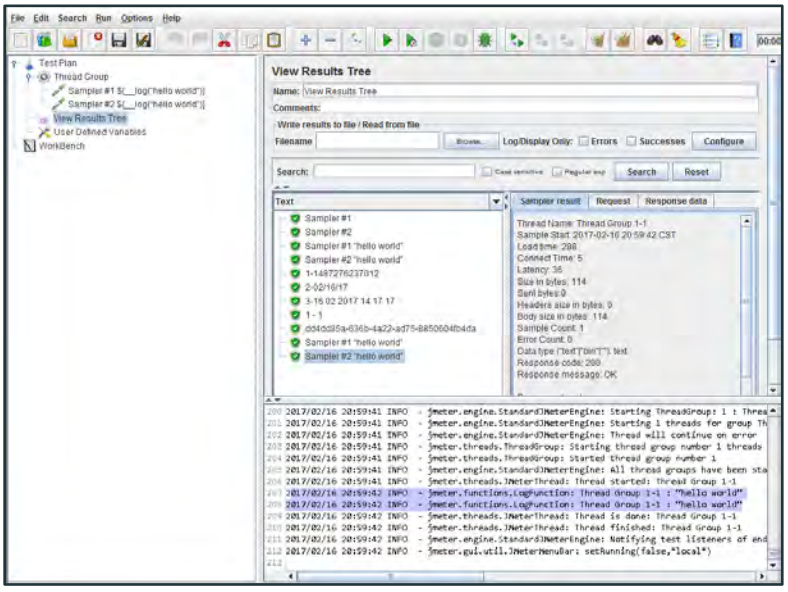
6. Click **Start** to begin your test.

7. View your test results and note that the log function added “hello world” to the name of each sampler.



8. Select **Options ▶ Log Viewer**. The log view opens at the bottom of the View Results Tree window.

9. In the Log Viewer, note the number of times that the thread group ran and where “hello world” appears in the log.



Part 3: Configure a JMeter Test Using the Time Function

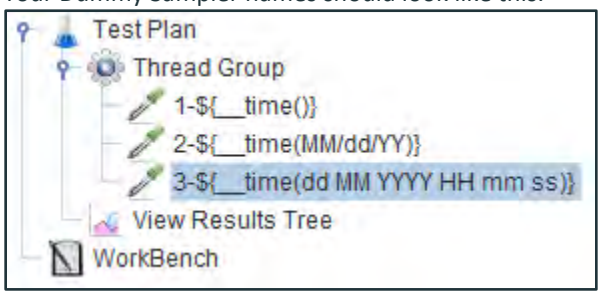
Configure and Run a Time Function

Complete these steps to configure and run a Time function:

1. Right-click **Sampler #2 ${__log("hello world")}** and select Duplicate to give you three dummy samplers in total.

2. Rename each sampler as follows:
   a. Change the name of sampler 1 from Sampler #1 ${__log("hello world")} to **1-${__time()}**.
   b. Change the name of sampler 2 from Sampler #2 ${__log("hello world")} to **2-${__time(MM/dd/YY)}**.
   c. Change the name of the sampler 2 duplicate from Sampler #2 ${__log("hello world")} to **3-${__time(dd MM YYYY HH mm ss)}**.

   Your Dummy Sampler names should look like this:

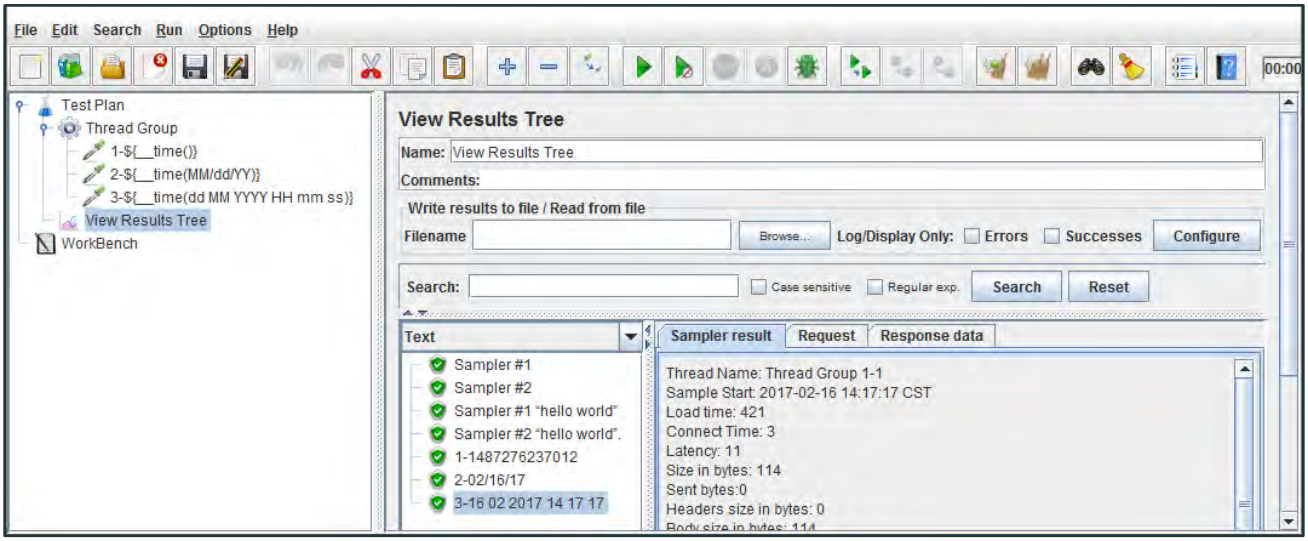


3. Click **View Results Tree** to view the test results pane.

4. Click **Start** to begin your test.

5. Note the test results in the View Results Tree pane:
   a. The first sampler gives the time stamp as its name. This is because we did not give the time function any other values to apply.
   b. The second sampler gives the current date as its name in the format 01/31/16.
   c. The third sampler gives the current date and time as its name in the format 01 31 2016 15 26 56.

Your test results should look like this:



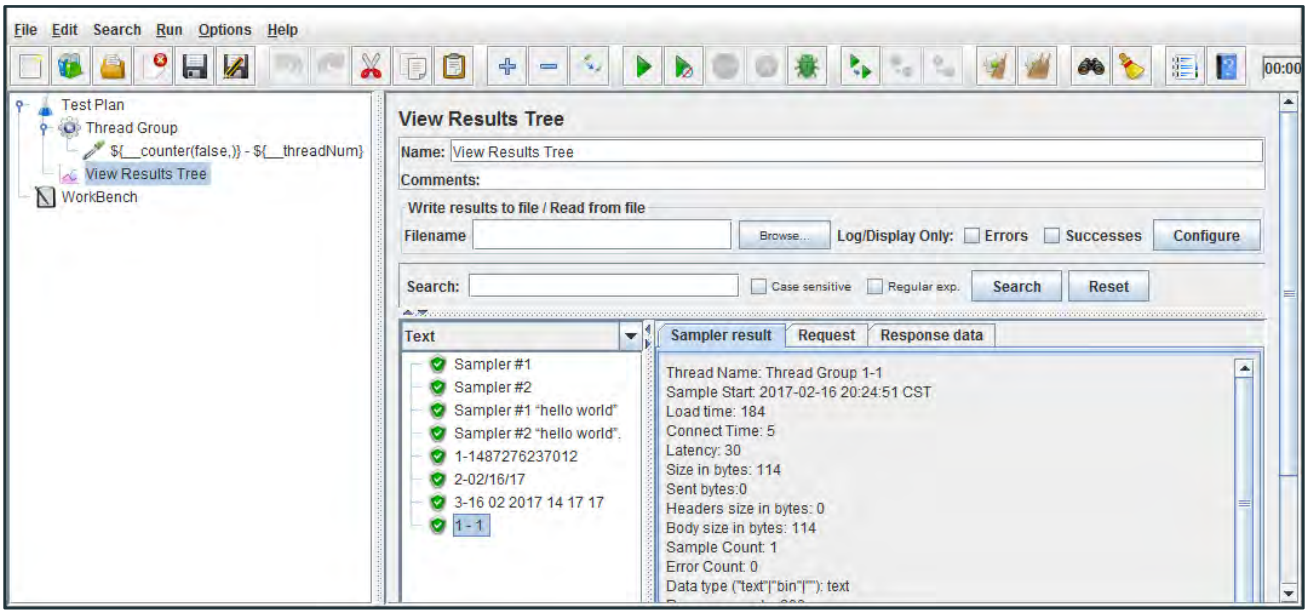
Part 4: Configure a JMeter Test Using the Counter & ThreadNum Function

Configure and Run a Counter & ThreadNum Function

Complete these steps to configure and run a Counter and ThreadNum function:

1. Select the second and third samplers in your thread group to highlight them.
2. Right-click the highlighted samplers and select **Remove**.
3. Click **Yes** in the popup window to finish removing two samplers. Sampler 1-${__time()} is the only remaining sampler.
4. Click sampler **1-${__time()}** to view its Dummy Sampler configuration pane.
5. Replace 1-${__time()} with **${__counter(false,)} - ${__threadNum}** in the Name field to rename the sampler.

   Note: *There is no need to make any other changes to the test. The thread group should still be configured to run one thread for one loop as it did in previously configured tests.*
6. Click **View Results Tree** to view the test results pane.
7. Click **Start** to begin your test.
8. Notice that the sampler name is **1 - 1** in the Text pane of the View Results Tree page. The first number 1 in the sampler name is a result of the counter function. The second number 1 in the sampler name gives the number of thread groups in the test as a result of the threadNum function.



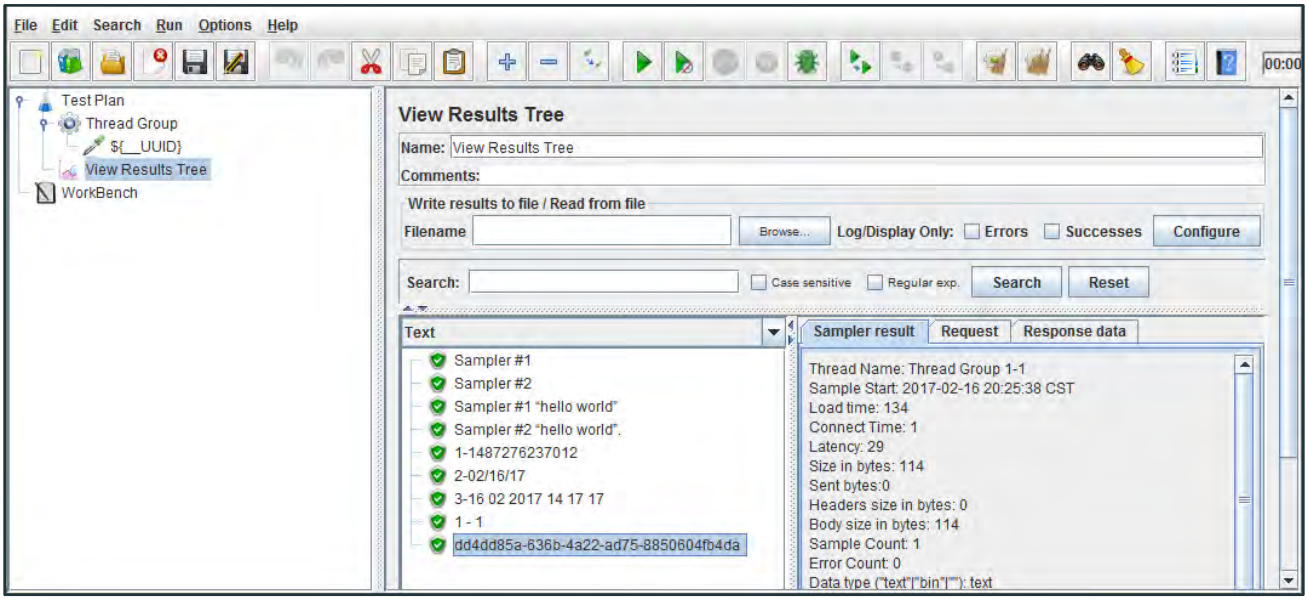
Part 5: Configure a JMeter Test Using the UUID Function

Configure and Run a UUID Function

The UUID function provides a unique value every time you use it. For example, a common use case of the UUID function occurs when you have a web application and you want to performance test the signup phase of a scenario. In this scenario, you may create 100 new users. If you include a new UUID every time that you provide a user name, then you can be assured that each username is unique.

Complete these steps to configure and run a UUID function:

1. Click sampler **${__counter(false,)} - ${__threadNum}** to view its Dummy Sampler configuration pane.
2. Replace ${__counter(false,)} - ${__threadNum} with **${__UUID}** in the Name field to rename the sampler.
3. Click **View Results Tree** to view the test results pane.
4. Click **Start** to begin your test.
5. View your test results in the View Results Tree pane and note that the function generates a 128-bit universally unique identifier. This identifier can be used to randomize logins or any other field when testing with a high level of concurrency.



Part 6: Configure a JMeter Test Using the Property Function

Configure and Run a Property Function

Complete these steps to configure and run a Property function:

1. Right-click **Test Plan** and select **Add ▶ Config Element ▶ User Defined Variables**. The User Defined Variables configuration pane opens.

2. Click **Add** to add a row to the User Defined Variables section.

3. Enter **threads** in the Name field of the new row.

4. Enter **${__P)PropNumOfThreads,5)}** in the Value field of the same row. You now have a Property function with two attributes, a name of *threads* and a default value of *5*, added to your test.

5. Click **Thread Group** to view the Thread Group configuration pane.

6. Enter **${threads}** in the Number of Threads field. The Thread Group will now use the default value of 5 that you set in your Property function to determine how many threads to run in your test.

7. Go to the command line on your system.

8. From the command line, go to your **JMeter/bin** folder.

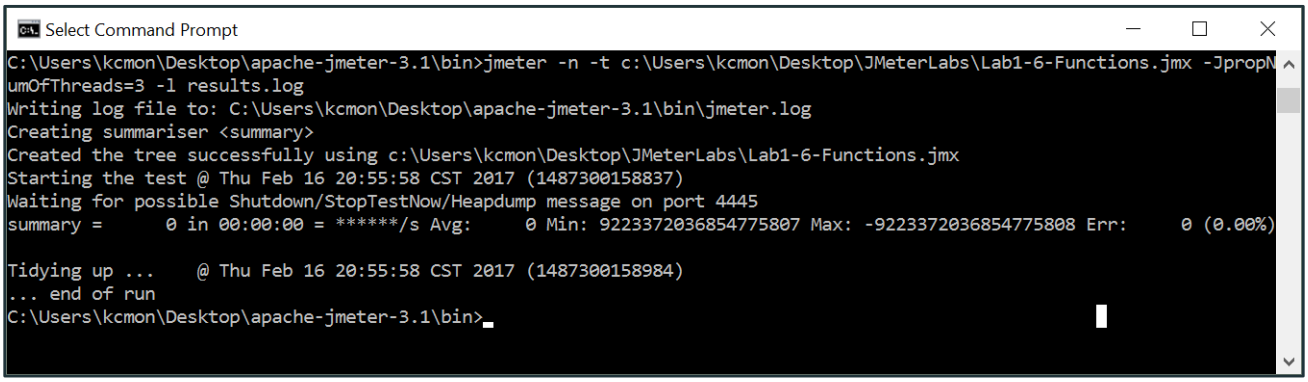
9. Enter **jmeter -n -t *</path to the location of your Functions_lab.jmx file>*/Functions_lab.jmx -JpropNameOfThreads=3 -l results.log** at the command line prompt and run the command.

   For example, *jmeter -n -t /Users/jsmith/downloads/Functions_lab.jmx -JpropNumOfThreads=3 -l results.log*.

When you run the .jmx script, it changes the default value that you set in the Property function from 5 to 3 and writes the results to the results.log file.

Note: *The variable name given at the command line must match the variable name in your JMeter test exactly. Variable names are case sensitive.*

10. View the test results displayed in the command line window as the command is executed. Note that the value of 3 was applied to your Property function from the command line. This value populated the threads variable to set the number of threads to execute, overriding the value of 5 that you originally configured.



Part 7: Configure a JMeter Test Using the Property Function with an instanceId

Configure and Run a Property Function with an instanceId

In this lab, you will use the instanceId function to verify which instance is giving errors or behaving strangely. For example, if you want to debug a specific engine because you are receiving errors on one engine, but not other engines, you can use the instanceId function to identify the engine that is not working properly.

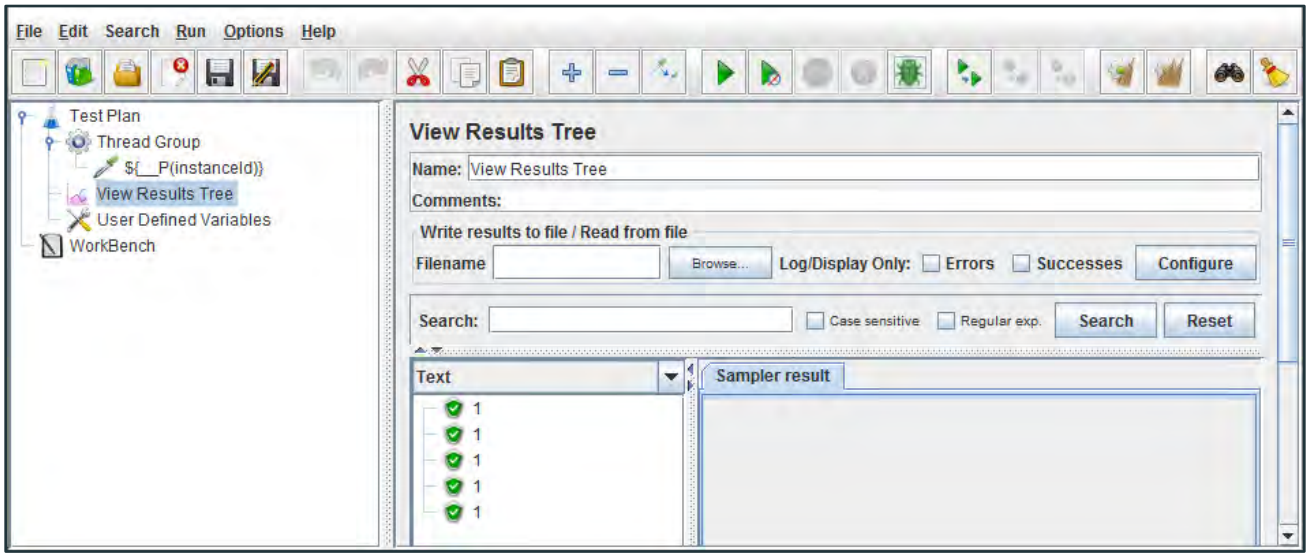
You can do this by running a test using multiple load engines. The instanceId function identifies each specific engine or instance. For example, if we use a distribution of three load engines, then their instance IDs would be 1, 2, and 3, respectively, allowing you to identify the malfunctioning engine.

Complete these steps to incorporate the instanceId function into your Property function:

1. Return to your previous instance of JMeter.
2. Click **${_UUID}** to view its Dummy Sampler configuration page.
3. Enter **${__P(instanceId)}** in the Name field to rename the Dummy Sampler.
4. Click **View Results Tree** to view the test results pane.

5. Click **Start** to begin your test.

6. View your test results in the View Results Tree pane and note that you receive the value of 1 five times. This occurs because you are only running one engine on your local computer for five threads.



Lab 7 - Configuring Complex Script

Goals During this lab, you will learn how build a script that includes several scenarios that run consecutively.

Scenario This lab builds on what you learned in JMeter Lab 1-5 – Correlations by taking an advanced, realistic scenario a step further. In this lab, you will apply your new knowledge of thread groups, assertions, and so on with additional skills such as incorporating a CSV file. You will also learn about and apply a Throughput Controller and a Logic Controller while incorporating a BeanShell Sampler.

- Configure a JMeter Test that Applies Multiple Throughput Controllers
- Configure a JMeter Test that Applies a Logic Controller and a BeanShell Sampler

Time 30 minutes

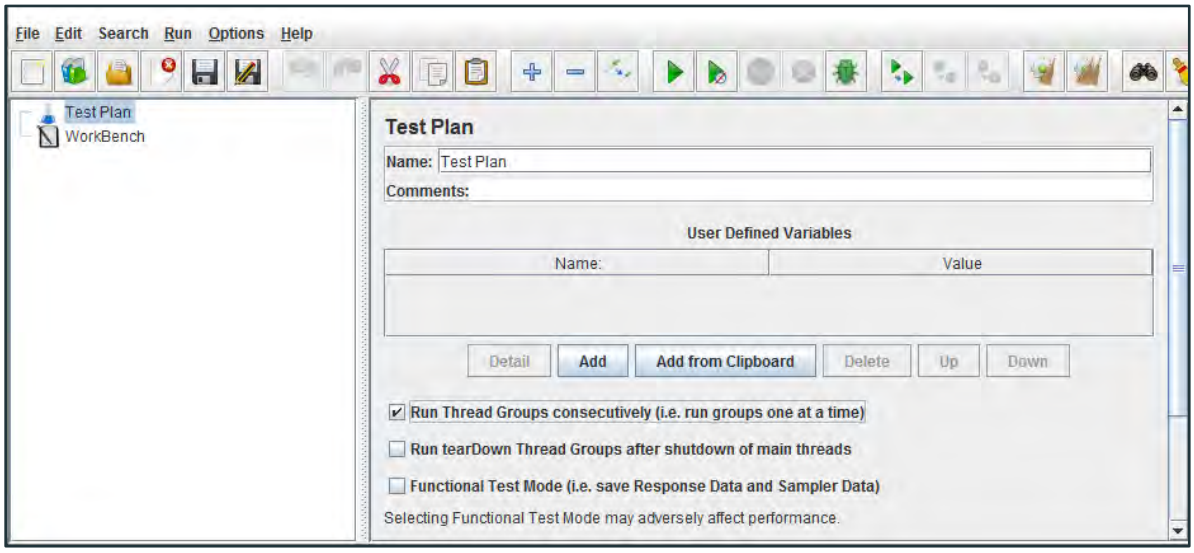
Part 1: Configure a JMeter Test that Applies Multiple Throughput Controllers

Create and Configure a New JMeter Test

Complete these steps to configure a Thread Group for ten threads with a ten second ramp up that runs for ten minutes:

1. Launch **JMeter** from your desktop shortcut.
2. Select **File ▶ Save Test Plan As**. The Test Plan.jmx window opens.
3. Enter **Lab1-7-ComplexScripts.jmx** in the File Name field and click **Save**.
   Note: *Save the lab periodically throughout this exercise to preserve your work.*
4. Click **Test Plan** to view the Test Plan configuration pane.

Check the **Run Thread Groups consecutively (i.e. run groups one at a time)** check box.



Add a Thread Group

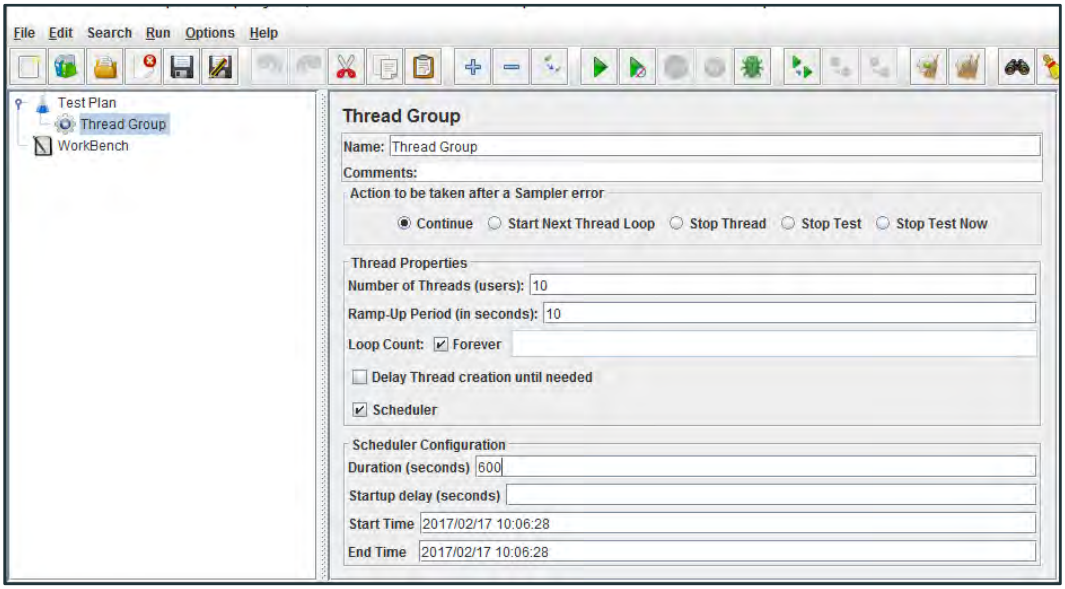
Complete these steps to add a Thread Group:

5. Right-click **Test Plan** and select **Add ▶ Threads (Users) ▶ Thread Group**. The Thread Group configuration pane opens.

6. Configure your Thread Group as follows:

Field	Value	Notes
Number of Threads	**10**	
Ramp-Up Period	**10**	
Loop Count: Forever	**Checked**	
Scheduler	**Checked**	
Duration	**600**	This value configures the test to run for ten minutes.

Your Thread Group configuration should look like this:



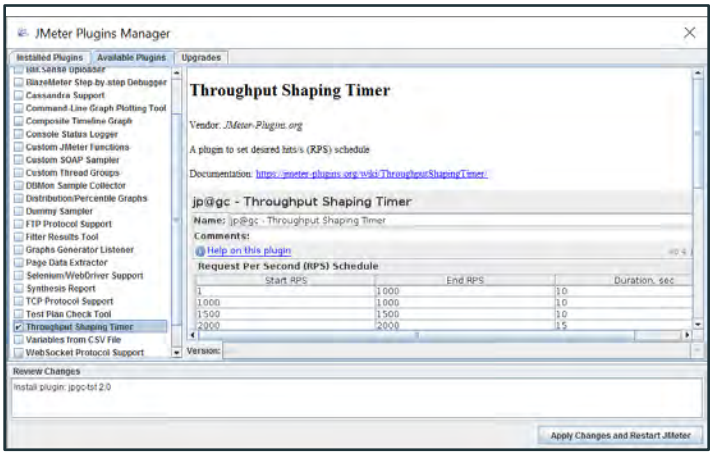
Modify the Test Throughput with a Throughput Shaping Timer

Complete these steps to add the Throughput Shaping Timer plug-in and limit the throughput of the test to 5 hits per second (hits/s) for the first five minutes. After five minutes, ramp up to 10 requests per second (RPS) until the end of the test.

7. Select **Options ▶ Plug-In Manager**. The JMeter Plugins Manager opens.

8. Click **Available Plugins** and scroll down to **Throughput Shaping Timer**.

   Note: *If the plugin is not found, click Installed Plugins and verify that it has already been installed and proceed to the next step.*Check the **Throughput Shaping Timer** checkbox and click **Apply Changes and Restart JMeter** to install the plugin.



9. Right-click **Thread Group** and select **Add ▶ Timer ▶ Throughput Shaping Timer**. The Throughput Shaping Timer configuration pane opens.

10. Click **Add Row** two times to add two rows to the Requests Per Second (RPS) section.

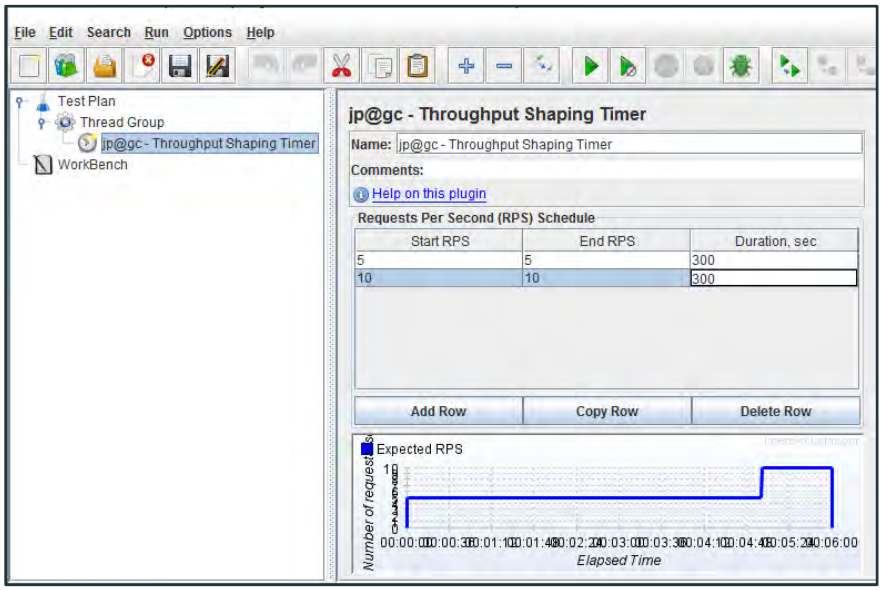
11. Configure the first row in the Requests Per Second (RPS) section as follows:

Field	Value
Start RPS	**5**
End RPS	**5**
Duration	**300**

12. Configure the second row in the Requests Per Second (RPS) section as follows:

Field	Value
Start RPS	**10**
End RPS	**10**
Duration	**300**

Your Throughput Shaping Timer configuration should look like this:



Add a Throughput Controller

Why should you use a Throughput Controller? A Throughput Controller allows you to create tests that imitate virtual users completing specific tasks to test the performance of a website.

For example, you may want to test the performance of an ecommerce site in a realistic scenario and show 1000 concurrent users accessing the website in a realistic manner. You may assume that 30% of the users are currently logging in, 20% of the users are already logged in and busy adding items to their cart, 40% of the users are browsing and searching through your products, and 20% are logging out. In this example, a Throughput Controller could imitate this behavior to thoroughly test the performance of the site.

Complete these steps to add a Throughput Controller that sends 10% of test plan requests to the BlazeMeter demonstration registration web page and signs up the users:

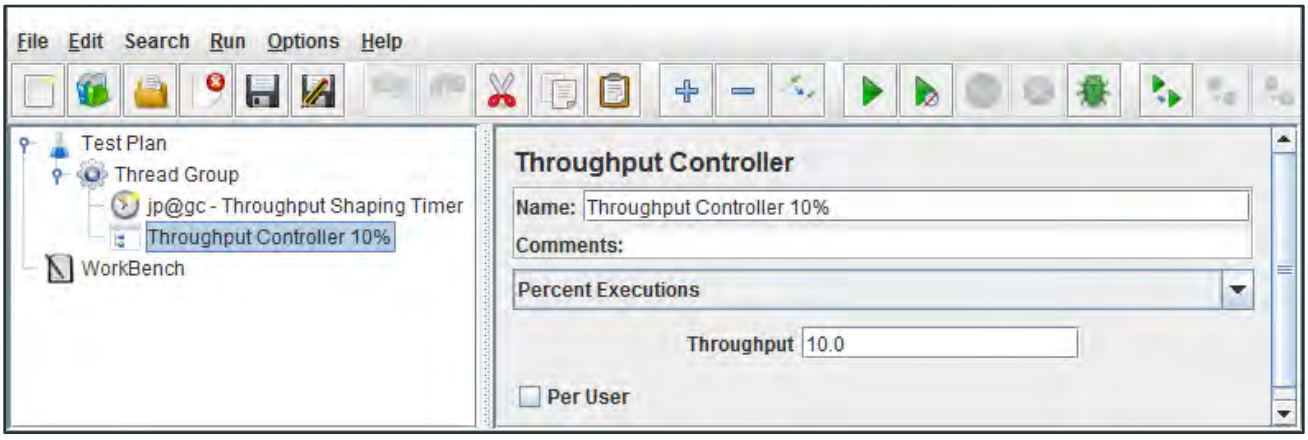
13. Click **Thread Group** to select it.

14. Right-click **Thread Group** and select **Add ▶ Logic Controller ▶ Throughput Controller**. The Throughput Controller configuration page opens.

15. Configure your Throughput Controller as follows:

Field	Value	Notes
Name	**Throughput Controller 10%**	
Percent Executions	**Selected**	
Throughput	**10.0**	This value sets the throughput at 10%.

Your Throughput Controller configuration should look like this:



Add an HTTP Request

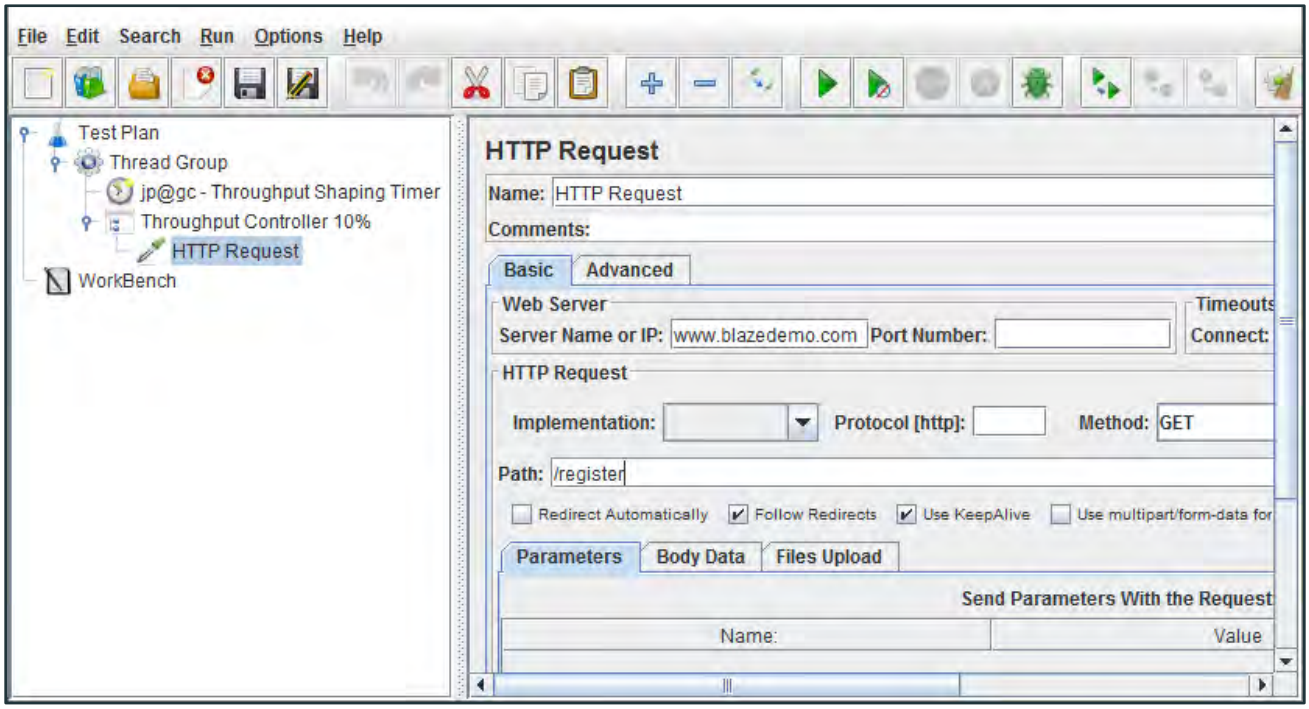
Complete these steps to add an HTTP Request to your Throughput Controller:

16. Right-click **Throughput Controller** and select **Add ▶ Sampler ▶ HTTP Request**. The HTTP Request configuration pane opens.

17. Configure your HTTP Request as follows:

Field	Value
Server or IP	**blazedemo.com**
Method	**GET**
Protocol	**http**
Path	**/register**

Your HTTP Request configuration should look like this:



Add an HTTP Cookie Manager

Complete these steps to add a Cookie Manager to your Throughput Controller:

18. Right-click **HTTP Request** and select **Add ▶ Config Element ▶ HTTP Cookie Manager**. The HTTP Cookie Manager configuration pane opens.

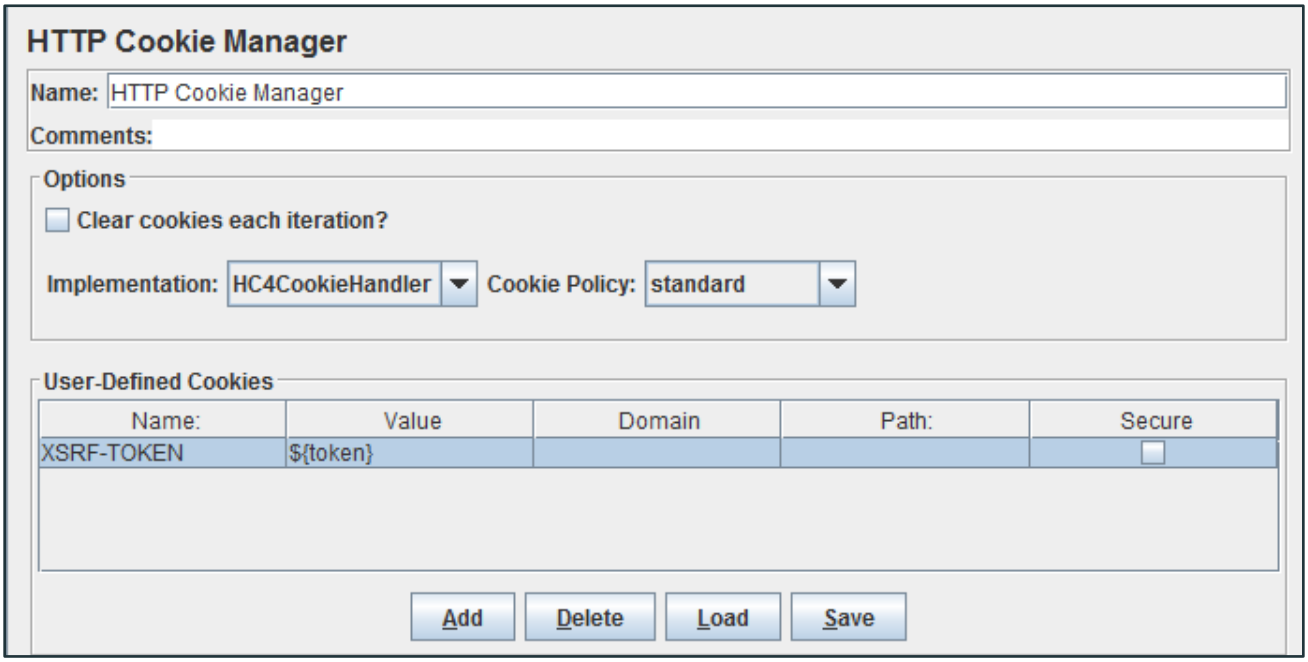
19. Configure your HTTP Cookie Manager as follows:
    a. Verify and accept these values that are configured by default:

Field	Value
Clear cookies each iteration?	**Unchecked**
Implementation type	**HC4CookieHandler**
Cookie Policy	**Standard**

b. Click **Add** to add a row to the User-Defined Cookies section.

Field	Value
Name	**XSRF-TOKEN**
Value	**${token}**

Your HTTP Cookie Manager configuration should look like this:



Add a Simple Data Writer Listener to Download Content

Complete these steps to configure your test to download all page content and save it to a file:

20. Click **HTTP Request** to select it.

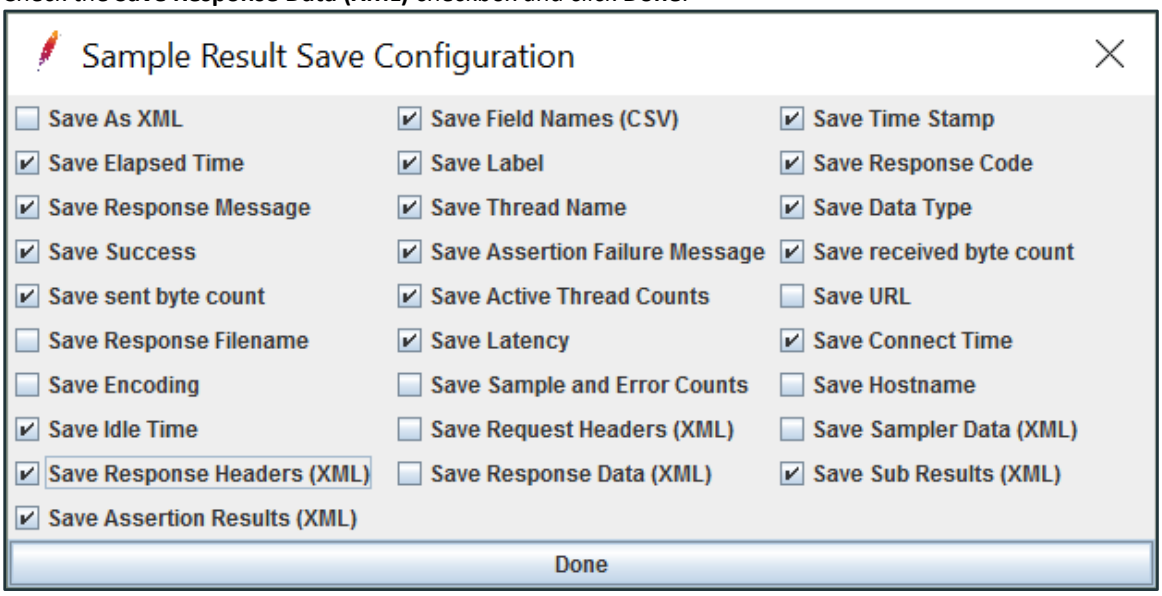
21. Right-click **HTTP Request** and select **Add ▶ Listener ▶ Simple Data Writer**. The Simple Data Writer configuration pane opens.

    Note: *The Simple Data Writer listener can be used to write the data from a request to a file.*

22. Enter **page_content.txt** in the Filename field.

23. Click **Configure**. The Sample Request Save Configuration window opens.

24. Check the **Save Response Data (XML)** checkbox and click **Done**.



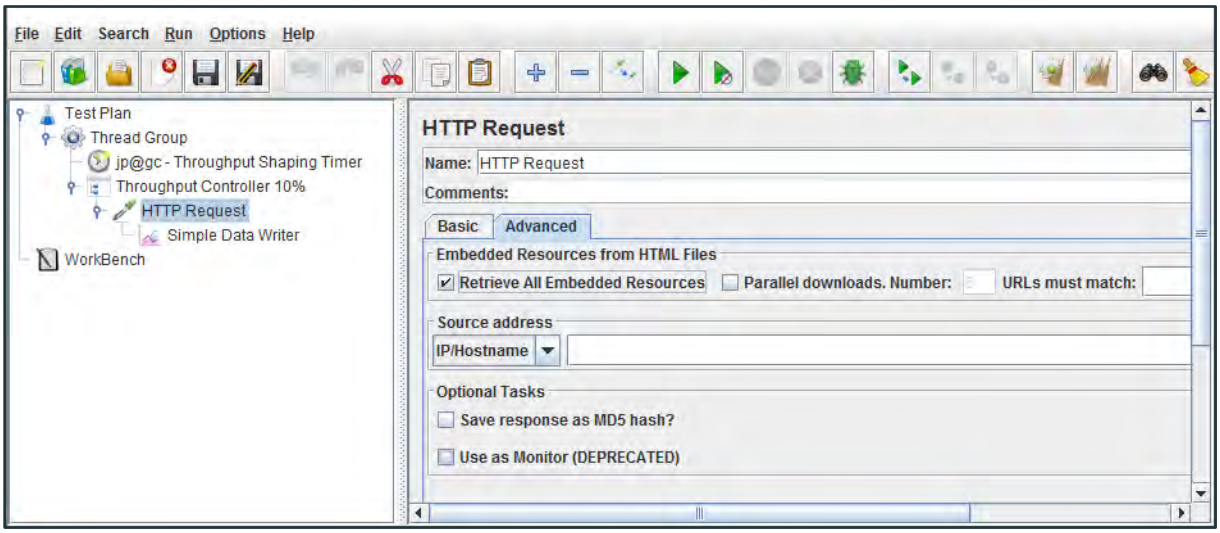
Configure the Advanced Tab of the HTTP Request

Complete these steps to configure the Advanced tab of the HTTP Request:

25. Click **HTTP Request** to view the HTTP Request configuration pane.

26. Click the **Advanced** tab to view the advanced configuration options of the HTTP Request.

27. Check the **Retrieve All Embedded Resources** checkbox to include all embedded resources when downloading page content.



Modify an HTTP Request to Identify Unique Users

Complete these steps to ensure that each user that is registered for the BlazeMeter Demo is unique no matter how many engines or threads are used to run the script:

28. Click the **Basic** tab of the same HTTP Request.

29. Click **Add** five time to add five rows to the Send Parameters With the Request section.

30. Configure row one as follows:

Field	Value
Name	**name**
Value	**Jane Smith ${__UUID}**

31. Configure row two as follows:

Field	Value
Name	**company**
Value	**BlazeMeter**

32. Configure row three as follows:

Field	Value
Name	**email**
Value	**jsmith_${__UUID}@blazemeter.com**

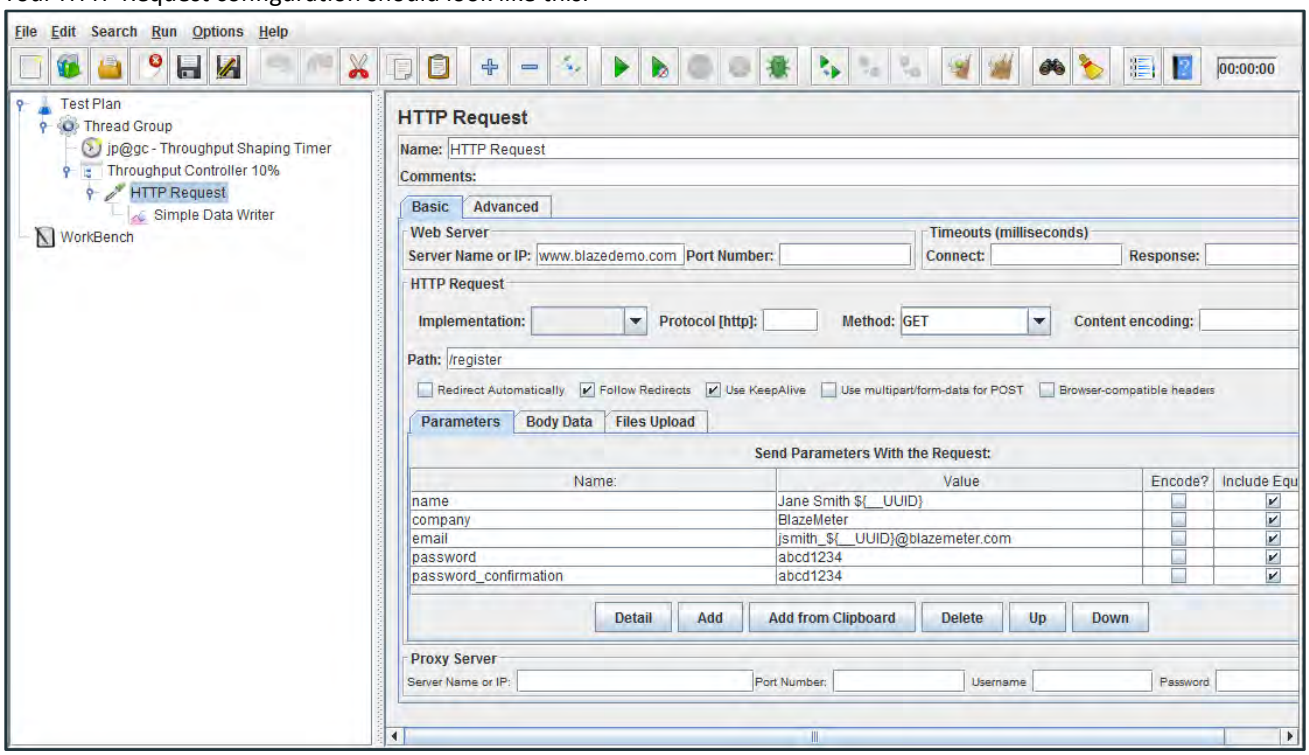
33. Configure row four as follows:

Field	Value
Name	**password**
Value	**abcd1234**

34. Configure row five as follows:

Field	Value
Name	**password_confirmation**
Value	**abcd1234**

Your HTTP Request configuration should look like this:



Part 2: Configure a JMeter Test that Applies a Logic Controller and a BeanShell Sampler

Submit a Request to a Website with Wait Time Requirements

The tasks that you will complete in this section of the lab apply a BeanShell Sampler. Elements such as BeanShell Samplers add functionality to JMeter scripts that are not available in JMeter by default. A BeanShell Sampler allows you to add scripting language and is like Java in that it enables you to do more with basic JMeter elements. It would be extremely difficult to complete these tasks in JMeter without a BeanShell Sampler.

Complete these steps to send a request to Blazedemo.com with a wait time between 500 milliseconds and 1 second occurring between each iteration with an average wait time of 750 milliseconds:

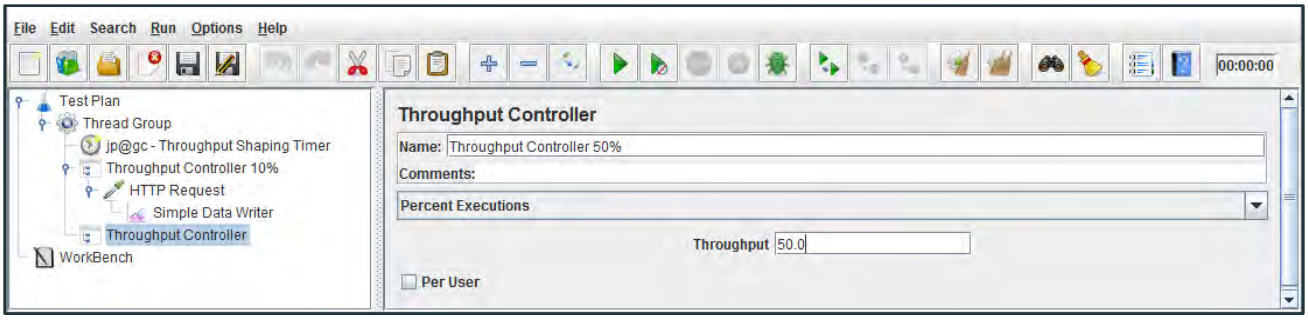
35. Click **Thread Group** to select it.

36. Right-click **Thread Group** and select **Add ▶ Logic Controller ▶ Throughput Controller**. The Throughput Controller configuration page opens.

37. Configure the Throughput Controller as follows:

Field	Value	Notes
Name	**Throughput Controller 50%**	This Throughput Controller will address 50% of your requests.
Selected from dropdown menu	**Percent Executions**	
Throughput	**50.0**	This value sets the throughput to 50%.

Your Throughput Controller configuration should look like this:



38. Right-click **Throughput Controller 50%** and select **Add ▶ Sampler ▶ HTTP Request**.

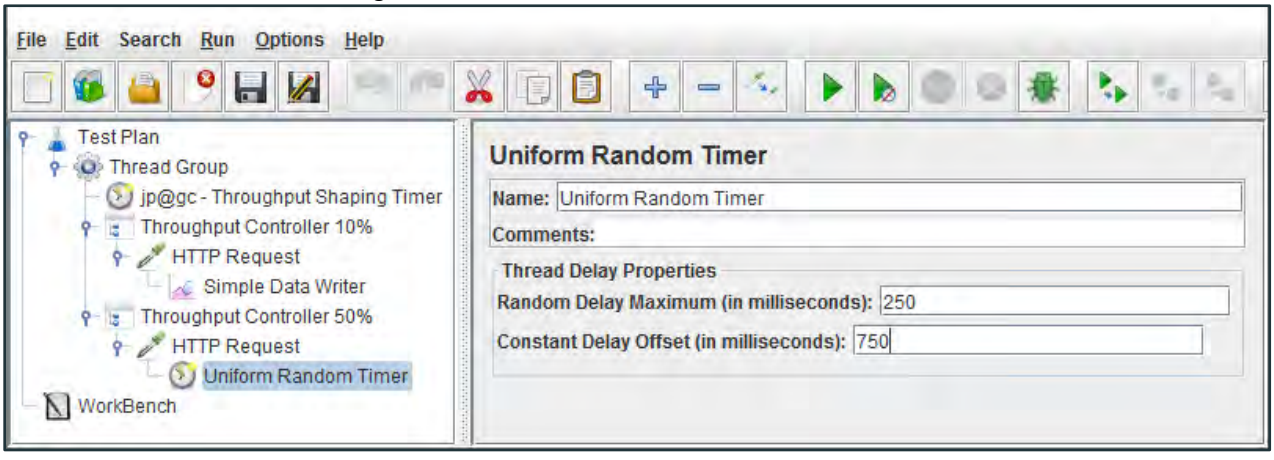
39. Enter **www.blazedemo.com** in the Server Name or IP field.

40. Right-click **HTTP Request** and select **Add ▶ Timer ▶ Uniform Random Timer**.

41. Configure the Uniform Random Timer as follows:

Field	Value
Random Delay Maximum	**250**
Constant Delay Offset	**750**

Your Uniform Random Timer configuration should look like this:



The result of setting a Random Delay Maximum of 250ms and a Constant Delay Offset of 750ms is that the maximum delay will not be longer than one second, the minimum delay will not be less than 500ms, and the average delay will be 750ms.

Submit a Request to a Website

Complete these steps to loop the test thirty times, read a CSV file, push the resulting content to a log, and then apply a command line parameter that determines if it is necessary to loop again:

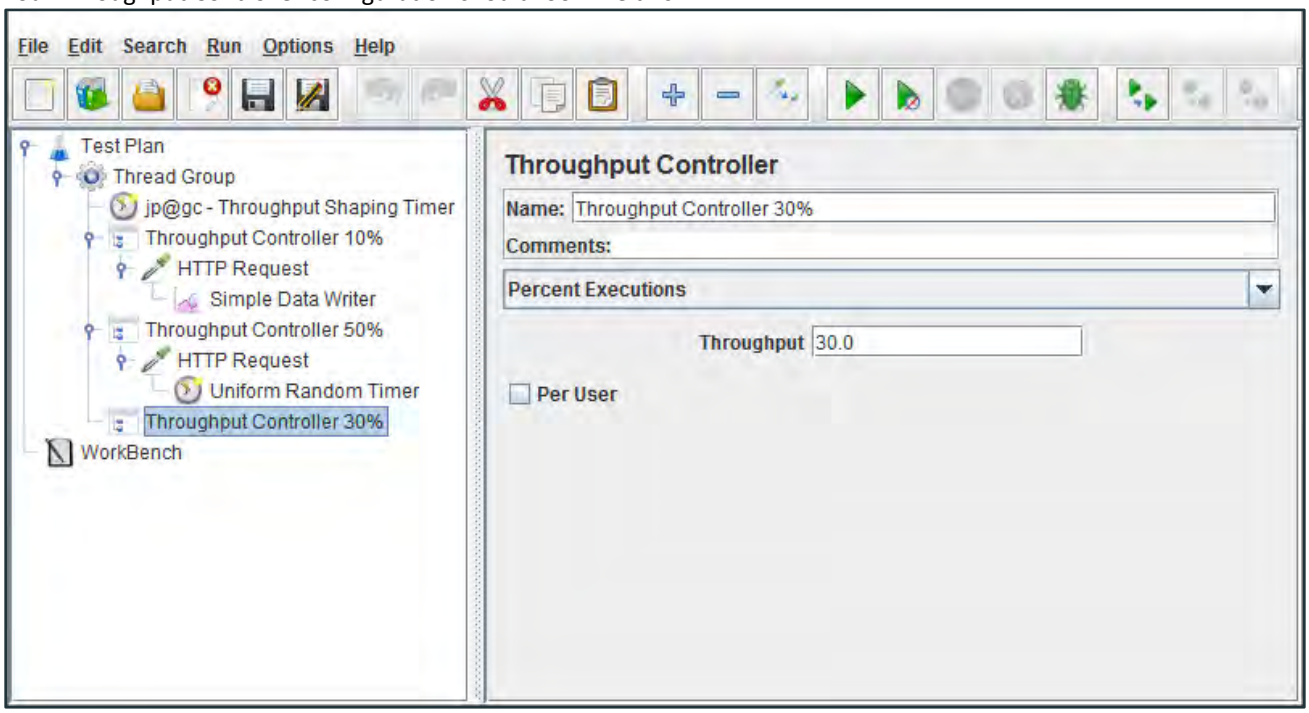
42. Click **Thread Group** to select it.

43. Right-click **Thread Group** and select **Add** ▶ **Logic Controller** ▶ **Throughput Controller**. The Throughput Controller configuration page opens.

44. Configure the Throughput Controller as follows:

Field	Value	Notes
Name	**Throughput Controller 30%**	This Throughput Controller will address 30% of your requests.
Selected from dropdown menu	**Percent Executions**	
Throughput	**30.0**	This value sets the throughput to 30%.

Your Throughput Controller configuration should look like this:



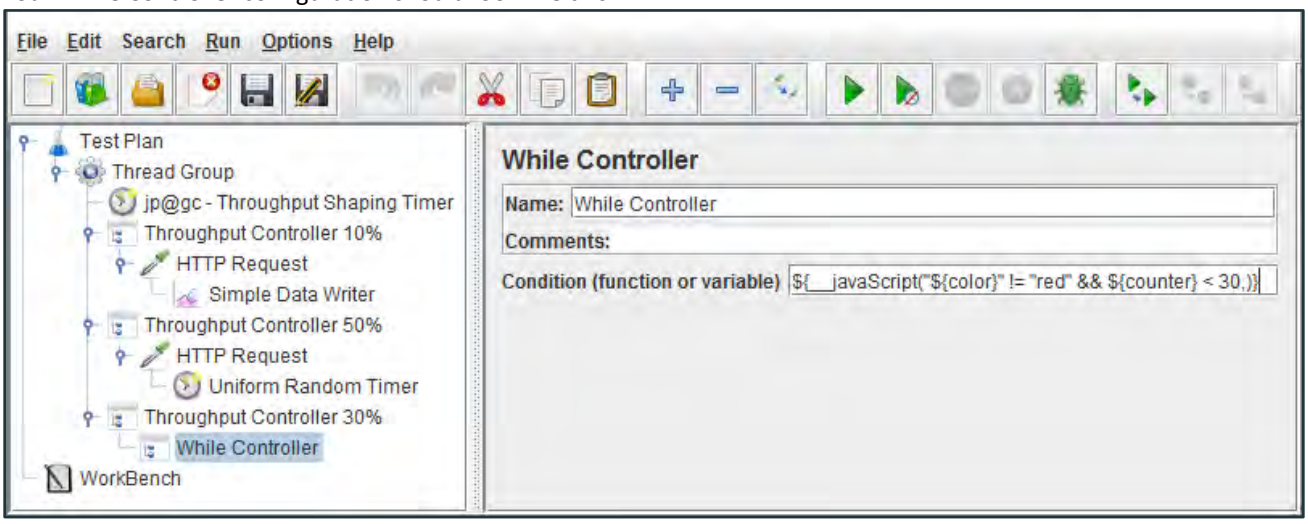
Add a While Controller

Complete these steps to add and configure a While Controller for your Throughput Controller:

45. Right-click **Throughput Controller 30%** and select **Add ▶ Logic Controller ▶ While Controller**. The While Controller configuration pane opens.

46. Enter **${__javaScript("${color}" != "red" && ${counter} < 30,)}** in the Condition (function or variable) field. Applying this function allows the loop to run thirty times unless the color is red. When the color is red, the function exits the loop.

    Your While Controller configuration should look like this:



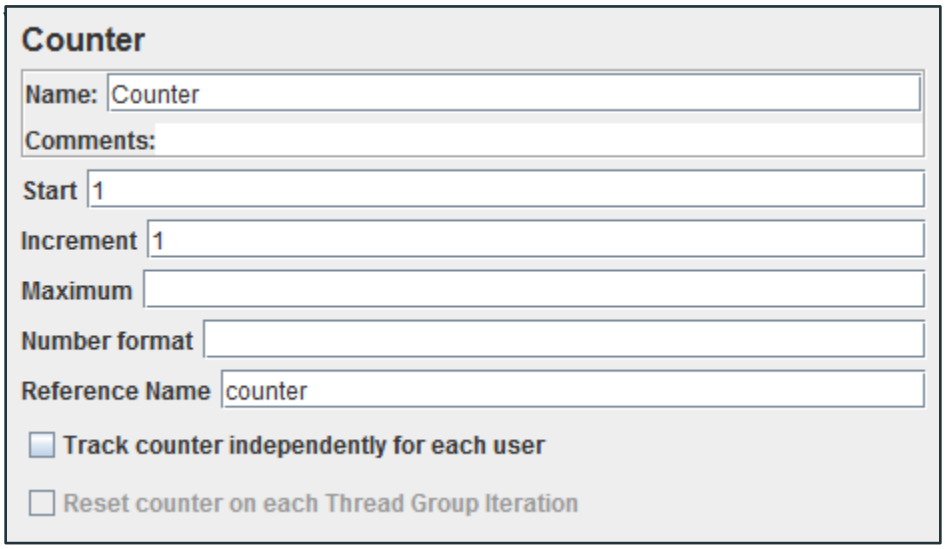
Add a Counter to Your Throughput Controller

Complete these steps to add a Counter to your Throughput Controller:

47. Right-click **While Controller** and select **Add ▶ Config Element ▶ Counter**. The Counter configuration pane opens.

48. Configure the Counter as follows:

Field	Value
Start	**1**
Increment	**1**
Reference Name	**counter**

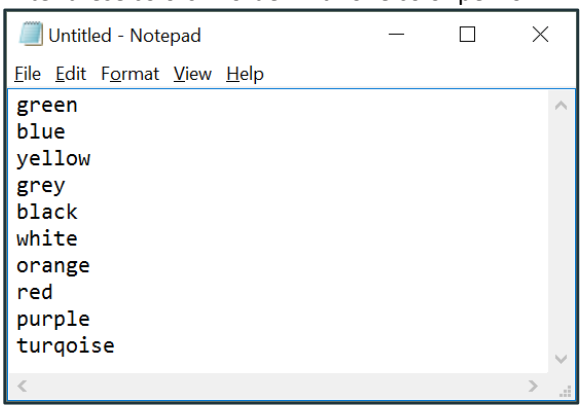


Create a CSV File for Colors

Complete these steps to create a CSV file that will be used by the While Controller function:

49. Open a basic word processor such as **Notepad**.

50. Enter these colors in order with one color per row in Notepad:



51. Select **File** ▶ **Save As** and enter **colors.csv** in the Filename field.

52. Navigate to the directory that contains the script that you are configuring for this lab, click **Save** to save the colors.csv file in the same directory, and close Notepad.

    Note: *You must save the colors.csv file to the same directory containing the script that you are creating for this lab to execute the script properly.*

Include a CSV File in a While Controller

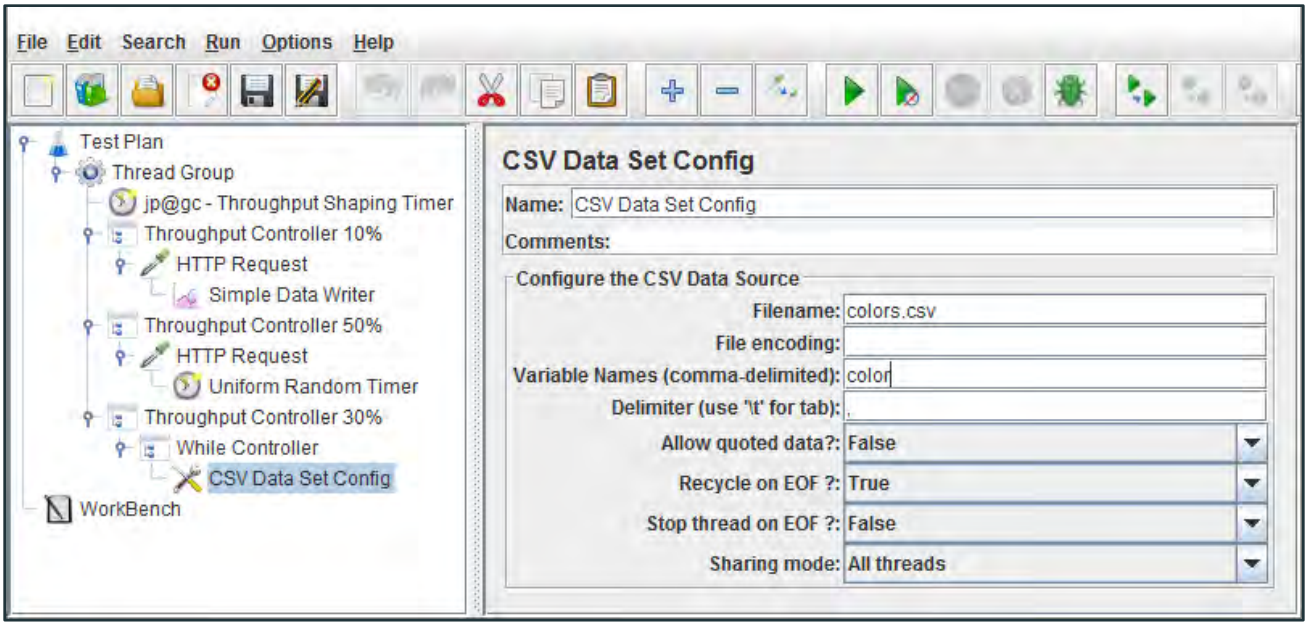
Complete these steps to add a CSV Data Set Config to the While Controller:

53. Right-click **While Controller** and select **Add ▶ Config Element ▶ CSV Data Set Config**. The CSV Data Set Config pane opens. This element is used to read lines of content from a comma separated value (CSV) file.

54. Configure the CSV Data Set Config as follows:

Field	Value
Filename	**colors.csv**
Variable Names	**color**

Your CSV Data Set Config configuration should look like this:



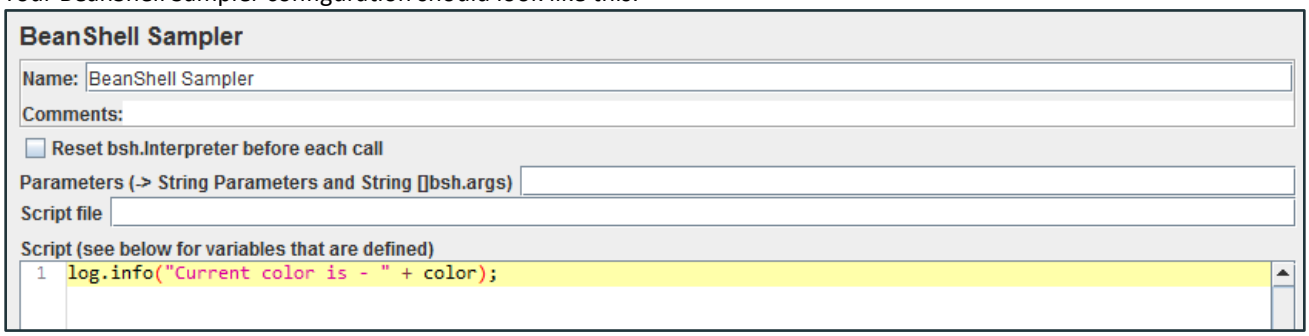
Add a BeanShell Sampler

When you execute your test, the lines of the colors.csv file that you just created will be read one-by-one. You will use a BeanShell Sampler to push the content of the line currently being read to the log. Complete these steps to add and configure a BeanShell Sampler:

55. Right-click **While Controller** and select **Add ▶ Sampler ▶ BeanShell Sampler**.

56. Enter **log.info("Current color is - " + color);** in the Script section of the BeanShell Sampler configuration pane.

Your BeanShell Sampler configuration should look like this:



Add a Dummy Sampler

Complete these steps to add a Dummy Sampler to your test:

57. Right-click **While Controller** and select **Add ▶ Sampler ▶ Dummy Sampler**. The Dummy Sampler configuration pane opens.

58. Enter **${color} - ${__P(instanceId)}** in the Name field of the Dummy Sampler. This step pushes the content of the current line being read in the CSV file to the log file.

Add Another Throughput Controller

Complete these steps to add another Throughput Controller:

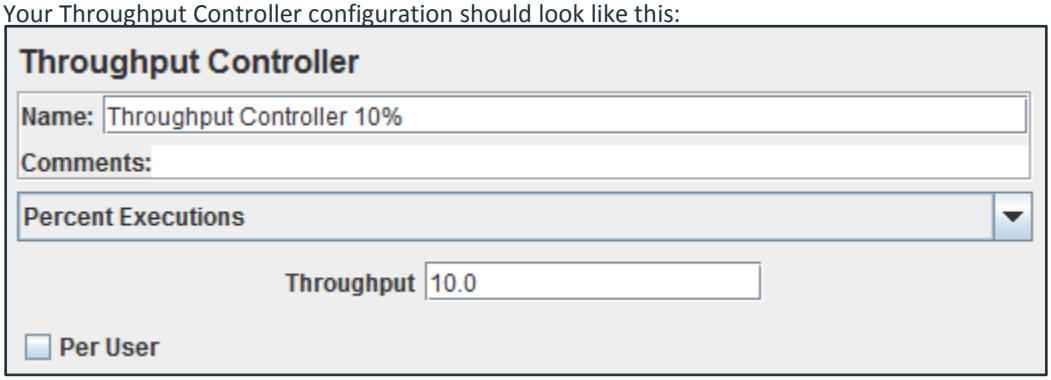
59. Click **Thread Group** to select it.

60. Right-click **Thread Group** and select **Add ▶ Logic Controller ▶ Throughput Controller**. The Throughput Controller configuration window opens.

61. Configure the Throughput Controller as follows:

Field	Value	Notes
Name	**Throughput Controller 10%**	This Throughput Controller will address the remaining 10% of your requests. Your test is now configured to address 100% of all requests through one of the four Throughput Controllers that you configured.
Selected from dropdown menu	**Percent Executions**	
Throughput	**10.0**	This value sets the throughput to 10%.

Your Throughput Controller configuration should look like this:

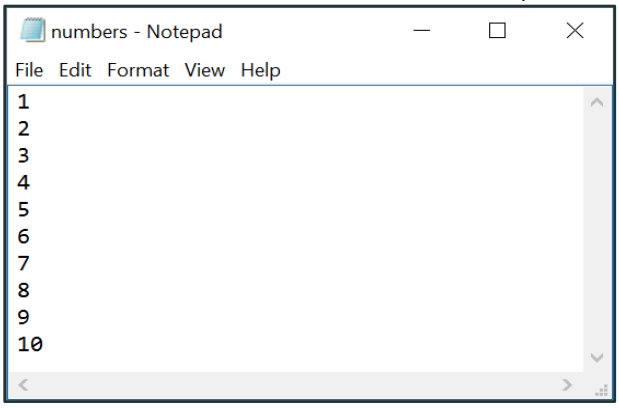


Create a CSV File for Numbers

Complete these steps to create a CSV file that will be used in by the While Controller function:

62. Open a basic word processor such as **Notepad**.

63. Enter these numbers in order with one number per row in Notepad:



64. Select **File** ▶ **Save As** and enter **numbers.csv** in the Filename field.

65. Navigate to the directory that contains the script that you are configuring for this lab, click **Save**, and close Notepad.

    Note: *You must save the colors.csv file to the same directory containing the script that you are creating for this lab to execute the script properly.*

Add a CSV Data Set Config Element

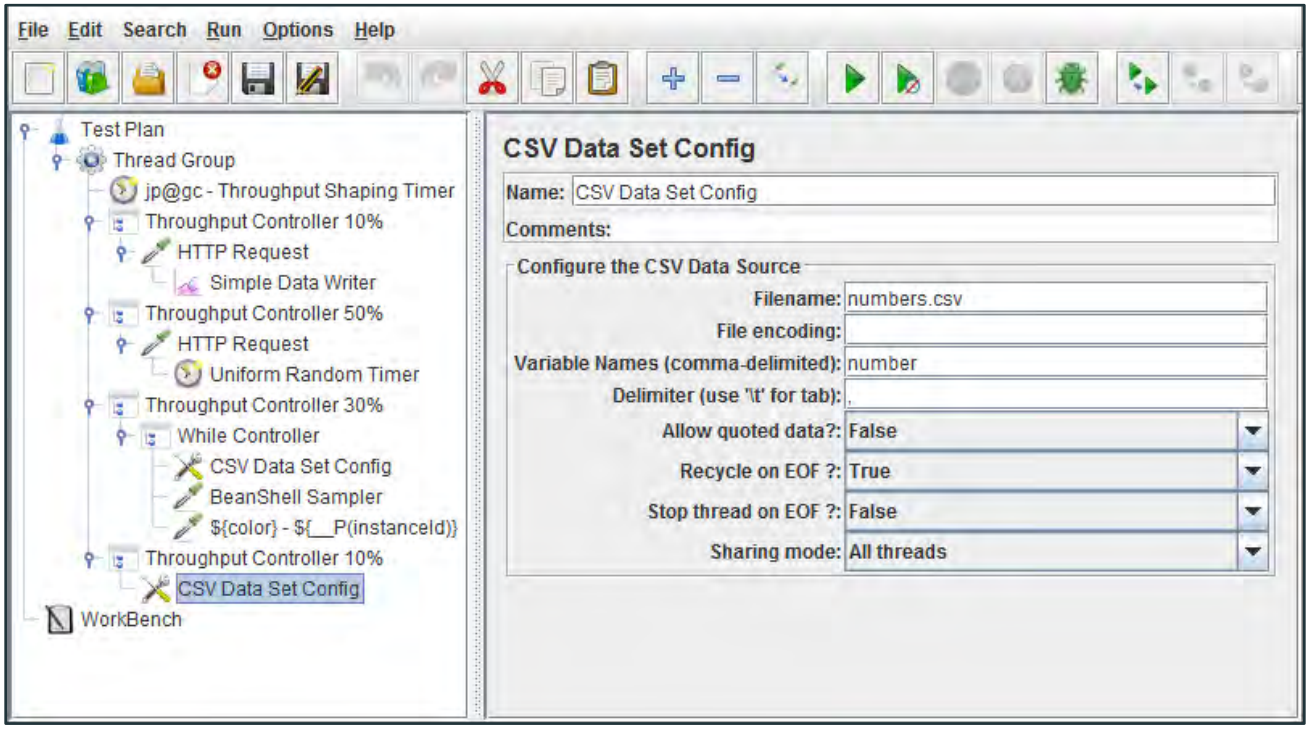
Complete these steps to add a CSV Data Set Config element to your Throughput Controller:

66. Right-click **Throughput Controller 10%** at the bottom of the Test Plan pane and select **Add** ▶ **Config Element** ▶ **CSV Data Set Config**. The CSV Data Set Config configuration pane opens.

67. Configure the CSV Data Set Config as follows:

Field	Value
Filename	**numbers.csv**
Variable Names	**number**

Your CSV Data Set Config configuration should look like this:



Add a BeanShell Sampler

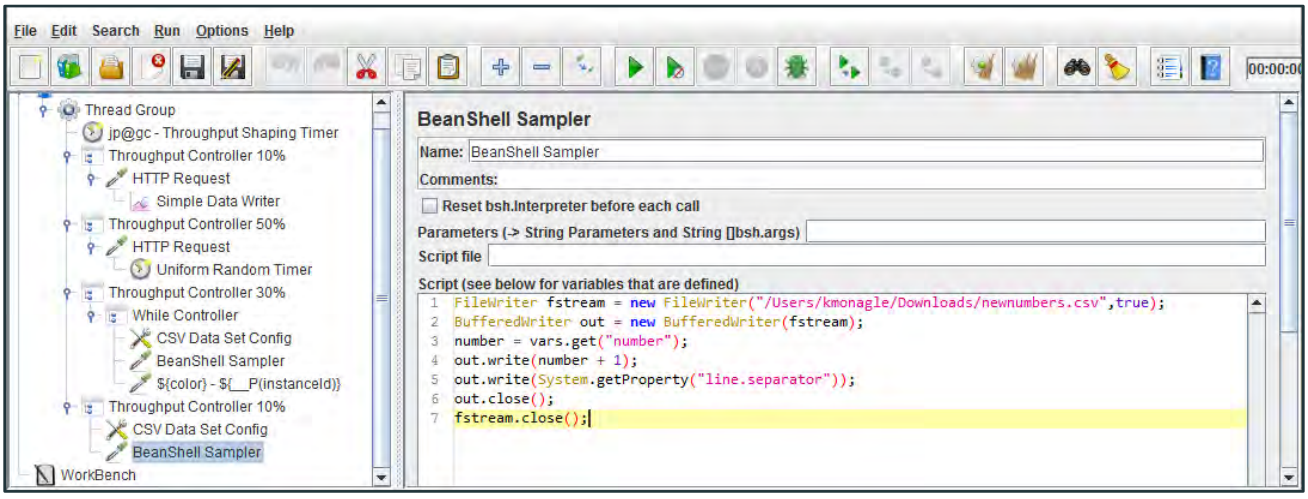
Complete these steps to add a BeanShell Sampler to your test:

68. Right-click the last **Throughput Controller 10%** in the Test Plan pane and select **Add ▶ Sampler ▶ BeanShell Sampler**.

69. Add this BeanShell code to the Script field on the BeanShell Sampler configuration pane. This script will write to a new CSV file called newnumbers.csv. It will write the number variable plus one for each loop.

Field	Value
Row 1	**FileWriter fstream = new FileWriter("/Users/*your username*/Downloads/newnumbers.csv",true);** ***Note:*** *Replace your username in the URL with your username as it appears on your local drive to force the file to download to the Downloads folder on your local drive.*
Row 2	**BufferedWriter out = new BufferedWriter(fstream);**
Row 3	**number = vars.get("number");**
Row 4	**out.write(number + 1);**
Row 5	**out.write(System.getProperty("line.separator"));**
Row 6	**out.close();**
Row 7	**fstream.close();**

Your BeanShell Sampler configuration should look like this:



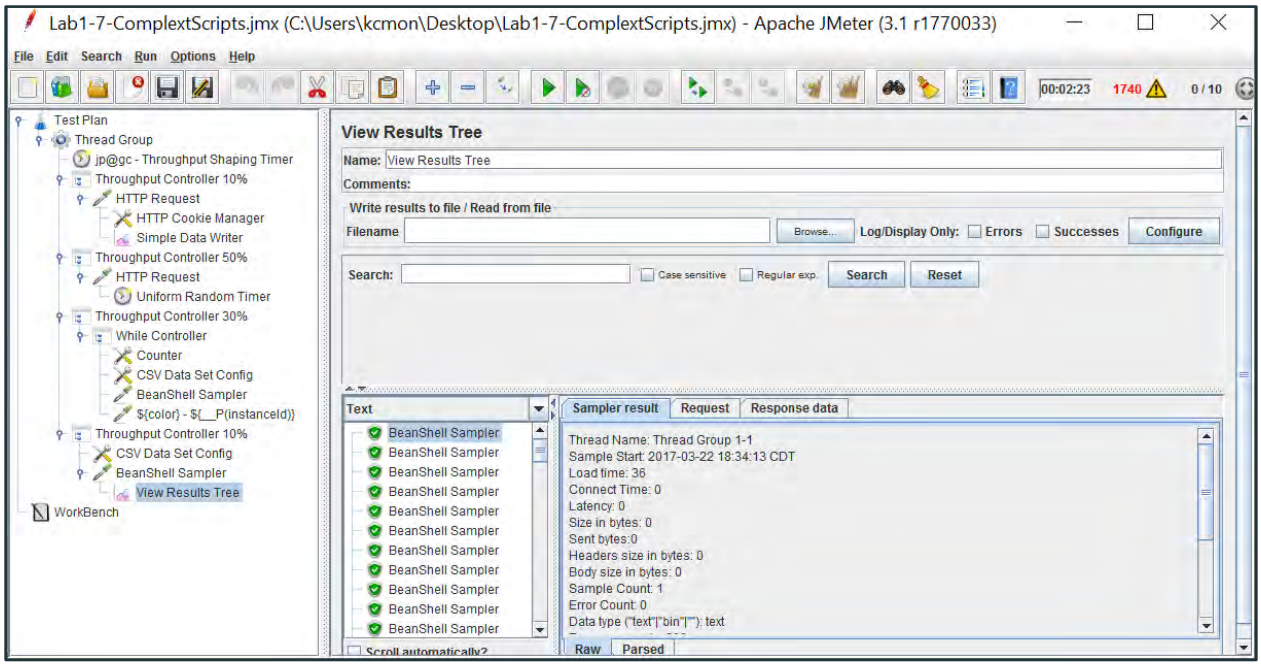
Add a View Results Tree Listener

Complete these steps to add a View Results Tree Listener to your test and then run the test and view the results:

70. Right-click **Thread Group** and select **Add ▶ Listener ▶ View Results Tree**. The View Results Tree listener configuration page opens.

71. Click **Start** to run your test.

72. Note the test results in the View Results Tree pane.



Note: *Each Throughput Controller in this lab is an independent, mini-exercise. To verify that each step was completed correctly, copy the contents of each Throughput Controller to a new JMeter script under a standard Thread Group and run it as a separate script.*