Jmeter Performance Testing

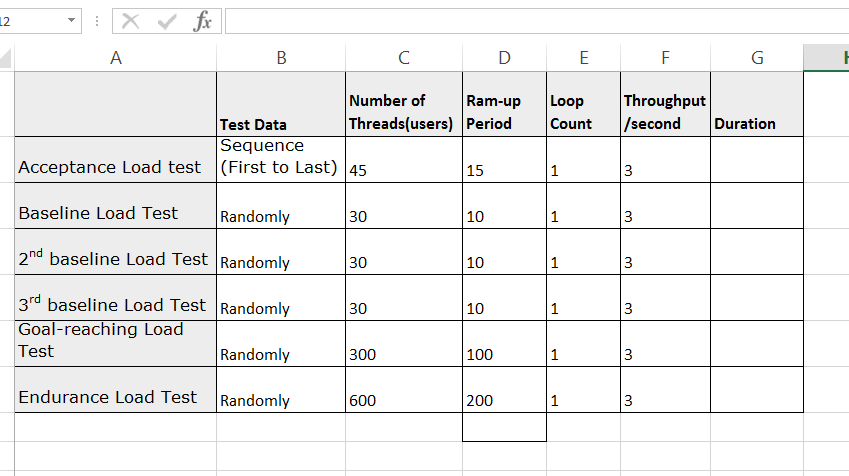
Winter 2016

Contents

[1. Week 6: Throughput, Function Helper, Saving Test Result 4](#_Toc467782966)

[2. Contact Us 9](#_Toc467782967)

# Week 6: Throughput, Function Helper, Saving Test Result



* What Is Throughput?
  + Throughput or TPS (Transaction per second) is to measure number of requests sent to webs.
  + Throughput is to measure as total no. of transaction or requests in a given time or TPS (transaction per second). It in a way reflects the capacity of the server. The ability of the server in terms of how much load it can take. It is one of the significant indicator that helps in evaluating the performance of application.
  + For example: for a public web application such as kijiji, 401dixiehyundai we need to test for the performance goal how many request per second - In peak hour, throughput goal of kijiji is 1000 requrests/second; thoughput goal of 401dixiehyundai is 40 requests/second. In regular hour, thoughtput goal of kijiji is 500 requests/second, throughput goal of 401dixiehyundai is 15 request/second. With this goal, we need to set up Jmeter test plan so that Jmeter will send 1000 requests/second to kijiji.com or send 20 requests/second to 401dixiehyundai.com in 1 hour and the expected result is the server still working fine and is not down or get any error. Also, the response time is still in the acceptable time.
* Throughput in JMeter
  + One of the performance indicator we often verify is the ‘throughput’. We measure throughput by observing the test results. Test results are observed through listener components. The following listener components will allow you to measure or track throughput value:
    - Summary Report
    - Aggregate Graph
    - Aggregate Report
    - Graph Results
* How To Set Up Throughput Goal
  + Constant Throughput Timer: this timer will allow you to specify a constant throughput value. It will maintain that constant throughput during the test. Often you will see throughput coming close to the specified value when running the tests that lasts for longer duration. Apart from specifying the constant throughput value, it also allows you to select the mode of throughput calculation.
  + Throughput Shaping Timer : in some load testing scenarios, you need to know how many requests per second your application under load can handle (within acceptable response times) with different target request per second, Attempting to determine it by tinkering with ramp-up and think times can be time consuming and not always accurate. Throughput Shaping Timer is to configure the ramp-up period to meet different target throughput.
    - Increase the load from 1 to 3 RPS over 20 seconds (ramp-up)
    - Hold the load at 3 RPS for 40 seconds (hold-for)
* List of Function Helper: Function helper dialog is a list of all functions in Jmeter to return a value based on the input variables.
  + The Function Helper Dialog is available from JMeter's Options tab. Using the Function Helper, you can select a function from the pull down, and assign values for its arguments. The left column in the table provides a brief description of the argument, and the right column is where you write the value for that argument. Different functions take different arguments. Once you have done this, click the “Generate" button, and the appropriate string is generated, which you can copy-paste into the test plan wherever you need to
  + A function call looks like this:
* ${\_\_functionName(var1,var2,var3)}
* Where "\_\_functionName" matches the name of a function.
* var1, var2, var3 is variable required or optional of the function.
* Following table lists a group of functions loosely grouped into types:

|  |  |  |
| --- | --- | --- |
| **Function Type** | **Name** | **Comment** |
| Information | threadNum | Get thread number. |
| Information | samplerName | Get the sampler name (label). |
| Information | machineIP | Get the local machine IP address. |
| Information | machineName | Get the local machine name. |
| Information | time | Return current time in various formats. |
| Information | log | Log (or display) a message (and return the value). |
| Information | logn | Log (or display) a message (empty return value). |
| Input | StringFromFile | Read a line from a file. |
| Input | FileToString | Read an entire file. |
| Input | CSVRead | Read from CSV delimited file. |
| Input | XPath | Use an XPath expression to read from a file. |
| Calculation | counter | Generate an incrementing number. |
| Calculation | intSum | Add int numbers. |
| Calculation | longSum | Add long numbers. |
| Calculation | Random | Generate a random number. |
| Calculation | RandomString | Generate a random string. |
| Calculation | UUID | Generate a random type 4 UUID. |
| Scripting | BeanShell | Run a BeanShell script. |
| Scripting | javaScript | Process JavaScript (Mozilla Rhino). |
| Scripting | jexl, jexl2 | Evaluate a Commons Jexl expression. |
| Properties | property | Read a property. |
| Properties | P | Read a property (shorthand method). |
| Properties | setProperty | Set a JMeter property. |
| Variables | split | Split a string into variables. |
| Variables | V | Evaluate a variable name. |
| Variables | eval | Evaluate a variable expression. |
| Variables | evalVar | Evaluate an expression stored in a variable. |
| String | regexFunction | Parse previous response using a regular expression. |
| String | escapeOroRegexpChars | Quote meta chars used by ORO regular expression. |
| String | char | Generate Unicode char values from a list of numbers. |
| String | unescape | Process strings containing Java escapes (e.g. \n & \t). |
| String | unescapeHtml | Decode HTML-encoded strings. |
| String | escapeHtml | Encode strings using HTML encoding. |
| String | TestPlanName | Return name of current test plan. |

* Save Responses to a file: this test element will save the Reponse of each sampler into the file. The primary use for this is in creating functional tests, but it can also be useful where the response is too large to be displayed in the View Results Tree Listener.
* Using Simple Data Writer To Save Test Result:
  + This listener is to record results to a file. It is meant to provide an efficient means of recording data by eliminating GUI overhead. When running in non-GUI mode, the -l flag can be used to create a data file. The fields to save are defined by JMeter properties.
  + Jmeter properties: Results file configuration
* jmeter.save.saveservice.output\_format=csv #format of file (jmx or csv or xml)
* jmeter.save.saveservice.assertion\_results\_failure\_message=false #save assertion message if it is failed.
* jmeter.save.saveservice.assertion\_results=none #save assertion
* jmeter.save.saveservice.data\_type=true #save data type
* jmeter.save.saveservice.label=true #save label of sampler
* jmeter.save.saveservice.response\_code=true #save response code
* jmeter.save.saveservice.response\_data=false #save response data
* jmeter.save.saveservice.response\_data.on\_error=false #save response data if it is failed.
* jmeter.save.saveservice.response\_message=true #save response message
* jmeter.save.saveservice.successful=true #save if it is passed
* jmeter.save.saveservice.thread\_name=true #save ThreadGroup name
* jmeter.save.saveservice.time=true #save time
* jmeter.save.saveservice.subresults=true #save sub-result
* jmeter.save.saveservice.assertions=true #save assertion setup
* jmeter.save.saveservice.latency=true #save latency value
* jmeter.save.saveservice.samplerData=false #save Data in sampler
* jmeter.save.saveservice.responseHeaders=false #save Response header
* jmeter.save.saveservice.requestHeaders=false #save Request header
* jmeter.save.saveservice.encoding=false #save encoding
* jmeter.save.saveservice.bytes=true #save data bytes
* jmeter.save.saveservice.url=false #save URL of request
* jmeter.save.saveservice.filename=false #save filename
* jmeter.save.saveservice.hostname=false #save hostname of request
* jmeter.save.saveservice.thread\_counts=false #save number of thread group
* jmeter.save.saveservice.sample\_count=false #save number of sample
* jmeter.save.saveservice.idle\_time=false #save idle time
* jmeter.save.saveservice.timestamp\_format=ms # set up format time
* jmeter.save.saveservice.timestamp\_format=yyyy/MM/dd HH:mm:ss.SSS
* jmeter.save.saveservice.default\_delimiter=, #default delimiter of csv file
* jmeter.save.saveservice.default\_delimiter=\t #default delimiter of csv file
* jmeter.save.saveservice.print\_field\_names=false #save field names
* jmeter.save.saveservice.xml\_pi=<?xml-stylesheet type="text/xsl" href="../extras/jmeter-results-detail-report\_21.xsl"?>
* jmeter.save.saveservice.base\_prefix=~/
* Some useful Listeners:
  + Aggregate Report: The aggregate report creates a table row for each differently named request in your test. For each request, it totals the response information and provides request count, min, max, average, error rate, approximate throughput (request/second) and Kilobytes per second throughput.
  + Aggregate Graph: The aggregate graph is similar to the aggregate report. The primary difference is the aggregate graph provides an easy way to generate bar graphs and save the graph as a PNG file.
  + Assertion Results: The Assertion Results visualizer shows the Label of each sample taken. It also reports failures of any Assertions that are part of the test plan.
  + Graph Results: The Graph Results listener generates a simple graph that plots all sample times. Along the bottom of the graph, the current sample (black), the current average of all samples (blue), the current standard deviation (red), and the current throughput rate (green) are displayed in milliseconds.
  + Monitor Results: Monitor Results is a new Visualizer for displaying server status. There are two primary tabs for the monitor. The first is the "Health" tab, which will show the status of one or more servers. The second tab labled "Performance" shows the performance for one server for the last 1000 samples. The equations used for the load calculation is included in the Visualizer.
  + Response Time Graph: The Response Time Graph draws a line chart showing the evolution of response time during the test, for each labelled request. If many samples exist for the same timestamp, the mean value is displayed.