Table of Contents

[Chapter 7 1](#_Toc4852917)

[Pre-Processor 1](#_Toc4852918)

[HTML Link Parser 1](#_Toc4852919)

[HTML URL rewriting Modifier 1](#_Toc4852920)

[User Parameters 1](#_Toc4852921)

[Post-Processors 1](#_Toc4852922)

[Regular Expression Extractor 1](#_Toc4852923)

[XPath Extractor 1](#_Toc4852924)

[Result Status Handler 1](#_Toc4852925)

[Debug Post Procesor 1](#_Toc4852926)

JMeter is a open source performance testing tool, it is java based, You can do testing for websites, webservices, database.

Anatomy of a JMeter test

------------------------

Test Plan

=========

This is the root element of the JMeter scripts and houses the other components, such as threads, config elements, times, Preprocessors, Postprocessors, assertions and listener. It also offer a few configuration of its own.

Thread Groups

=============

- This is entry point of a test plan.

- They represent the number of threads/users JMeter will use to execute the test plan.

- All controllers and samplers for a test must reside under a thread group. Listener may remain under thread group or under test plan.

- Thread Group configuration provide options to specify the number of threads that will be used for the test plan, how long it will take for all threads to

become active (ramp up ) and the number of times test will be executed.

Controllers

===========

Controllers drive the processing of a test and comes in two flavors -

1. Sampler controller - sampler controller sends reqest to a server. These includes HTTP, FTP, JDBC etc.

List of sampler controllers -

1.1 HTTP request

1.2 JDBC Request

1.3 LDAP Request

1.4 SOAP/XML-RPC Requst

1.5 WebService (SOAP) reqest

1.6 FTP Request

2. Logical Controller - This allows the customization of the logic used to send the requets. Like loop controller can be used to repeat an operation a certain number of times. if controller is for selectively executing a request. Helps customzie the logic used to decide how requests are sent to server. They can modify requests, repeat reqest, interleave request, control the duration of request execution, switch request, measure the overall time taken to perform request. it is like a container which holds steps.

List of logical controller -

Test Fragments

==============

A spcial type of controller is used purely for code reuse within a test plan. They exist on the test plan tree at the same level as the thread group element and are not executed unless referenced either by any include or module controller.

Listeners

=========

Timers

======

Assertions

==========

Performance Testing

===================

performance testing is in general, a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Responsiveness - response of application

Throughput - request per second, number of unit of work in a unit of time

Reliablility - should not throw denial of service

Scalibility - should have ability to handle extra work load without impacting performance

Assertion - are required for functional testing

Listener - are helpful to view result data

Regular Expression

=================

. any number of characters

+ at least 1 character

[] - range of number / alphabet

[a-z] - any alphabet from a to z

[0-9]\* - any number of digit

<a class="theme-nav-link" href="/schedules" title="click to go to 'Schedules'" target="\_self" data-identifyelement="18">Schedules</a>

## Chapter 7

### Pre-Processor

### HTML Link Parser

### HTML URL rewriting Modifier

### User Parameters

### Post-Processors

### Regular Expression Extractor

### XPath Extractor

### Result Status Handler

### Debug Post Procesor

http://192.168.1.2:1080/WebTours/index.htm

## Chapter 8 – Timers

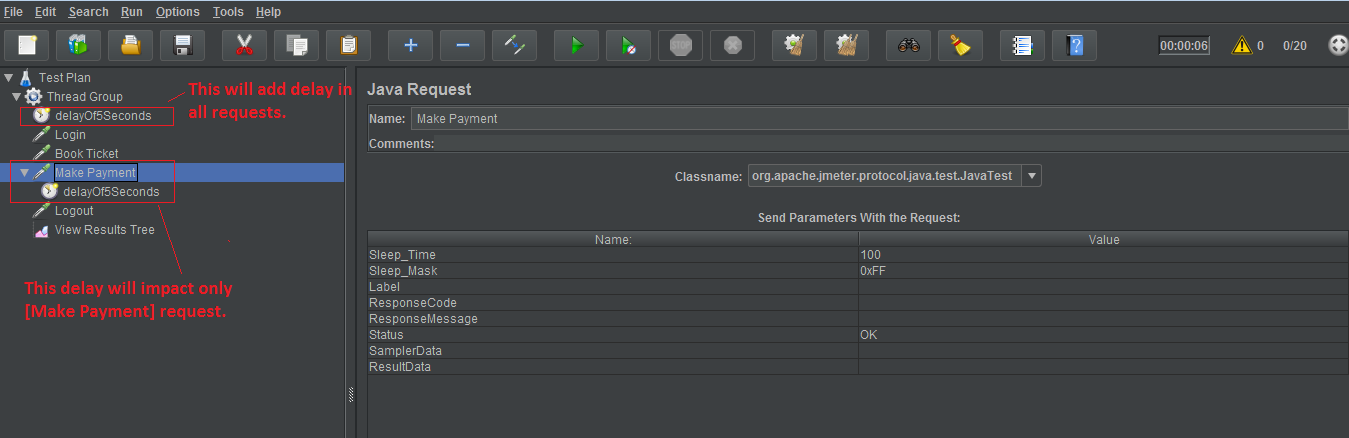
To make our script realistic we have to add some delay between two request called as think time. We can add delay in JMeter using Timer.

### Beanshell Timer

### BSF Timer

### Constant Timer

It add fix time delay between each request. Constant timer will add delay in all request in the scope. If you want to add delay in a particular request, add constant timer as a child of that request.



If you add two constant timer in same scope, sum of both timer will get added as delay.

### Gaussian Random Timer

We have 3 random timer, Gaussian, Uniform & Poisson. This timer follows distribution of time as like bell curve.

### Uniform Random Timer

### Poisson Random Timer

### Synchronizing Timer

This timer blocks the thread until a given number is reached. Then all threads are released at once. This helps in creating large number of threads on server.

In case value zero is passed into timer, by default it will take 100 threads. It is mandatory to provide value else JMeter shows error.

Helps in simulating peak load on server.

We can use random() function to create random number of thread for each request.

### Constant Throughput Timer

It adds variable pauses to calculate a possible given number.

### Custom Timer – gets added from google plugin

## Chapter 20 – JMeter Inbuilt Functions

Function syntax in JMeter ${\_\_funcName(v1,v2,v3)}

**${\_\_time(YMDHMS)}** – Append this function in request name, this will add timestamp when request was executed.

**${\_\_Random(2000,5000,refVar)} –** to set random value in constant timer. Add this function in **constant timer -> Thread delay (in milliseconds).** refVar variable contains value generated by this function

## Pointers

https://github.com/uttesh/JmeterAPISample/tree/master/lib

https://github.com/sayems/JMeter

http://uttesh.blogspot.in/2015/04/jmeter-load-testing-by-code-jmeter-api.html

http://www.agileload.com

https://www.blazemeter.com

https://www.blazemeter.com/blog/how-analyze-results-load-test-using-blazemeter-0

http://www.performancetestingfun.com/bottlenecks/

https://www.soasta.com/blog/

http://www.radview.com/

https://en.wikipedia.org/wiki/Arithmetic\_mean

https://en.wikipedia.org/wiki/Skewed\_distribution

https://en.wikipedia.org/wiki/Median

http://www.skillsyouneed.com/num/averages.html

**Rajendra Singh Some more links...**

http://www.quotium.com/performance/90-percentile-response-time/

http://www.loadtestingtool.com/blog/wapt-usage/how-to-analyze-load-test-report-response-time/

http://www.performancetestingfun.com/bottleneckanalysis/

http://sqa.stackexchange.com/questions/10090/load-testing-result-analysis-please-help-understand-a-few-keywords

https://www.thoughtworks.com/insights/blog/performance-testing-nutshell

http://topobaseinsiders.typepad.com/the\_topobase\_insiders/2007/09/oracle-database.html

**Open source performance monitoring tools**

\* http://www.slideshare.net/adrianco

\* http://www.strangeloopnetworks.com/blog/

\* http://www.perfdynamics.com/

\* http://www.igvita.com/

\* http://highscalability.com/

\* http://velocityconf.com/

\* http://blogs.impetus.com/test\_engineering/performance\_engineering/J2EEPerformanceTestingAndBottleneckIdentification.do

http://www.soasta.com/products/mpulse/

<https://www.blazemeter.com/blog/how-monitor-your-server-health-performance-during-jmeter-load-test>

## Chapter 1 | Installation, ENV Configuration, JMeter Test Recording Mode

### Download JMeter

### Installing JDK

### Setting Java environment

### Running JMeter

### JMeter Classpath

### Configuring the proxy server

### Running in Non-GUI Mode

### Running in Server mode

### Recording Script using HTTP Test Recorder

### Recording script using Chrome Brower Extension

### Converting HTTP Web Archives (HAR) to JMeter test plan

### Executing JMeter test

### Analyzing Test Result

## Chapter 2 | Anatomy of a JMeter Test

### Test Plan

#### ThreadGroup

#### Controllers

#### Sampler

#### Logic controller

### Test Fragments

#### Listeners

#### Timers

#### Assertions

### Configuration Elements

#### Preprocessor

#### Postprocessor

## Chapter 3 | Working on Forms

## Chapter 4 | Managing Sessions

### Managing session with cookies

### Managing session with URL Rewriting

## Chapter 5 | Handling Responses

### Regular Expression Extractor

### Regular Expression Tester

CSS/JQuery Extractor

XPath Extractor

XML Extractor

JSON Extractor

HTML Extractor

Response Assertion

Duration Assertion

Uploading files with your script

## Performance Testing Strategy

* Determine the user load that will use the application during the peak time.
* We should gather these requirements from the customer, product owner or the business analysts.
* We should have scenario where we will have a set of virtual users who will cover the maximum load and the average load.
* We should design scenarios where we hit the system with a maximum load of users once and also a scenarios where will hit the system slowly as a group.
* We should also have a baseline established when we are going test against the newly added features.
* We should have a procedure to run a test against existing code (baseline) and then run the same test against the newly added code where we are able to measure against.
* performance test environment which is similar to the production environment. Otherwise we would not yield correct performance test results. The hardware configurations, database servers and the user load should be similar to the production site.
* We should perform the performance test near to the area where the application being deployed. If we do the performance test remotely the network latency will affect our performance test results.
* As a server side performance testing. We should note that we should performance test the web services and it the load to the web services and see whether they will work with the maximum load specified.
* We should also have access to the applications server hardware as we may have to obtain the information of server's processor, memory, databases and also OS processes that are being run, when we are doing performance testing. We should keep a close eye on these parameters as the utilization rate will increase ones there is high load of users.
* From database side, We should look at the utilization of stored procedures, how many stored procedures are executed and whether there are unwanted database calls created. This DB calls are also an underlying reason for performance issues in many of the web solutions.
* We should also be mindful to create end to end performance scenarios rather testing each feature separately, as testing end to end gives more realistic view.
* Tools - JMeter, Neoload