Performance Testing

Lesson 3: Introduction to JMETER

Lesson Objectives

- To understand the following topics:
  - Introduction to Performance Testing Tools
  - Apache JMeter
  - Features of JMeter
  - The GUI
  - Jmeter Installation steps
  - Components in Jmeter GUI
  - JMeter Elements – Workbench, Test Plan, Thread Groups, Controllers, Samplers, Logic controllers, Listeners, Timers, Assertions
  - Summary

Introduction to Performance Testing Tools

- LoadNinja – It is cloud-based load testing tool that empowers teams to record & instantly playback comprehensive load tests & run these load tests in real browsers at scale.
- HP LoadRunner - Interactive tool for testing the performance of applications by running thousands of Virtual Users that are distributed over a network
- QALoad – It records the user interactions with the application using the Script Development Workbench
- eLoad – It allows multiple users to access e-Load from any networked machine using a web browser
- NeoLoad - It is the performance testing platform designed for DevOps that seamlessly integrates into your existing Continuous Delivery pipeline
- Apache JMeter - It is one of the leading tools used for load testing of web and application servers.

Apache JMeter

- It is one of the leading tools used for load testing of web and application servers.
- Apache JMeter is Opensource and 100% pure Java desktop application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions
- Apache JMeter may be used to test performance both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries, FTP Servers and more)
- It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types
- You can use it to make a graphical analysis of performance or to test your server/script/object behavior under heavy concurrent load

Features of JMeter

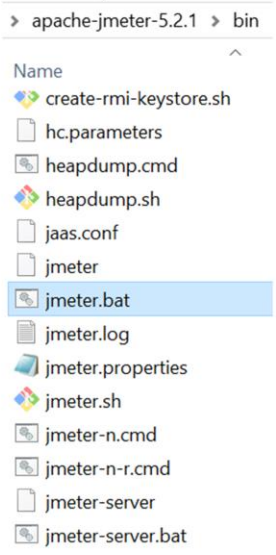
- Can load and performance test many different server types:
  - Web - HTTP, HTTPS
  - SOAP
  - Database via JDBC
  - LDAP
  - JMS
  - Mail - POP3
- Complete portability and 100% Java purity
- Full Swing and lightweight component support (precompiled JAR uses packages javax.swing.*)
- Full multithreading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups

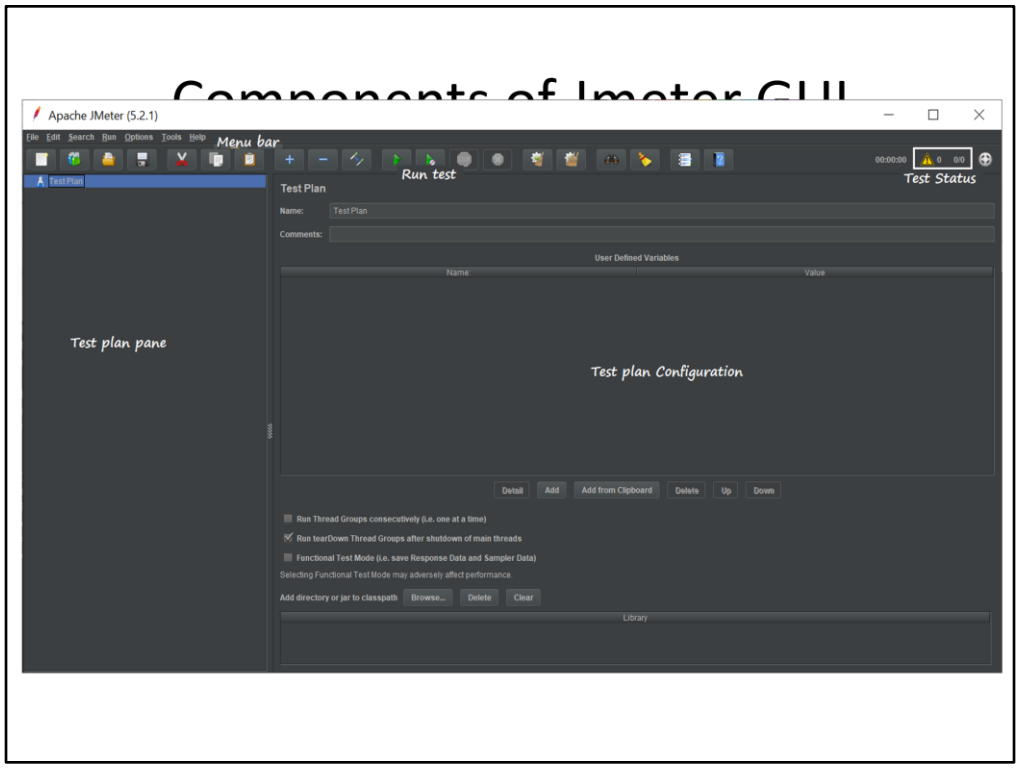
Features of JMeter (Cont..)

- Caching and offline analysis/replaying of test results
- Highly Extensible:
  - Pluggable Samplers allow unlimited testing capabilities
  - Several load statistics may be chosen with pluggable timers
  - Data analysis and visualization plug-ins allow great extendibility as well as personalization
  - Functions can be used to provide dynamic input to a test or provide data manipulation
  - Scriptable Samplers (BeanShell is fully supported; and there is a sampler which supports BSF-compatible languages)

JMeter Installation Steps

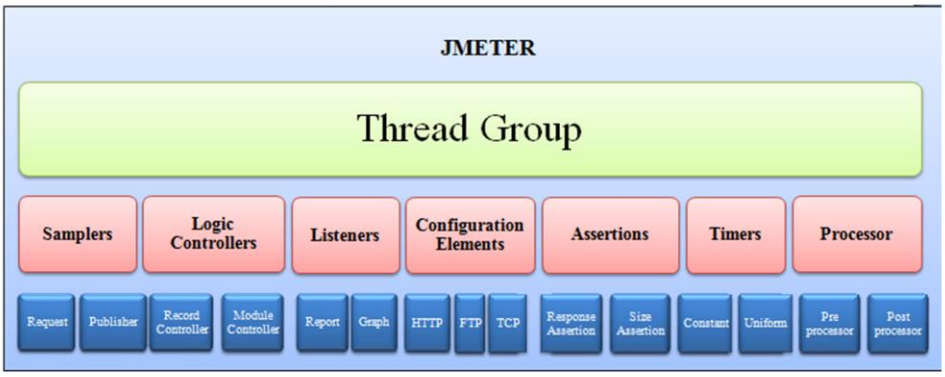
- Check whether JDK installed : cmd type java –version
- Google ….. Download Jmeter …. http://jmeter.apache.org/download_jmeter.cgi
- Download the binary Zip file. Unzip the file and keep it at any location of your choice.
- To Start Jmeter: Visit the downloaded folder jmeter /bin -> jmeter.bat
- This will open the JMeter window




Components of Jmeter GUI
Apache JMeter (5.2.1)
File Edit Search Run Options Tools Help
Menu bar
Run test
Test Status
Test Plan
Name:
Test Plan
Comments:
User Defined Variables
Name:
Value
Test plan pane
Test plan Configuration
Detail
Add
Add from Clipboard
Delete
Up
Down
Run Thread Groups consecutively (i.e. one at a time)
Run tearDown Thread Groups after shutdown of main threads
Functional Test Mode (i.e. save Response Data and Sampler Data)
Selecting Functional Test Mode may adversely affect performance.
Add directory or jar to classpath
Browse...
Delete
Clear
Library

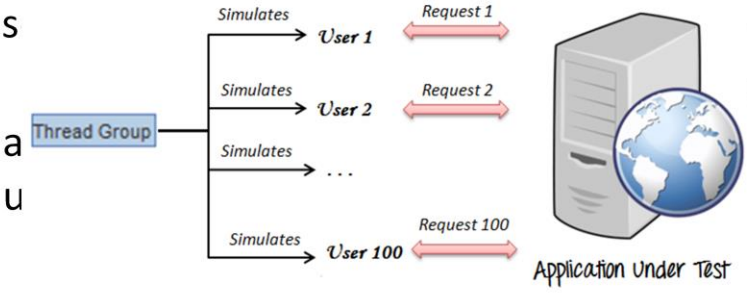
JMeter Elements

- Different components of JMeter are called



JMeter Element – Thread Group

- Thread Groups is a collection of Threads.
- Each thread simulates one real user using the AUT.
- The controls for a thread group allow you to s .
- eads a 100 u



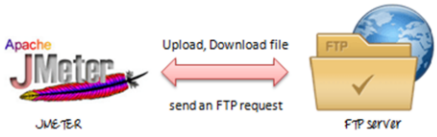
JMeter Element – Samplers

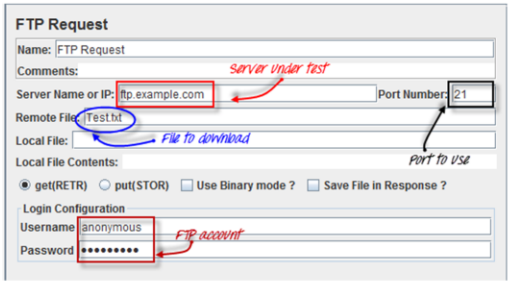
- Every individual user(Thread) might give any type of request (HTTP, FTP , JDBC, etc. protocols)
- In such a case, how does a Thread Group know which type of requests it needs to make?
  - This is possible using Samplers

Example : FTP Request

If you want to performance test an FTP server, use an FTP request sampler in JMeter which lets you send either an FTP "download file" or "upload file" request to an FTP server.

If you want to download a file "Test.txt" from an FTP server under test, you need to configure some parameters in JMeter as in the figure.

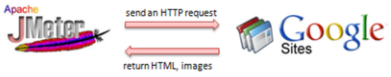




JMeter Element – Samplers

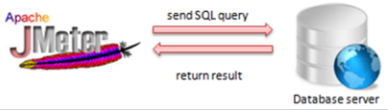
Example : HTTP Request

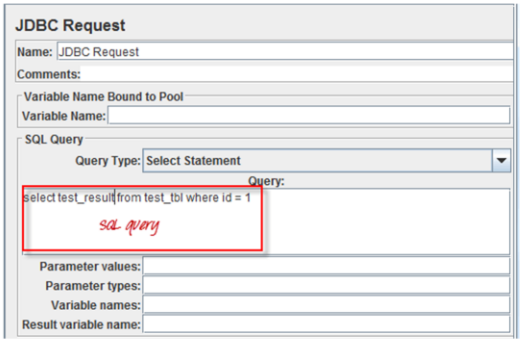
If you want to performance test an HTTP server, use an HTTP/HTTPS request sampler in JMeter which lets you retrieve HTML files or image from this website.



Example : JDBC Request

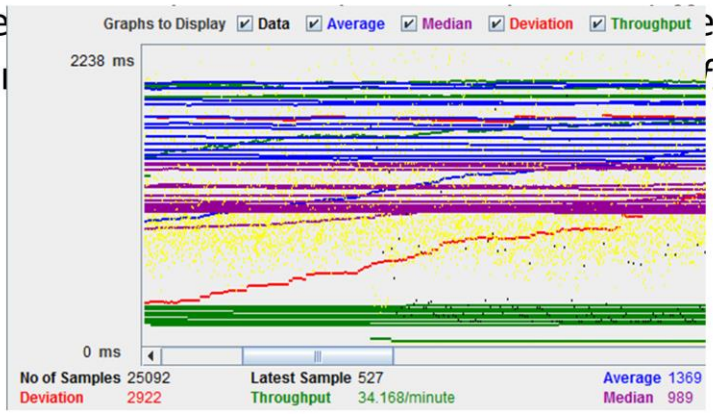
Consider a database server has a field test_result stored in a table name test_tbl. You want to query this data from the database server; you can configure JMeter to send a SQL query to this server to retrieve data.





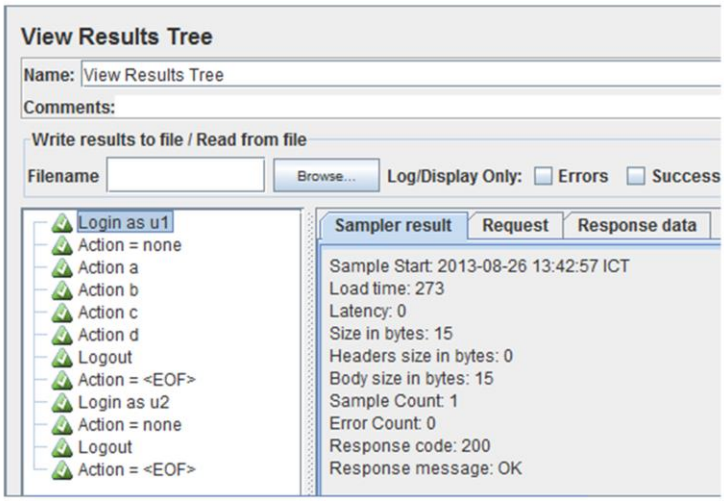
JMeter Element – Listeners

- Listeners show the results of the test exe erent for file



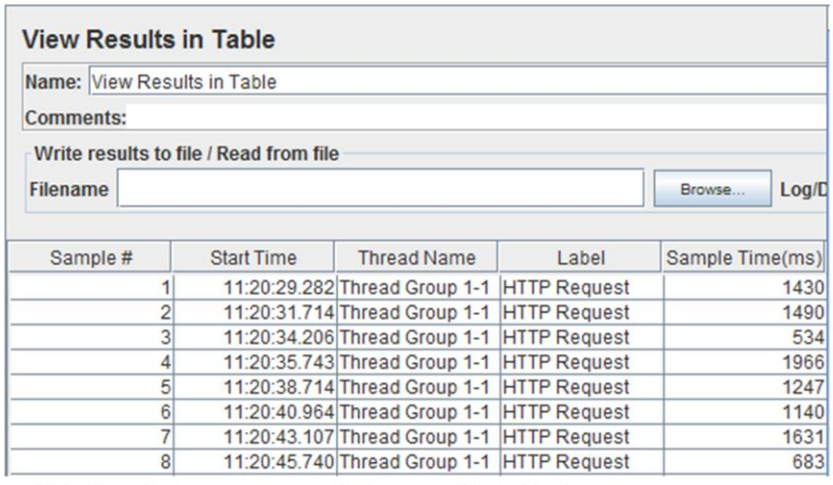
Graph result listeners display the server response times on a Graph

JMeter Element – Listeners



View Result Tree show results of the user request in basic HTML format

JMeter Element – Listeners



Sample #	Start Time	Thread Name	Label	Sample Time(ms)
1	11:20:29.282	Thread Group 1-1	HTTP Request	1430
2	11:20:31.714	Thread Group 1-1	HTTP Request	1490
3	11:20:34.206	Thread Group 1-1	HTTP Request	534
4	11:20:35.743	Thread Group 1-1	HTTP Request	1966
5	11:20:38.714	Thread Group 1-1	HTTP Request	1247
6	11:20:40.964	Thread Group 1-1	HTTP Request	1140
7	11:20:43.107	Thread Group 1-1	HTTP Request	1631
8	11:20:45.740	Thread Group 1-1	HTTP Request	683

Table Result show summary of a test result in table format

JMeter Element – Listeners

sampler_label	aggregate_	average	aggregate_
/v6exp3/redir.html	187	3517	185
/v6exp3/iframe.htm	168	1595	165
/first-android-testin	94	6009	1581
/search/adi/g.php	101	2999	21
/quality-center-tuto	67	3292	146
/b	41	3220	119
/bn/at_300.html	12	2174	1108
/getting-started-wit	8	2115	872
/sql.html	1	908	908
TOTAL	679	3225	21

Log show summary of a test results in the text file

JMeter Element – Configuration

- Configuration allows to set up defaults and variables for later use by samplers.

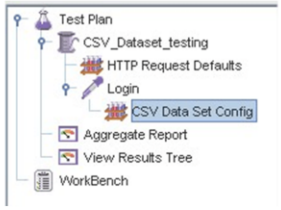
Example : CSV Data Set Config

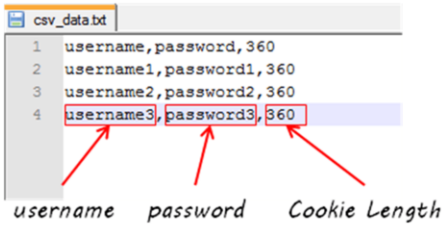
If you want to test a website for 100 users signing-in with different credentials, you do not need to record the script 100 times; you can parameterize the script to enter different login credentials.

This login information (e.g. Username, password) can be stored in a text file.

JMeter has an element that allows you to read different parameters from that text file.

It is "CSV Data Set Config", which is used to read lines from a file and split them into variables.



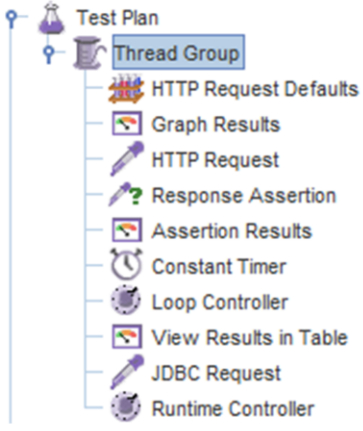


Configuration Elements :

1. CSV Data Set Config : Used to simulate multiple user login; Suitable for large numbers of parameters
2. HTTP Cookie Manager
3. Login Config Element : Used to simulate one user login; Suitable for login parameter only (user and password)
4. HTTP Request Defaults
5. FTP Request Defaults

JMeter Test Plan

- Test Plan is where you add elements required for your JMeter Test.
- It stores all the elements (like ThreadGroup, Timers etc) and their corresponding settings required to run your desired Tests.



JMeter WorkBench

- The WorkBench simply provides a place to store test elements **temporarily**.
- WorkBench has no relation with Test Plan. JMeter will **not save** the contents of the WorkBench. It only saves the contents of the Test Plan branch

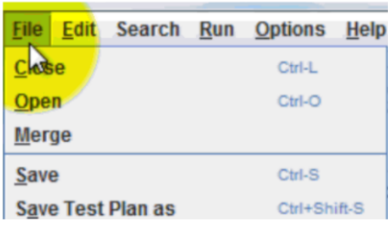
Adding Elements to Test Plan

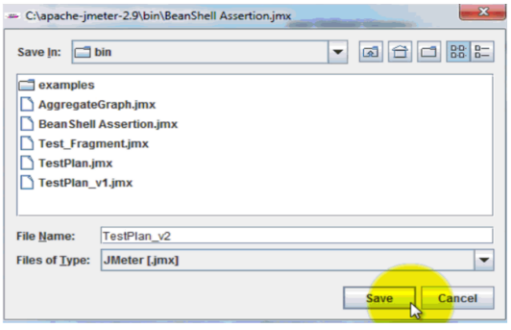
- Adding Elements is the **essential** step to build a Test Plan because without adding elements, JMeter **cannot** execute your Test Plan
- A Test Plan includes many Elements such as **Listener, Controller, and Timer**
  - You can add an element to test plan by right-clicking on a **Test Plan** and choose new elements from "**Add**" list.
  - Suppose, you want to add 2 elements to Test Plan **BeanShell Assertion** and **Java Request Default**
  - Right click **Test Plan -> Add -> Assertion-> Bean Shell Assertion**
  - Right click **Test Plan -> Add -> Config Element -> Java Request Default**

Remove an unused element :
Let's say, you want to remove element "**HTTP Request Defaults**", select "HTTP Request Default" -> Right click-> choose **Remove** from the context menu -> Click **Yes** to confirm delete this element on message box

Save a Test Plan

- Before running a test, you should save your Test Plan first. Saving your Test Plan helps you avoid unexpected error when running the test plan. Steps to saving Test plan -
  - ***File -> Save Test Plan as-> a Dialog box display***
  - *Enter a filename of Test Plan* ***->click Save***





Saving a Test Plan is **different** from saving elements.
Saving a Test Plan
Test Plan consists of one or many elements.
When you save your Test Plan, all those elements in the plan are saved.
Saving an Element
Element is a basic component of Jmeter.
When you save your elements, only one element is saved..

Run a Test Plan

- To run your single or multiple test plans, choose **Start** (Control + R) from the **Run** menu item.

When JMeter is running, it shows a small green box at the right-hand end of the menu bar.

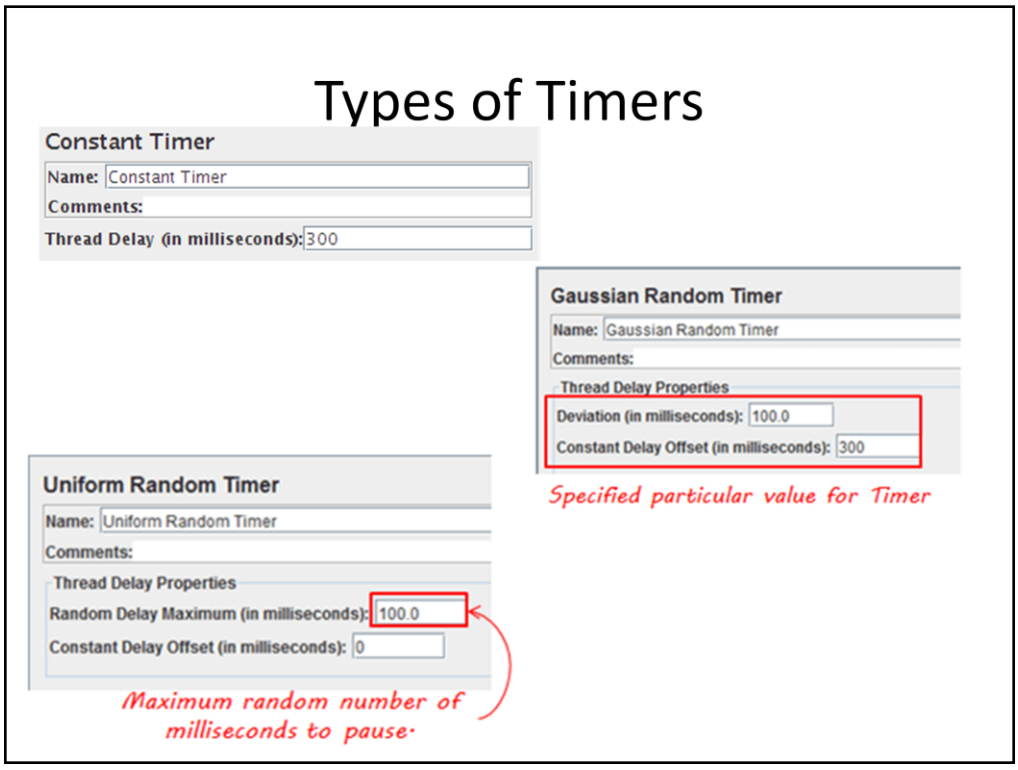
The numbers to the left of the green box are the number of **active threads** / **total number** of threads.
To Stop the Test, press **Stop** button or use short key Ctrl + '.'

Test Report
When test execution is done, you can get the test report. The test report includes the error log file, which is saved in jmeter.log, and the test results summary.

Jmeter- Timers

- By default, JMeter sends the request **without pausing** between each request due to which the test server gets over loaded.
- For example, assume you are sending thousands of requests to a web server under test in a few seconds. This is what happens!
- Timers allow JMeter to **delay** between each request thus, solving the server **overload** problem.
- Types of Timers :
  - Constant Timer
  - Gaussian Random Timer
  - Uniform Random Timer
  - BeanShell Timer
  - JSR223(ScRipting for Java Platform) Timer

In real life visitors do not arrive at a website all at the same time, but at different time intervals. So Timer will help mimic the real-time behavior.



- Constant Timer delays each user request for the **same** amount of time.
- Gaussian Random Timer delays each user request for a **random** amount of time i.e. Deviation time + Constant delay offset = Total delay
- Uniform Random Timer delays each user request for a random amount of time i.e. Random delay max. + Constant delay offset = Total delay.
- BeanShell Timer can be used to **generate** a delay time between each user request.
- JSR223 Timer can be used to generate a delay between each user request using a JSR223 scripting language

JMeter- Assertions

- Assertion help verifies that your server under test returns the **expected** results.
- Types of Assertions :
  - Response Assertion
  - Duration Assertion
  - Size Assertion
  - XML Assertion
  - HTML Assertion

- Response Assertion lets you add pattern strings to be compared against various fields of the server response.
  For example, you send a user request to the website http://www.google.com and get the server response. You can use Response Assertion to verify if the server response **contains** expected pattern string (e.g. "OK").
- Duration Assertion tests that each server response was received within a **given amount** of time. Any response that takes longer than the given number of milliseconds (specified by the user) is marked as a failed response.
  For example, a user request is sent to www.google.com by JMeter and get a response within **expected** time 5 ms then test case pass, else, test case failed.
- Size Assertion tests that each server response contains the expected number of byte in it. You can specify that the size be equal to, greater than, less than, or not equal to a given number of bytes.
  JMeter sends a user request to www.google.com and gets response packet with size less than **expected** byte 5000 bytes a test case pass. If else, test case failed.
- XML Assertion tests that the response data consists of a formally correct XML document.
- HTML Assertion allows the user to check the HTML syntax of the response data. It means the response data must be met the HTML syntax.

Jmeter- Controllers

- Logic Controllers let you define the order of processing request in a Thread. It lets you control "when" to send a user request to a web server.
- Few common types of Controllers :
  - Recording Controller is a placeholder to store these recording steps.
  - Simple Controller is just a container for user requests.
  - Loop Controller makes the user request run **a specified number of times** or run **forever**
  - Random Controller makes all the user requests run in **the random** order in each loop period.
  - Module Controller adds modularity to Jmeter.
  - Transaction Controller measures the **overall time** taken to **finish** a test execution

Recording Controller :
Jmeter records the user activity and stores in Recordign controller.

Random Controller
For example, you have 3 user requests to website http://www.google.com in following order:
1. HTTP request
2. FTP request
3. JDBC request

These 3 requests should run 5 times. Total 15 user requests will be sent to Google server by JMeter.
In *sequential order,* requests are sent sequentially in following order :
HTTP request ->FTP request->JDBC request for each loop.
In random order, requests are sent as randomly,
FTP request ->HTTP request->JDBC request
(or)
JDBC request ->FTP request->HTTP request for each loop.

Module Controller
Web applications consist of small units of functionality (i.e. Logon, Create Account, Logoff...). This functionality can be stored in Simple Controller as "modules". Module Controller will choose which module needs to run.
For example, consider that Simple Controller has 3 modules – Login, Search and Logout – the Module controller wil ldecide which module should run.

JMeter- Best Practices

- Limit the Number of Threads : The **maximum** number of threads you can effectively run with JMeter is **300**.
- Use a proxy server will enable useful features to record your testing.
- Use variables to use different values for different users/threads
- Reduce resource requirement to control the heavy load on computer memory.
- Check the JMeter logs to find the error early
- Erase the existing local path from CSV Data Set Config so that JMeter can find the CSV data file on your local PC.
- Follow file naming convention

Summary

- In this lesson, you have learnt:
  - Different tools used in Performance Testing
  - Introduction to Apache Jmeter GUI
  - Features of Jmeter
  - JMeter Installation steps
  - Components in Jmeter GUI
  - JMeter Elements