**JMETER**

To launch JMeter, launch "C:\Users\VJ\Documents\apache-jmeter-5.3\bin\ApacheJMeter.jar"

As JMeter is a java application, it is independent of platform and operating system. So it does not matter you are using windows or mac system, it will work exactly same manner on both the platforms.

**Components of JMeter**

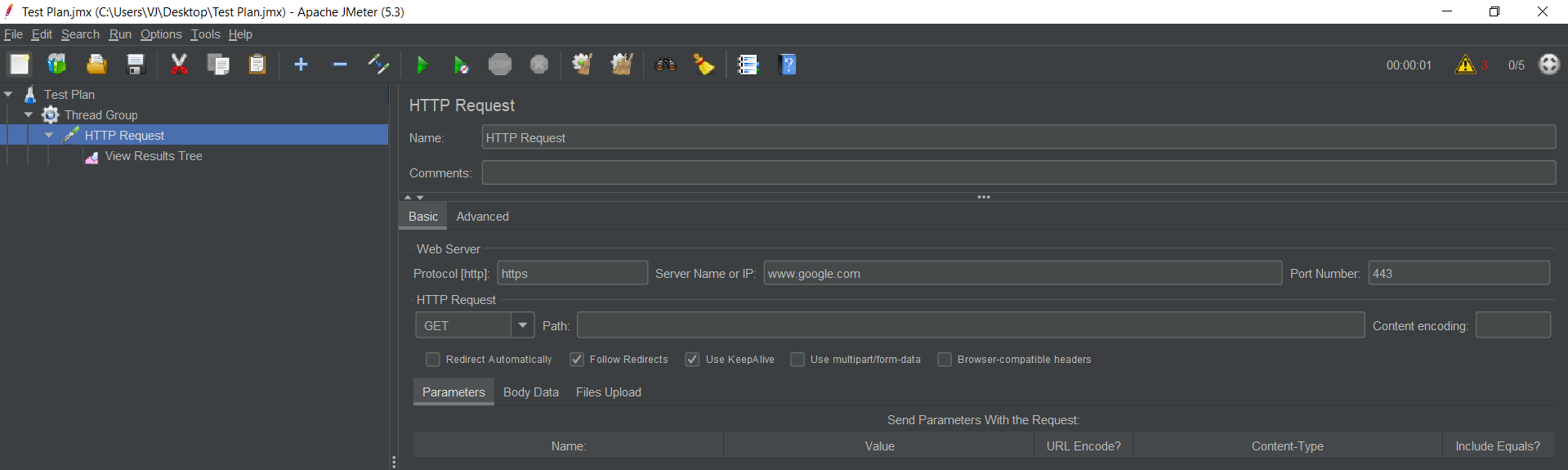
Test Plan – When we launch JMeter, it open with two default components Test Plan and Work Bench. All the components of JMeter are added in Test Plan means it is a container of all the components. One .jmx file can have only one test plan.

Thread Group – Here we define,

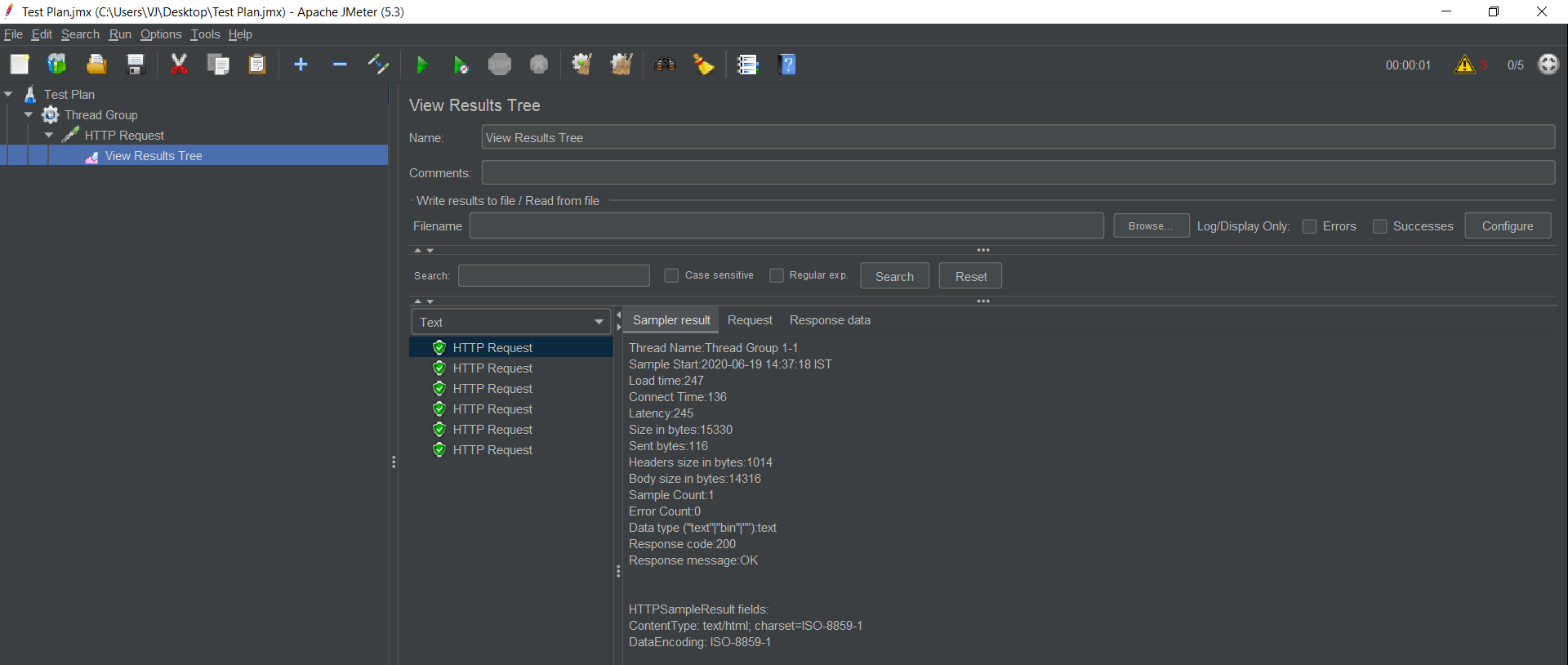
* + Number of Threads – Number of virtual users
  + Ramp up period – Total time allowed for complete execution of all threads.
  + Loop Count – How much time you want to execute the scenario

Note -: The ramp-up period tells JMeter how long to take to "ramp-up" to the full number of threads chosen. If 10 threads are used, and the ramp-up period is 100 seconds, then JMeter will take 100 seconds to get all 10 threads up and running. Each thread will start 10 (100/10) seconds after the previous thread was begun.

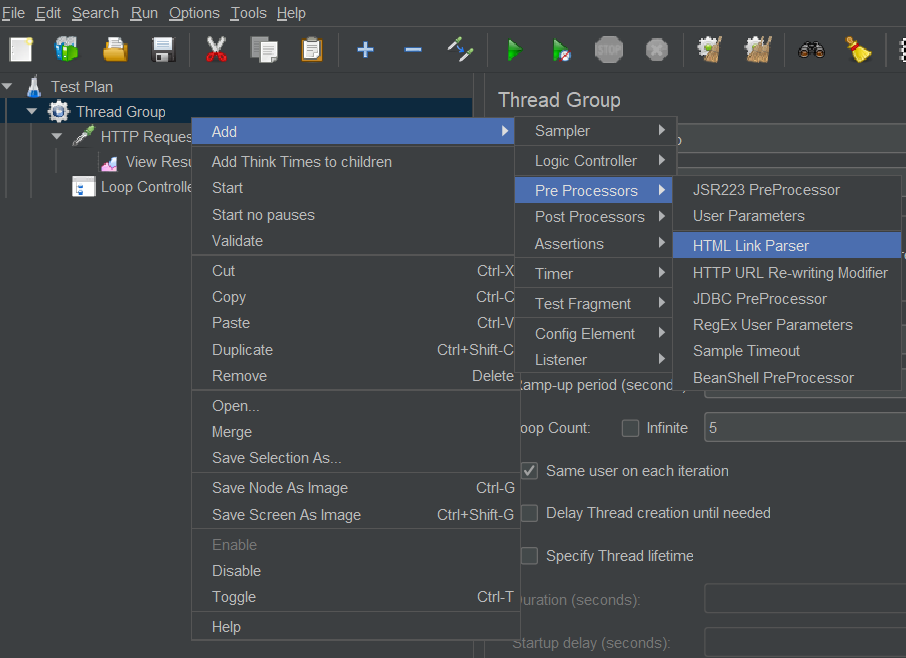
Sampler (HTTP Request) – It should be added under Thread Group. Samplers are used to interact with the servers.



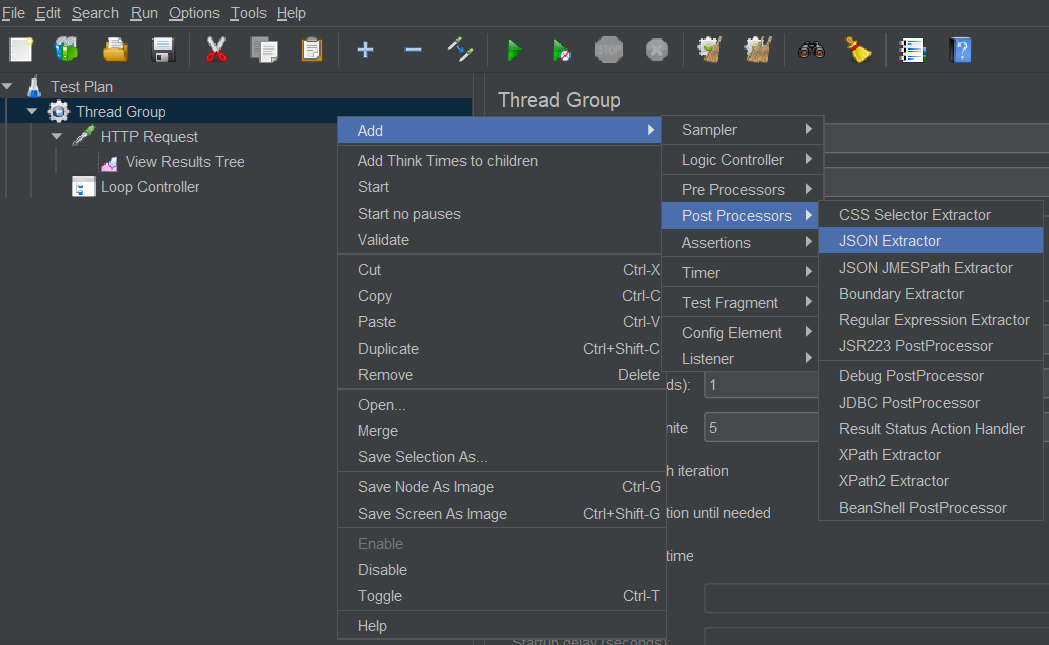
Listener (View Result Tree) – It is used to view the results. All the time shown here are always in milli seconds. In listener, we can also choose ‘view results in table’, ‘graph results’.



Preprocessors 🡪

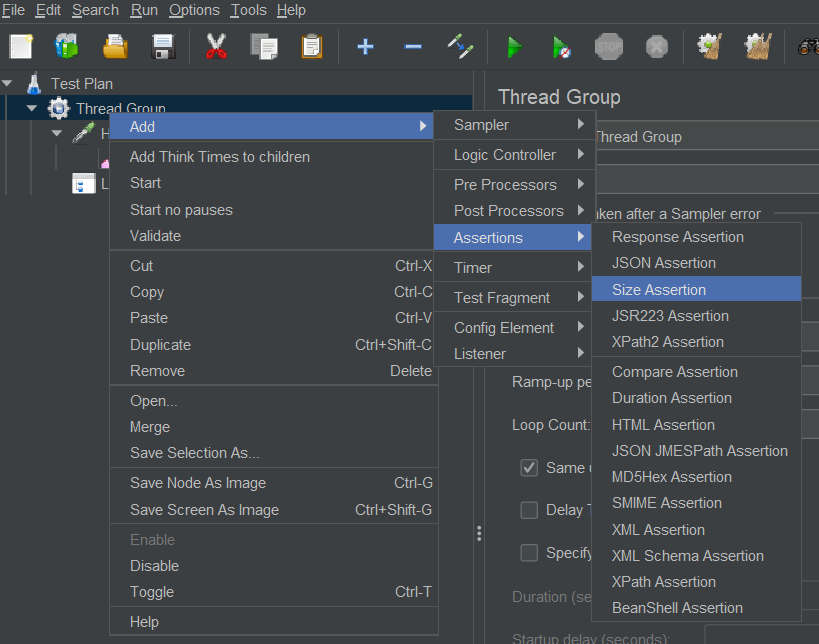


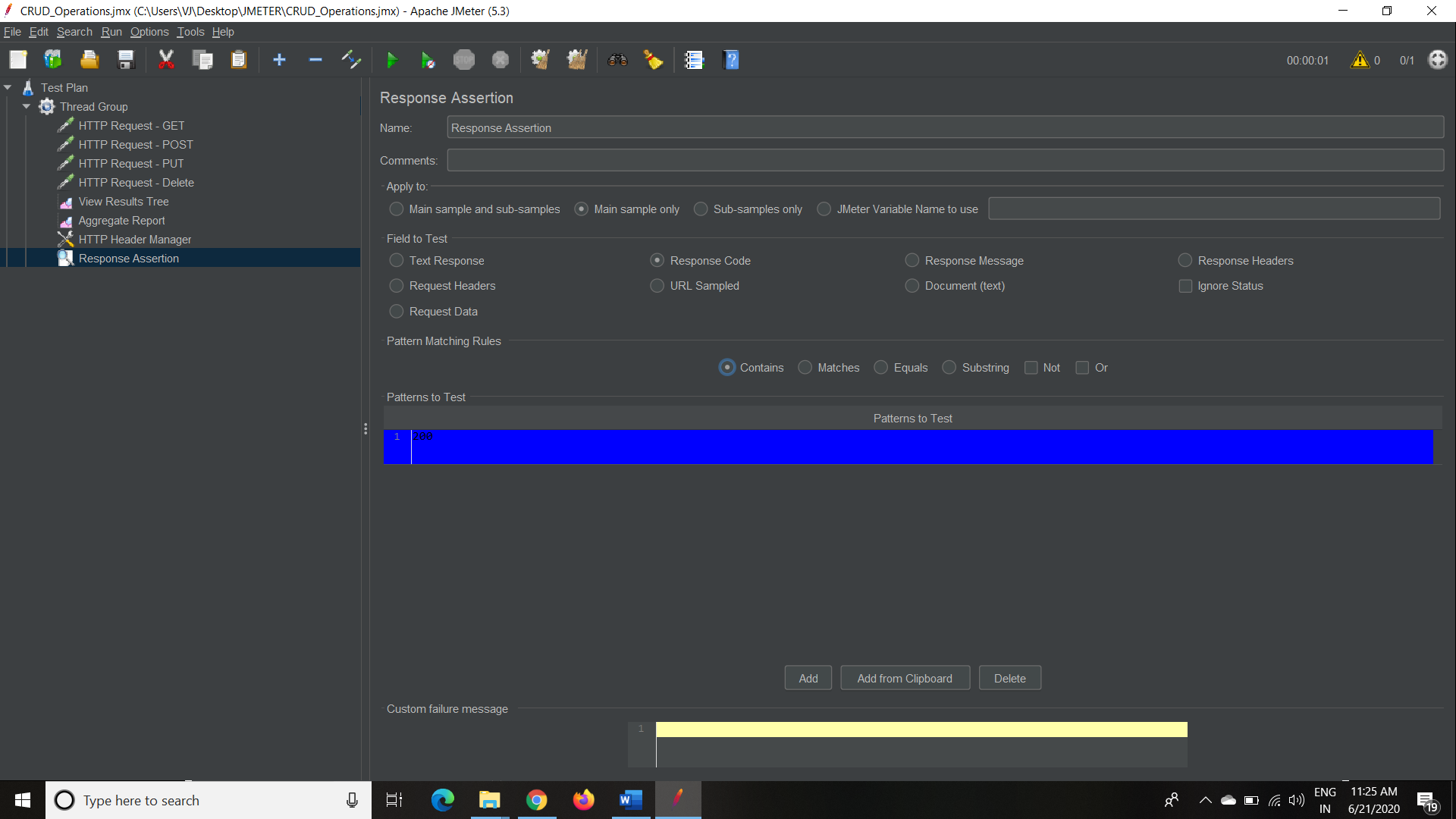
Post processors 🡪



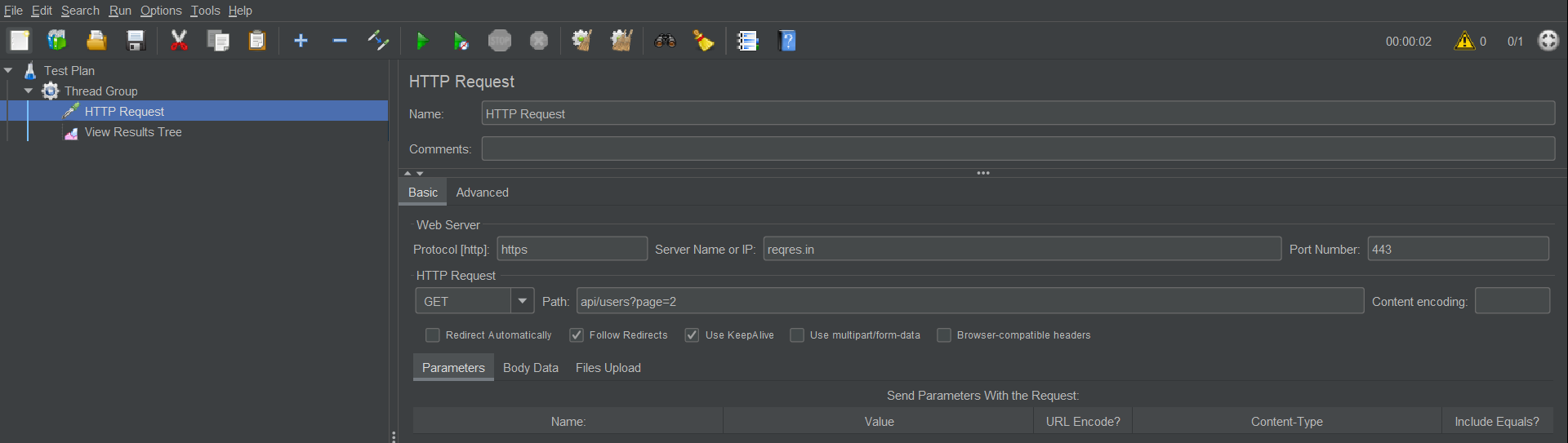
Assertions 🡪

Response Assertion – It is used to assert the response of request.

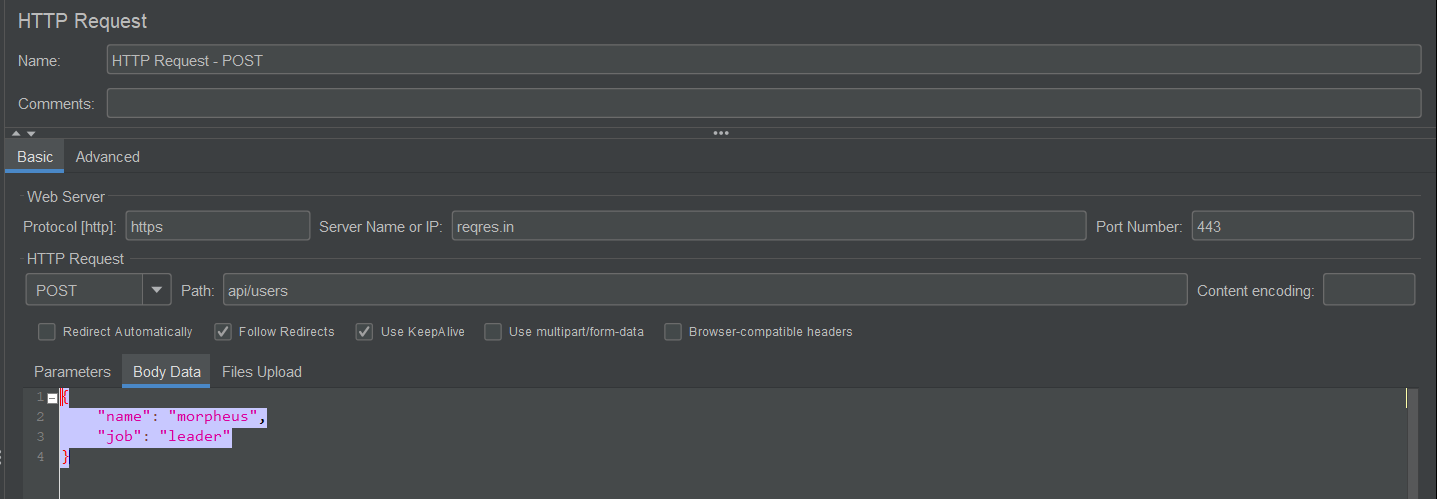




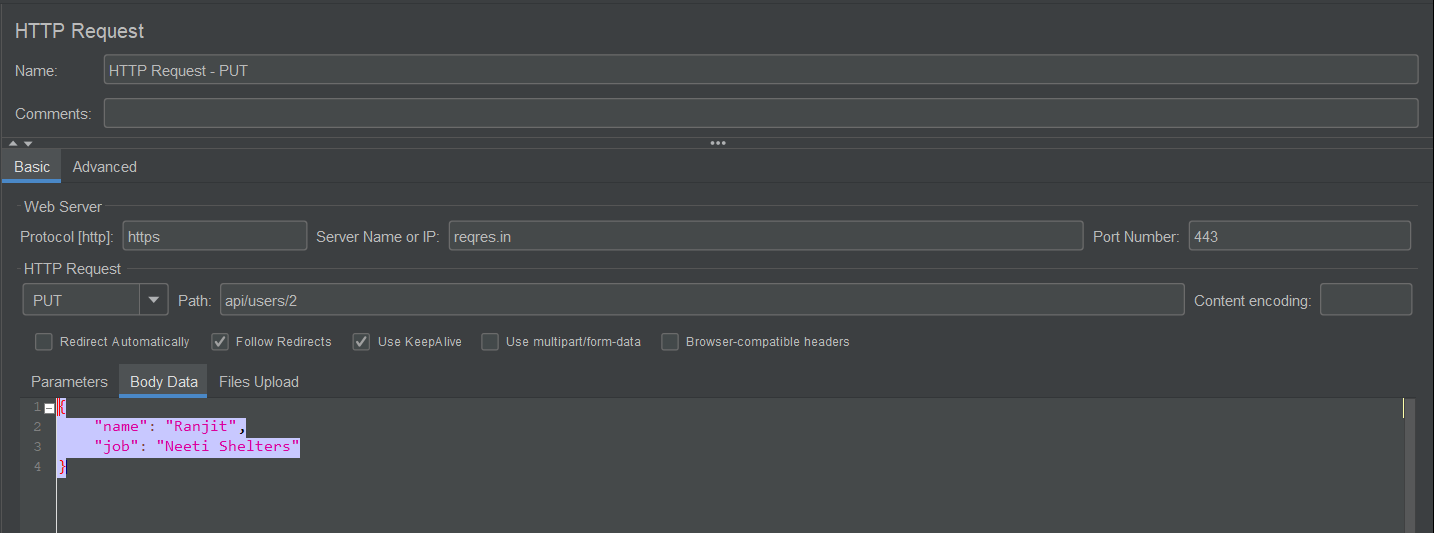
GET Request Example



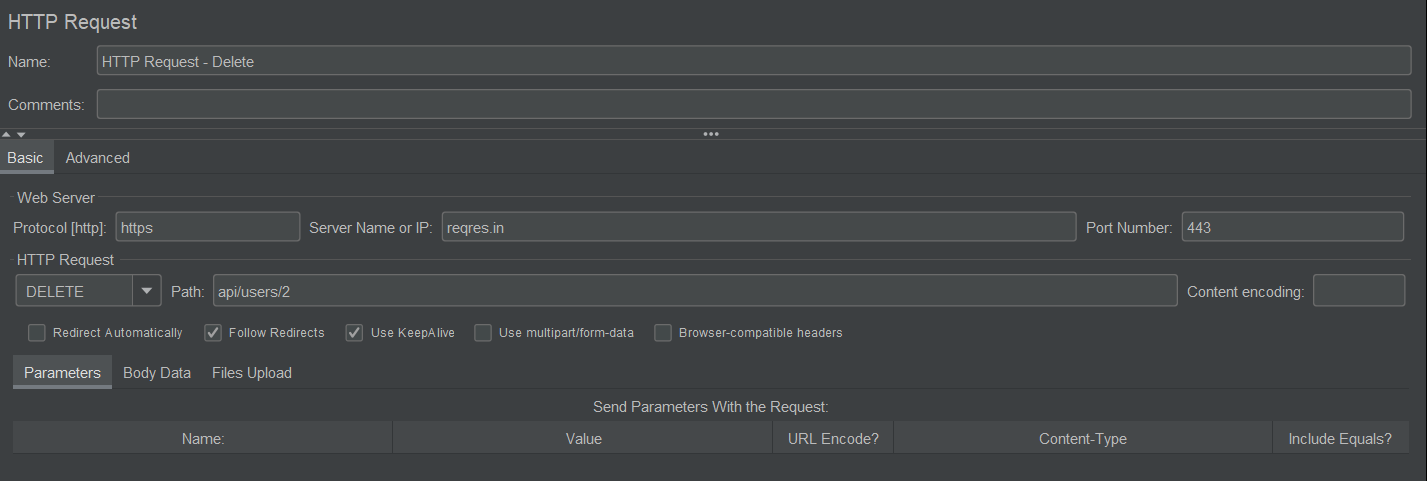
POST Example



PUT Example



DELETE Request –



Project is at - "C:\Users\VJ\Desktop\JMETER\CRUD\_Operations.jmx"

* Port number for HTTPs is 443 and for HTTP is 80.

**Terminologies 🡪**

Performance Testing – It is defined as a type of software testing that is conducted to ensure software applications that will perform well under their expected workload.

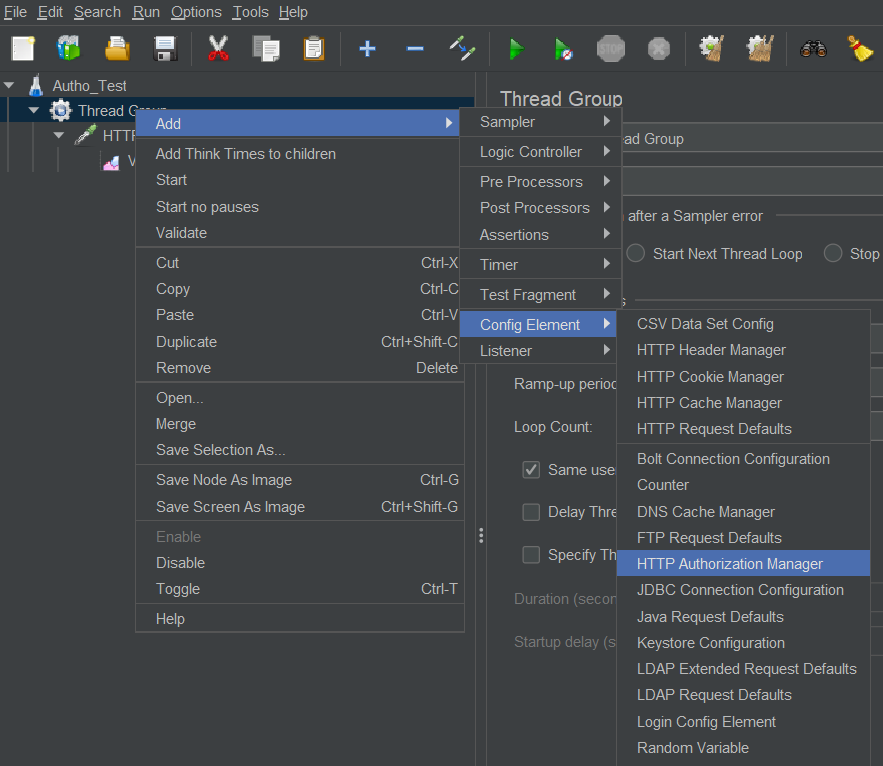
Load Testing - A type of Performance Testing used to evaluate the behavior of a system or component when the load on the system (via users and transaction) progressively increases up to and including peak levels means testing at desired load.

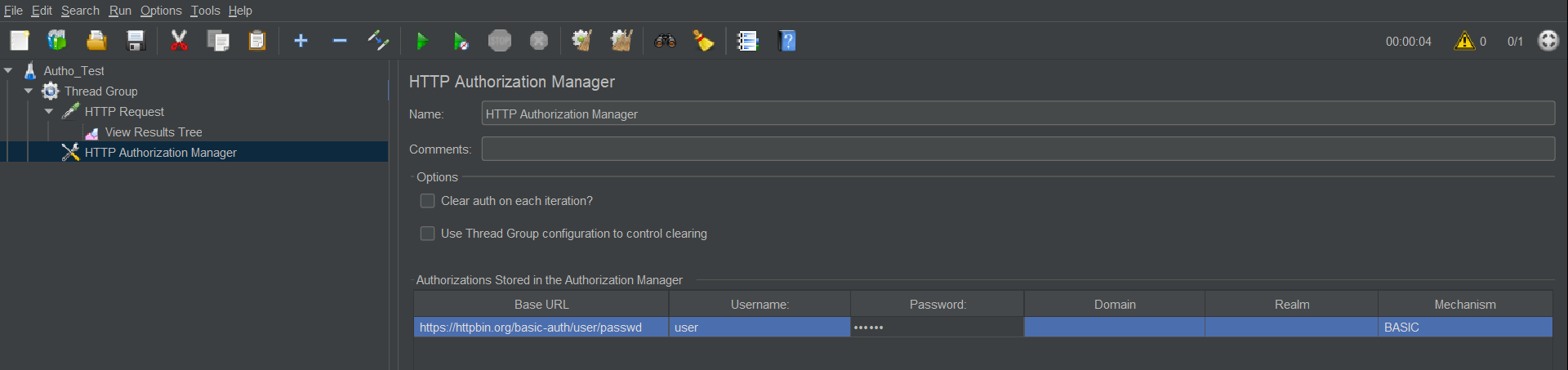
Stress Testing - A type of Performance Testing used to evaluate the behavior of a system or component when subjected to load beyond the anticipated workload or by reducing the resources the system can use, such as CPU or memory. It is also known as Endurance Testing. It is used to find a point where software / hardware breaks. Here also we check whether systems displays effective error message when the system is under stress.

**Authorization in JMeter 🡪**

To add Authorization, right click on Thread Group -> Add -> Config Element -> HTTP Authorization Manager.

In Base URL, provide the complete URL

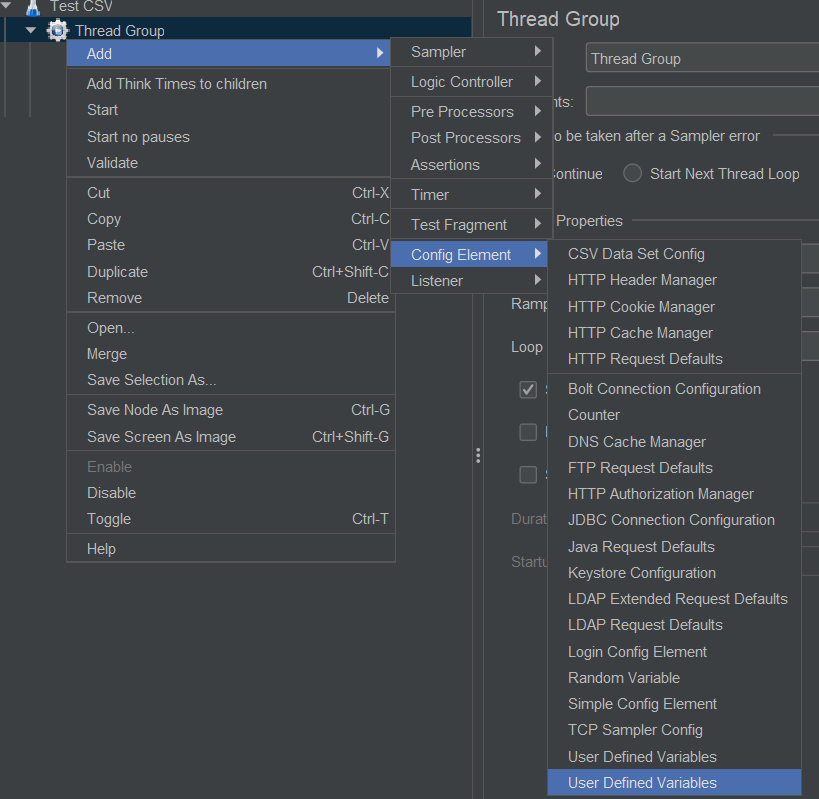




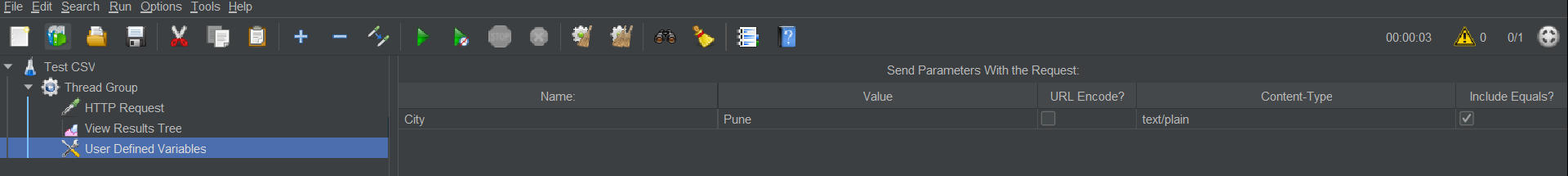
**Parameterization in JMeter 🡪**

To create any variable, right click on Test Plan / Thread Group -> Add -> Config Element -> User Defined Variables.

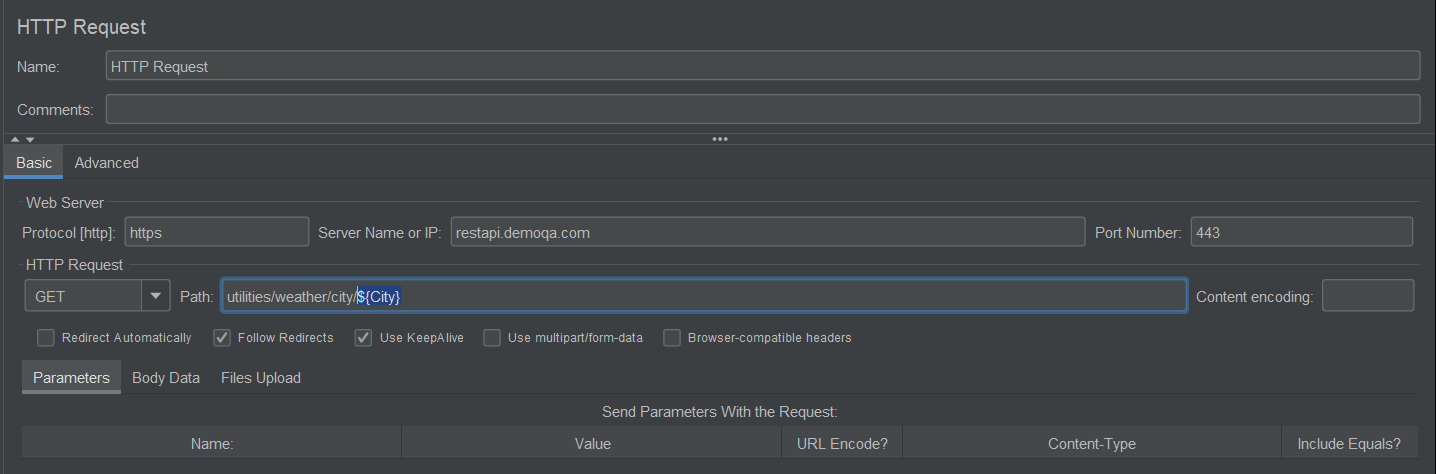
Note :- If variable is created under Test Plan, then it will be applicable to all the Thread Groups under it. But if variable is created under Thread Group then it is applicable to that specific Thread Group and will not be accessible to other Thread Groups of same Test Plan.



In this section, create a parameter (say City and provide value Pune).

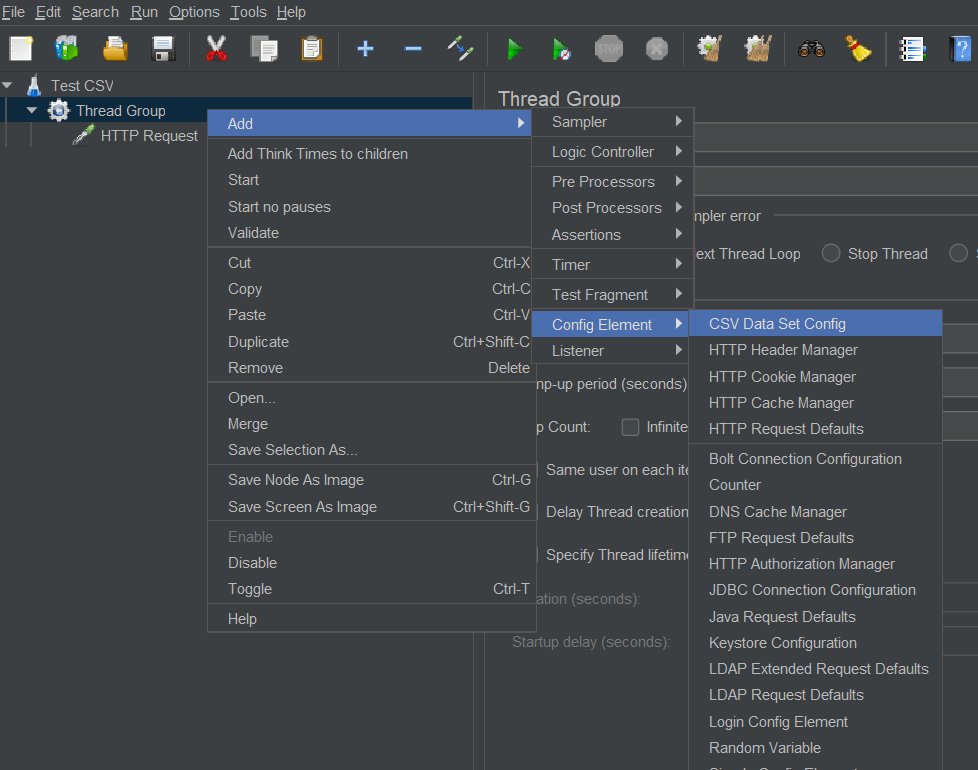


Now provide this parameter in HTTP request. Now move to HTTP request and change Pune with ${City}

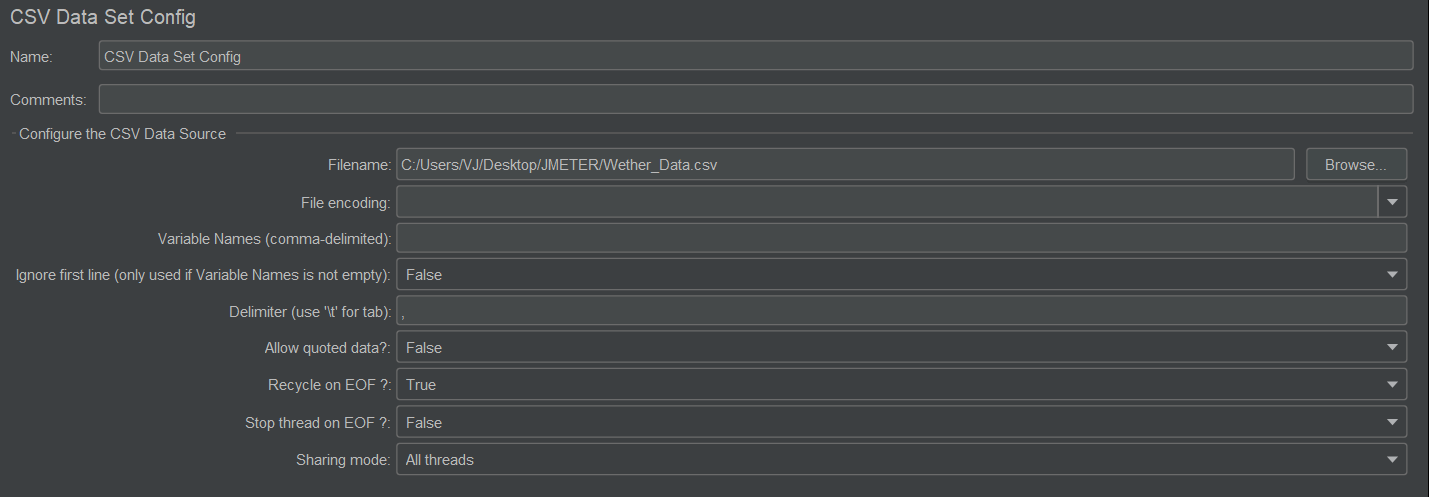


**Reading data from CSV**

To add CSV Config data element, right click on Thread Group -> Add -> Config Element -> CSV Data Set Config



Provide the file name

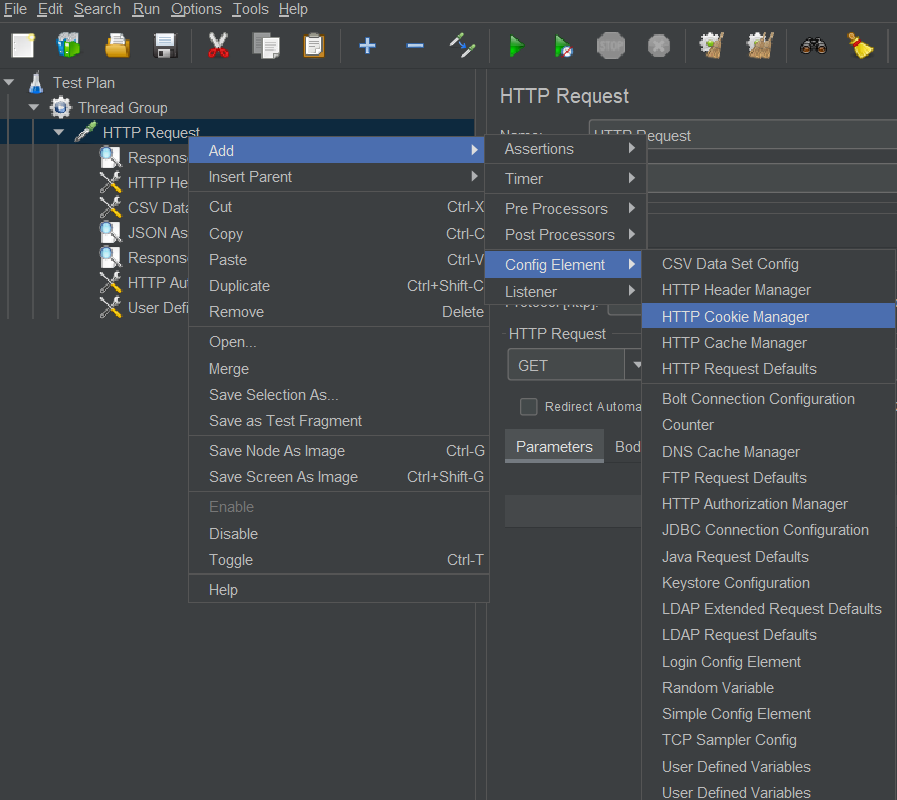


Modify request same as above to fetch the data.

Note -: If you have 5 set of data in CSV file then you must provide at least 5 ‘Number of Threads’ in ‘Thread Group’. If you provide more than 5 number of threads then some data sets will be executed again.

**Cookie Manager**

To add cookie manager, right click on Thread Group -> Add -> Config Element -> HTTP Cookie Manager



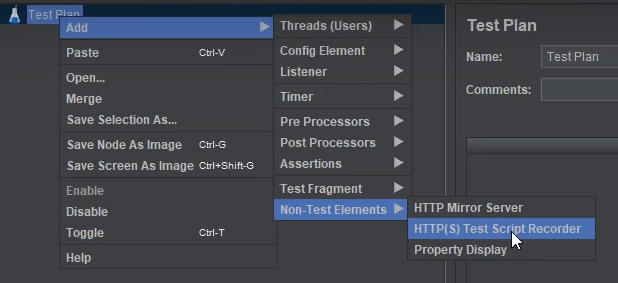
* By using cookie manager, we can bypass the authentication.

**Use of Test Script Recorder 🡪**

The purpose of test script recorder is to record a scenario (say login, search an item, add to cart and logout) and capture all the APIs using Proxy. So that next time you can just run this test script instead of manually running the scenario again.

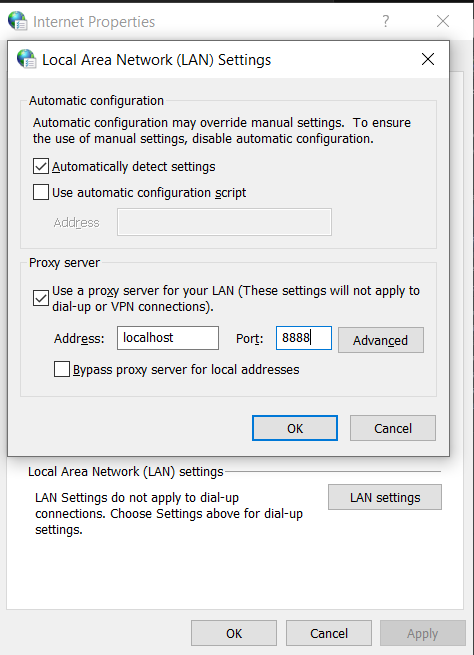
replay it by varying the number of threads, loop count etc as a part of Load Test.

* To create a new recorder,

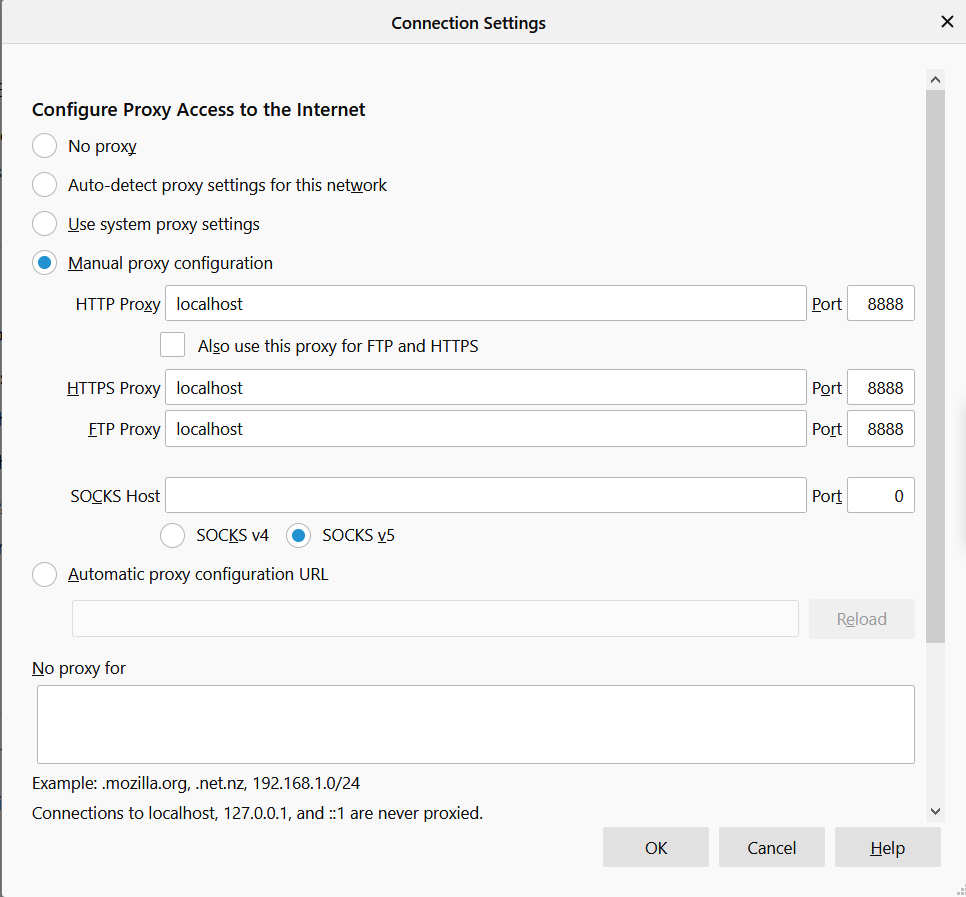


Configuring Internet Options,

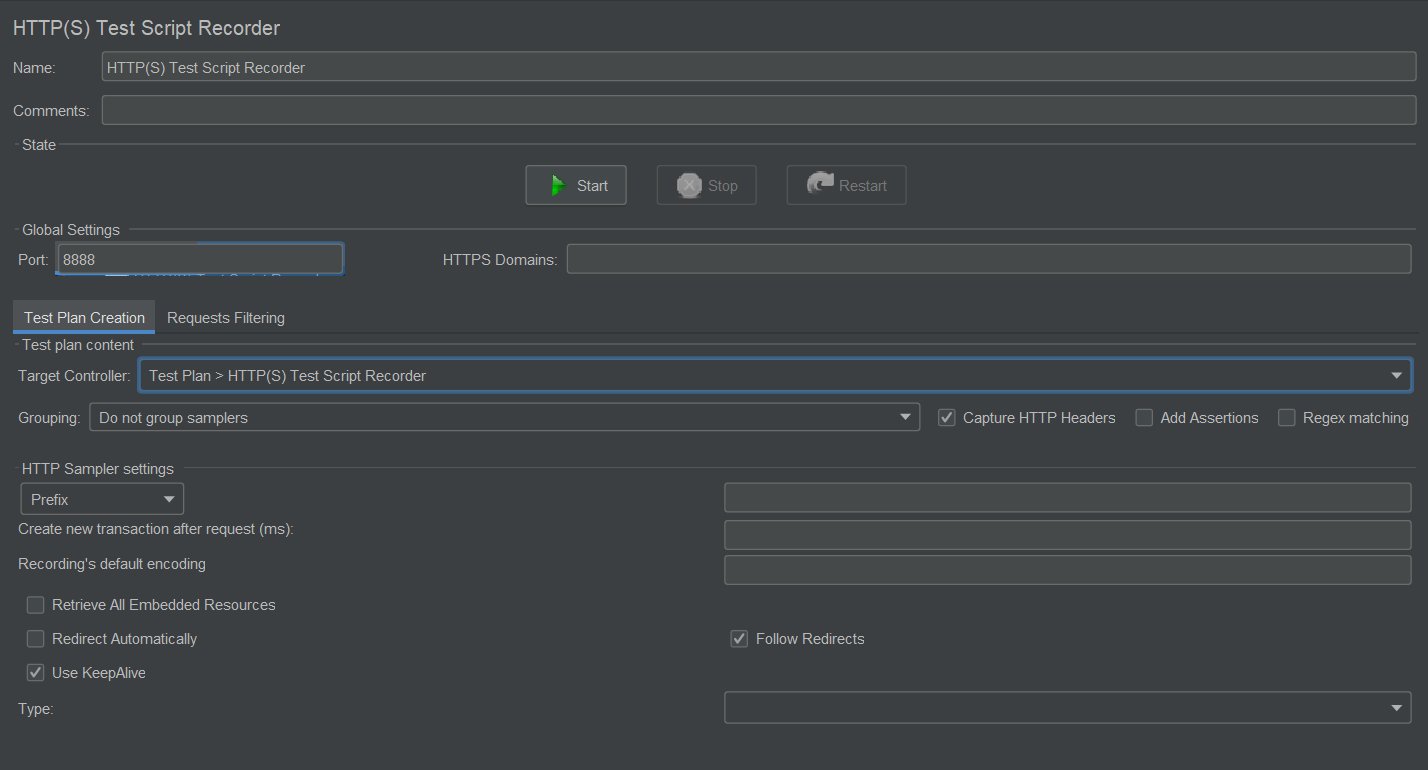
* Go to Internet Options on your system, Connections -> LAN Setting -> select proxy server -> Address as ‘localhost’ and Port as ‘8888’.



* This is global setting, we can configure proxy for only FireFox browser instead of above option which affects to complete system.
* From FF hamburger menu, select Options -> Network Setting (at the bottom) ->



* Now the start proxy from JMeter and go to <http://classic.crmpro.com/>
* At this point we can access HTTP but not HTTPs (as certificate is not added).
* To ignore some specific type of files being recorded, got to ‘URL Patterns to Exclude’ -> ‘Add Suggested Exclude’ and enter extensions (like jpg, css, gif) separated by pipe |
* In order to record any HTTPs connection, we need to import JMeter certificate to FF browser (Options -> Privacy & Security -> View Certificate) which is created by default by Jmeter at Jmeter Directory -> bin
* In JMeter, change ‘Target Controller’ to ‘Test Plan > HTTPs Test Script Recorder’ and click on Start.

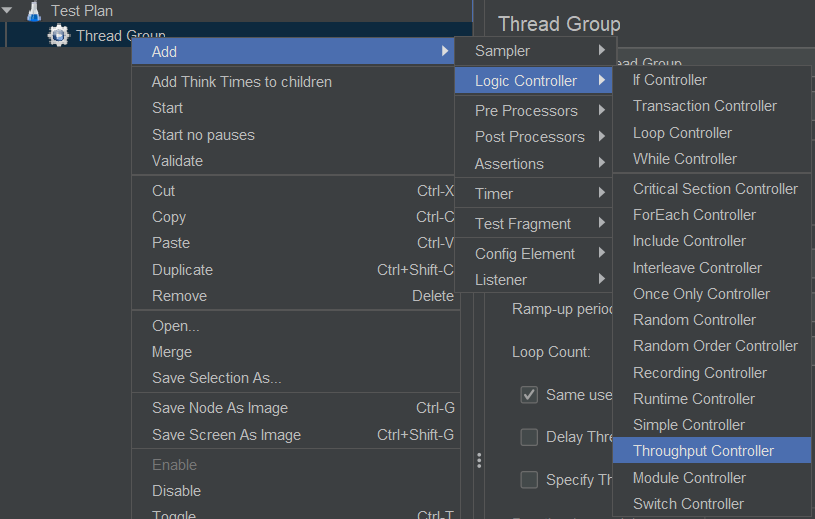


**Throughput Controller 🡪** https://www.youtube.com/watch?v=ChzsqP2XHWY&t=183s

Throughput Controller is used to create distributed load test plan.

General industrial scenario is like if 100% people visit home page, only 20% will visit About page or 80% will visit careers page. To achieve this distribution, Throughput Controller is used.

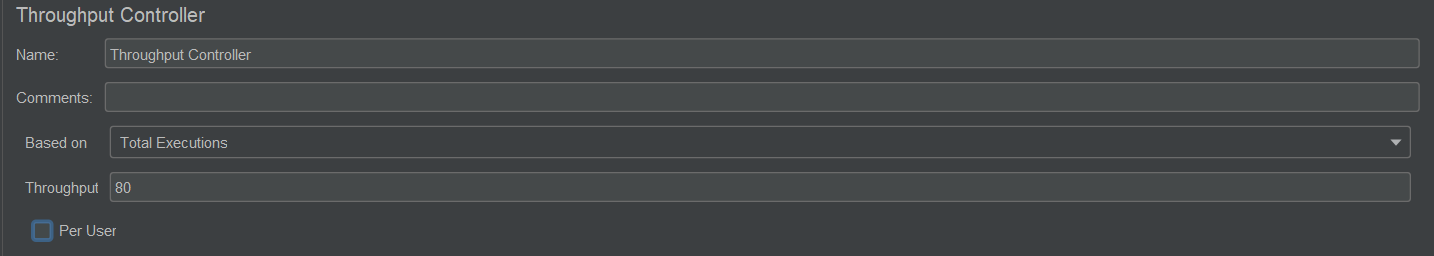
It needs to be added in Thread group and present under ‘Logic Controller’ section.



We add HTTP Sampler inside this Throughput Controller.

Throughput Controller will have two options, Based on –

* + Total Executions – will execute those many times (‘Per User’ checkbox to be used)
  + Percentage Execution – will execute provided % times (say 80%) out of the total provided virtual users.



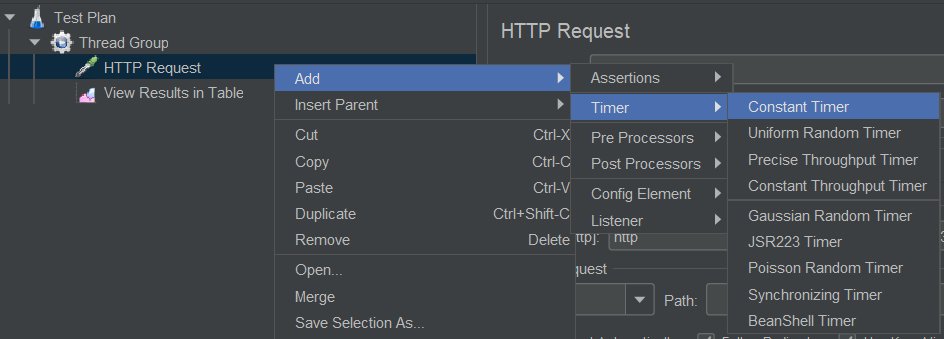
Complete project is at "C:\Users\VJ\Desktop\Switch\API\_Testing\JMETER\Throughput Controller.jmx"

**Timers in JMeter 🡪** https://www.youtube.com/watch?v=EUgLRmlkTGQ

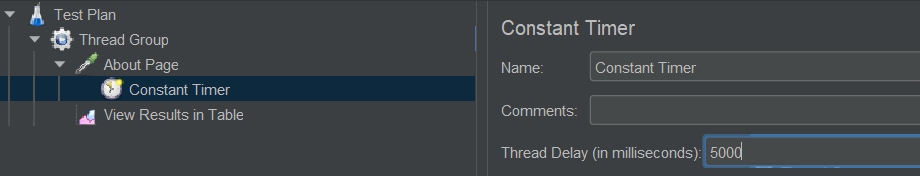
Timers help us to set up realistic performance test.

It is used to add delay between the threads so that we can avoid over flooding the server and can achieve real time behavior.

Timer can be provided at thread group or sampler level. We provide time in milliseconds.

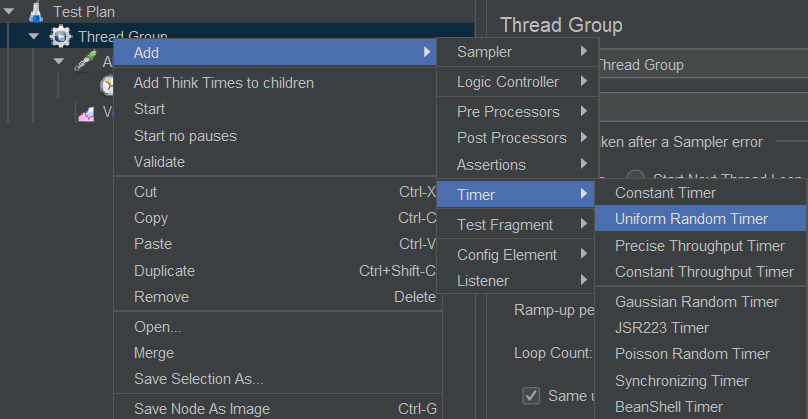


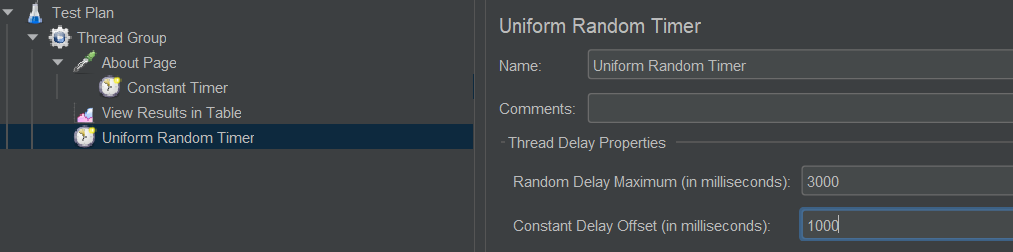
1. Constant Timer - JMeter will send the request only after finishing the provided thread delay time.



1. Uniform Random Timer- If we do not want fix delay then we choose random timer. It accepts two arguments ‘Random Delay Maximum’ and ‘Constant Delay Offset’. This random delay is calculated as,

0.x \* Random Delay Maximum + Constant Delay Offset, where x is any random number from 0 to 9.





"C:\Users\VJ\Desktop\Switch\API\_Testing\JMETER\Timer.jmx"

**UI Testing by Recording Script using BlazeMeter 🡪** <https://www.youtube.com/watch?v=WHhPmMV2Ico&list=WL&index=10>

Firstly we need to add plugin of BlazeMeter in the chrome browser.

During UI testing, we need to start the recording for BlazeMeter and after finishing, need to export it as JMeter or JMeter + Selenium file.

Open this saved file in JMeter and perform testing by increasing virtual users.

**Handling AJAX Components 🡪** https://www.youtube.com/watch?v=gb1vah009gc&list=WL&index=9

Firstly we need to record the script using BlazeMeter and import it in JMeter.

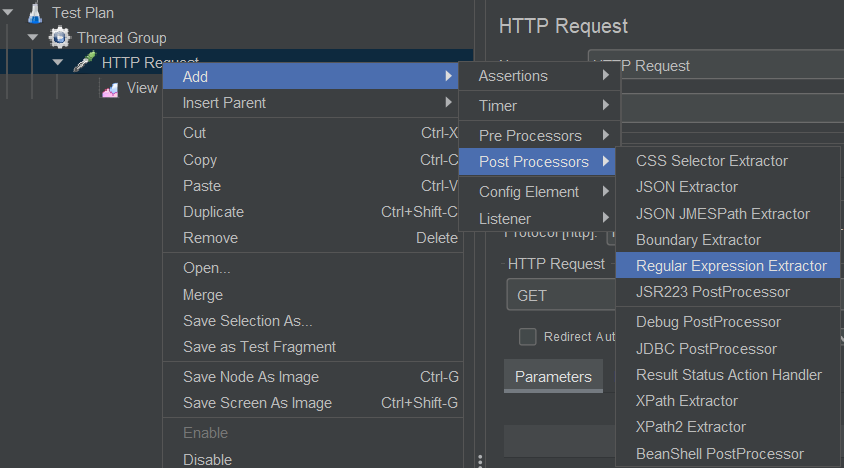
Logic is to extract the XSRF token from the request of firstly successfully executed request and pass it with the help of regular expression (parameterizing) to subsequent requests which are AJAX.

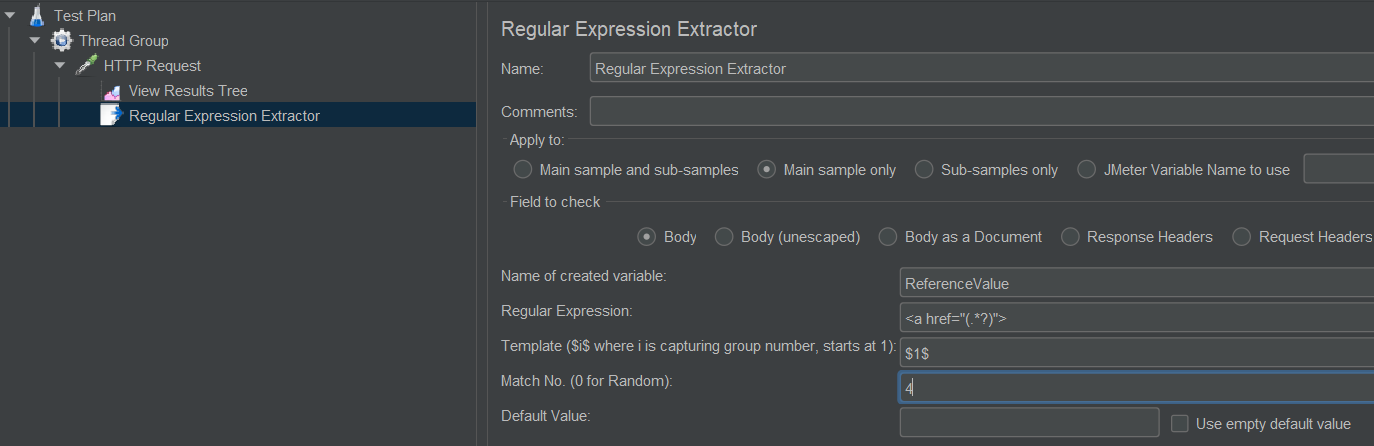
**Correlation (****Regular Expression Extractor) 🡪** <https://www.youtube.com/watch?v=7-NcyZuUQnw>

It is a process of extracting some value from the response of a step and referring it into the request of other subsequent step is called correlation. It happens at runtime, so it is also called as dynamic referencing.

Generally once you login to the webpage, you receive a session ID in the response and you need to refer it for your subsequent requests like using home page, logout etc to maintain valid session. But this ID is generated runtime. So this needs to be extracted runtime and should be given as reference for other requests.

Regular Expression Extractor is a part of Post Processors.



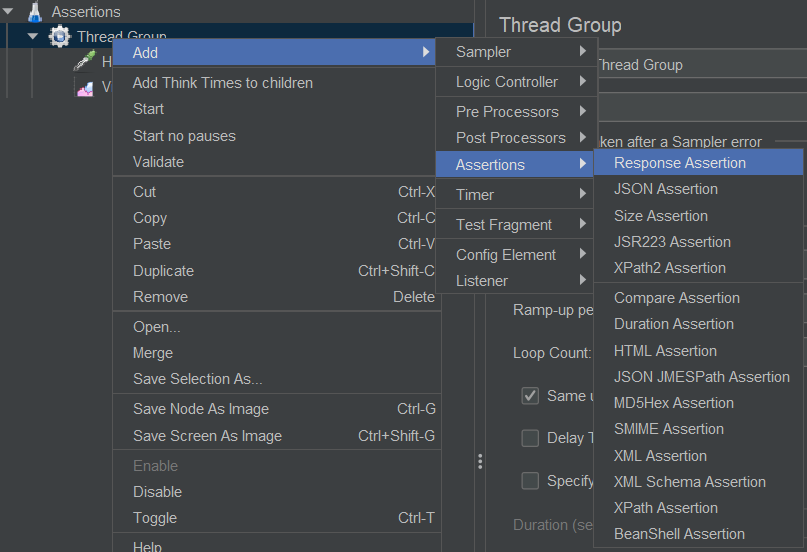


This will extract the fourth href value from the response body and will store in the ‘RefrenceValue’ variable. We can use this variable in our subsequent requests as per need.

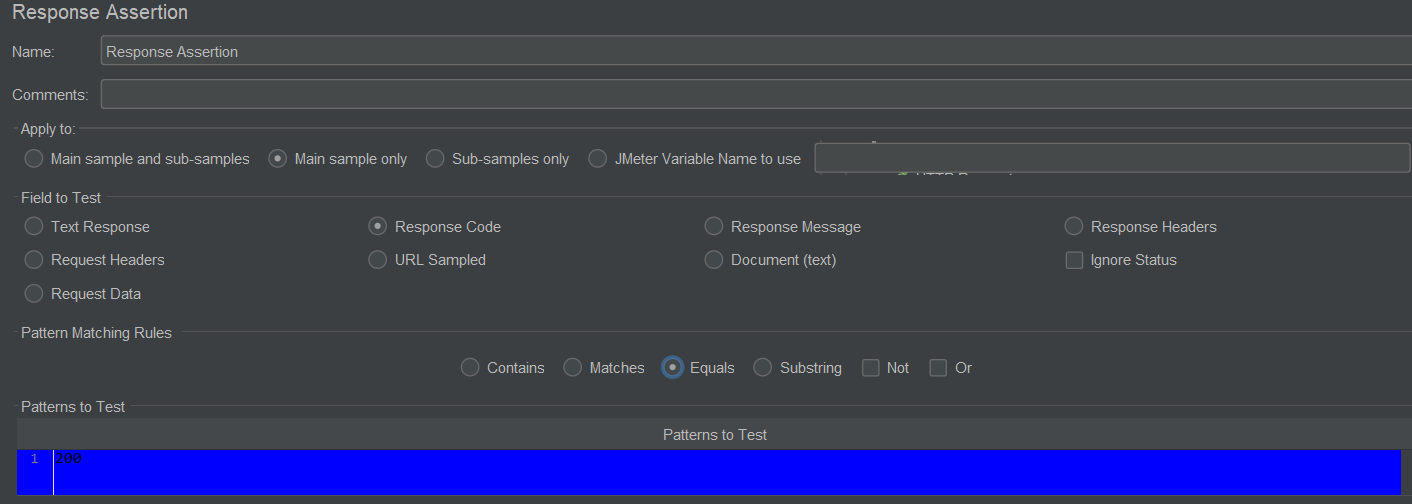
"C:\Users\VJ\Desktop\Switch\API\_Testing\JMETER\Correlation.jmx"

**Assertions 🡪** <https://www.youtube.com/watch?v=mXhC9CtQBC8&t=720s>

Assertions means check on the response.



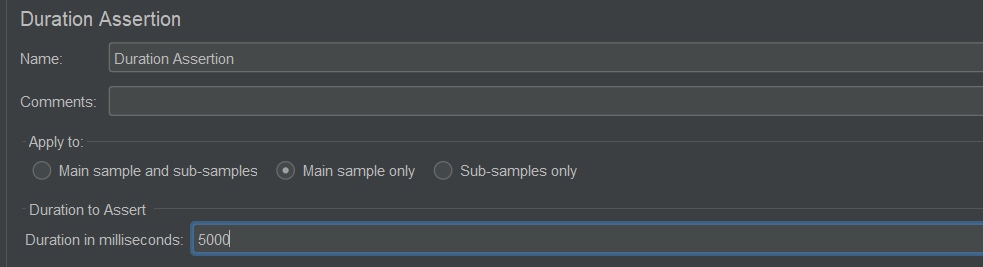
* Response Assertion - Here we can assert ‘Response Code’, ‘Text’ from response body, headers etc.



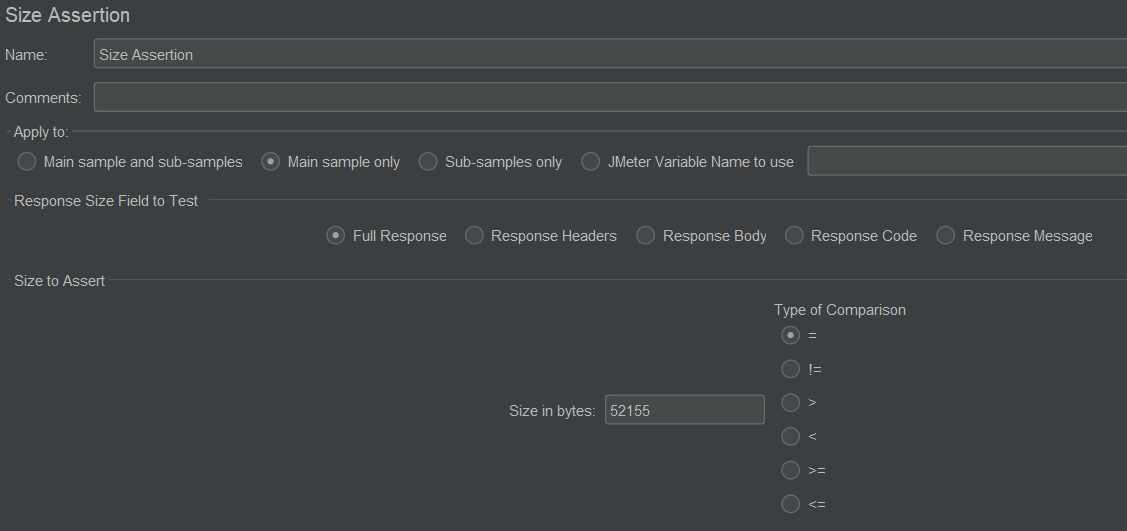
In View Result Tree / View Result Table, it will automatically create a sub category for every created assertion and will display green / red status accordingly.

In case of assertions, we can use ‘Assertion Results’ listener, it will show comparison of actual vs expected value only if assertion is failed.

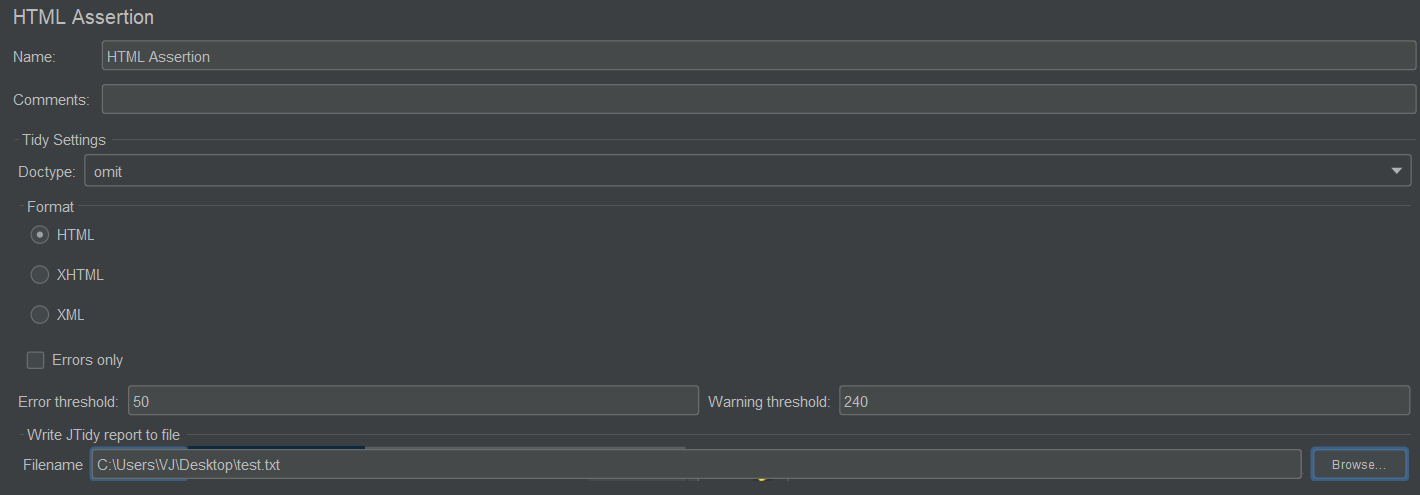
* Duration Assertion – If response took more time than specified then assertion will fail.



* Size Assertion – If size of response do not follow the mentioned criteria, then assertion will fail. In size assertion, we can use different operators like =, >, < etc.

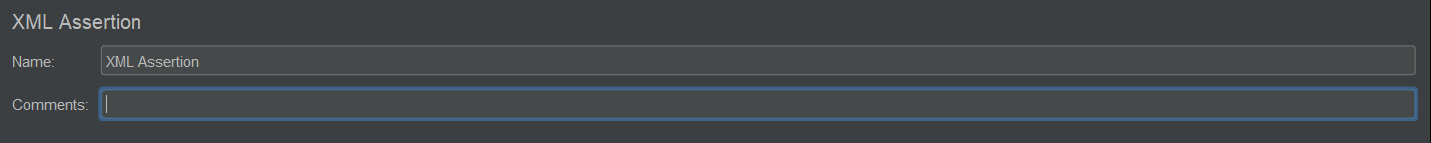


* HTML Assertion – It will check whether the format of assertion is valid HTML format or not. Here we can modify ‘Error threshold’ and ‘Warning threshold’ means if errors and warnings exceed the provided numbers then it will fail. By default its values are zero.



If we want to save all the error detected by HTML Assertion in the response, then we can provide the file in ‘Write JTidy report to file’ option.

* XML Assertion – It will check the format of response. If it is valid XML format then it will pass otherwise it will fail.

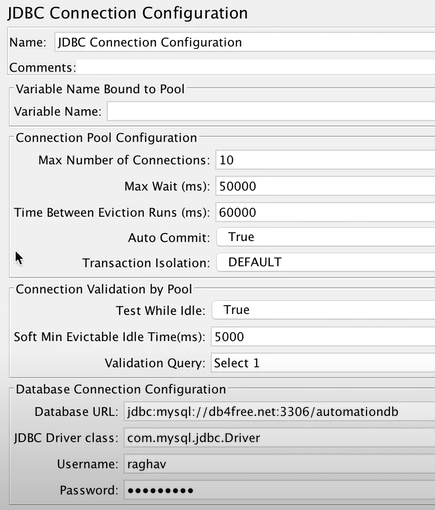


* XPATH Assertion – In response if we want to check any particular node value, we go for XPATH assertion.

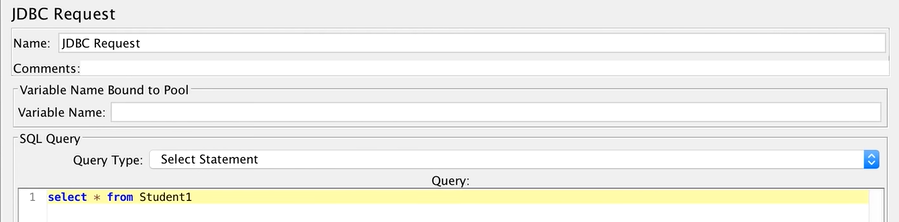
"C:\Users\VJ\Desktop\Switch\API\_Testing\JMETER\Assertions.jmx"

**Creating Database Test Pan 🡪** <https://www.youtube.com/watch?v=oy53KAKHpts&list=PLhW3qG5bs-L-zox1h3eIL7CZh5zJmci4c&index=9>

In thread group, add ‘JDBC Connection Configuration’ from Config element. Before this make sure that you have added MySQL JDBC jar in JMeter lib folder. Here you will provide the database URL, driver class, username and password.



Then in thread group, add ‘JDBC Request’ from Sampler. Here you will write the actual query.



Run this test plan by adding listeners.

**Logic Controllers 🡪**

<https://www.youtube.com/watch?v=8yybsSnvS9s&list=WL&index=12&t=9s>

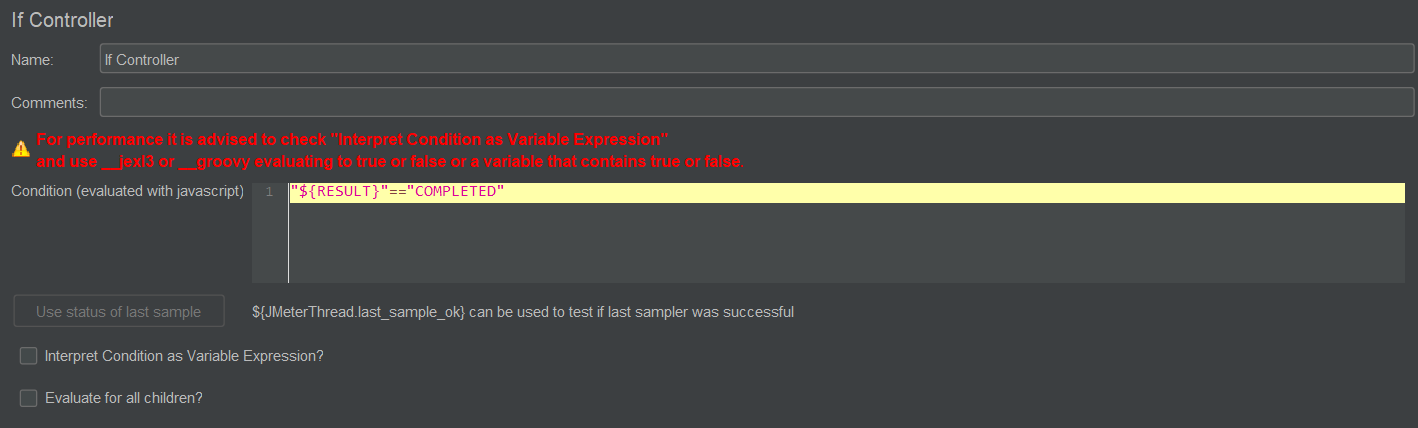
<https://www.youtube.com/watch?v=jLHnS0eg0Gc&list=PLUDwpEzHYYLs33uFHeIJo-6eU92IoiMZ7&index=7>

<https://www.youtube.com/watch?v=sAMlKJ_3DCM&t=147s>

Logic controllers let you handle the order of processing Samplers / Requests in a Thread. It will decide ‘when & how’ to send a request to web server.

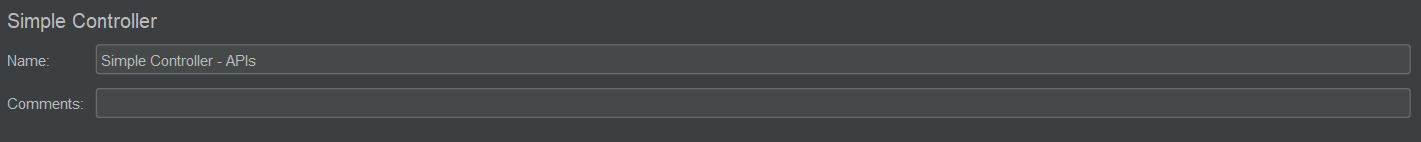
Logic controllers are always present under Thread Group and not under Samplers.

* If Controller – Requests under this controller will get executed only if the specified condition is satisfied. Condition should be written as Javascript code.

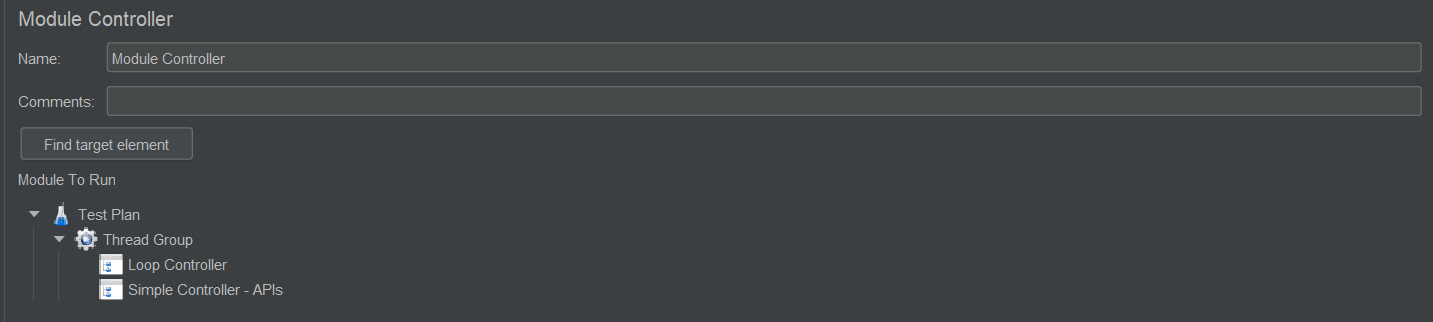


Here, RESULT is a variable with value as COMPLETED.

* Simple Controller – It is simply a folder to hold / segregate API requests under it. It is equivalent to Collection folder in POSTMAN.

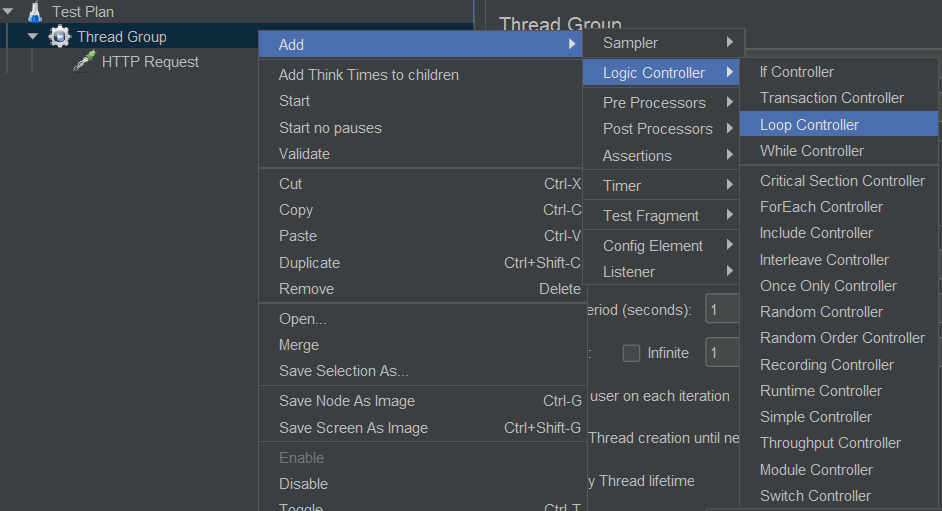


* Module Controller – It will list down all the available controllers in the test plan and will give option to start test plan execution as per selected controller.



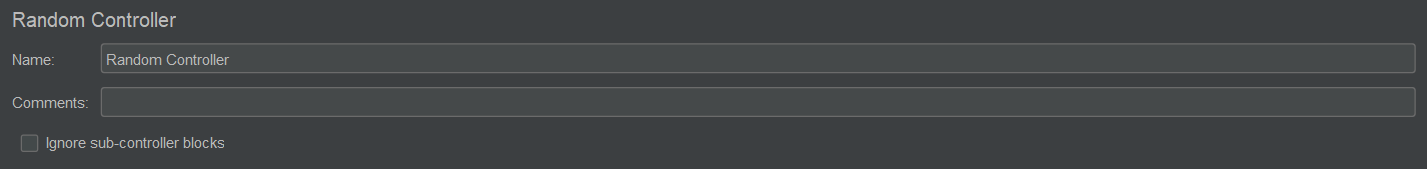
* Loop Controller – It makes the user request to run a specified number of times or run forever.

The ‘loop count’ present in Thread Group is applicable for all the Samplers present under it. But if we want to execute a specific request (Sampler) multiple times then we go for loop controller. Create a Sampler inside loop controller.

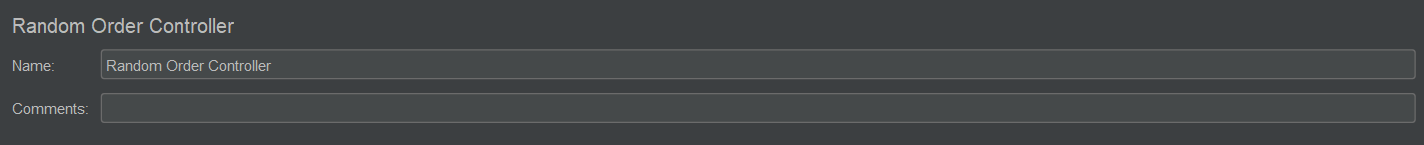




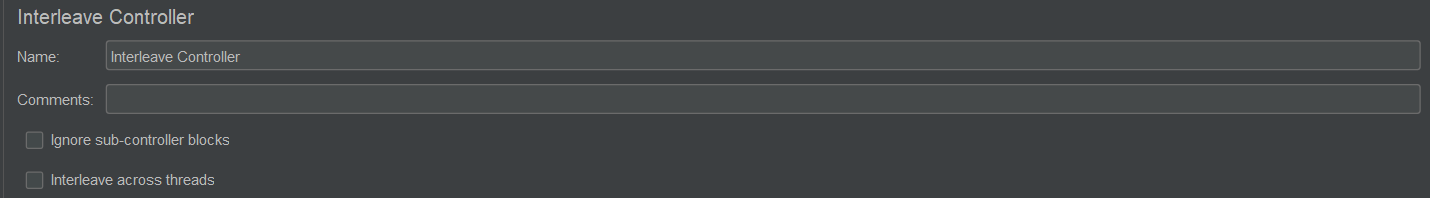
* Random Controller – It will execute the requests under this controller in random manner (not sequentially). In this way it can pick some request multiple times and some requests not even once.



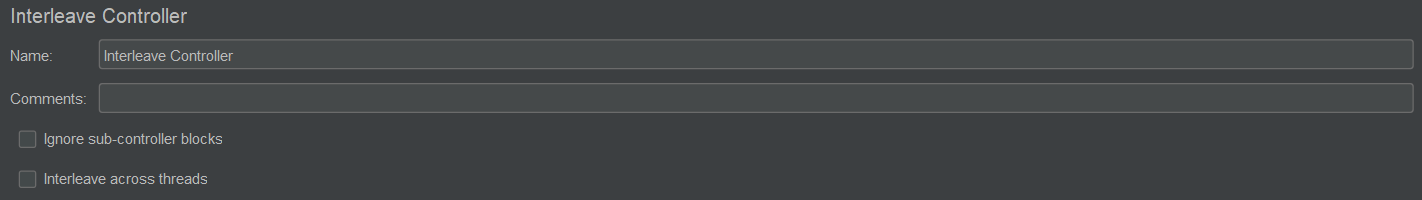
* Random Order Controller – It will execute the entire set of requests randomly for the specified number of times. Means here if there are 3 requests under Random Order Controller and loop count is 3 then total 9 requests will get executed.



* Interleave Controller – It will execute all the request under this controller in sequential order (not randomly).



* Once Only Controller – Irrespective of loop count present in the thread group, it will execute each request present under this controller only once per user. However if you increase the number of users (threads), it will execute each request once for each user irrespective of loop count.

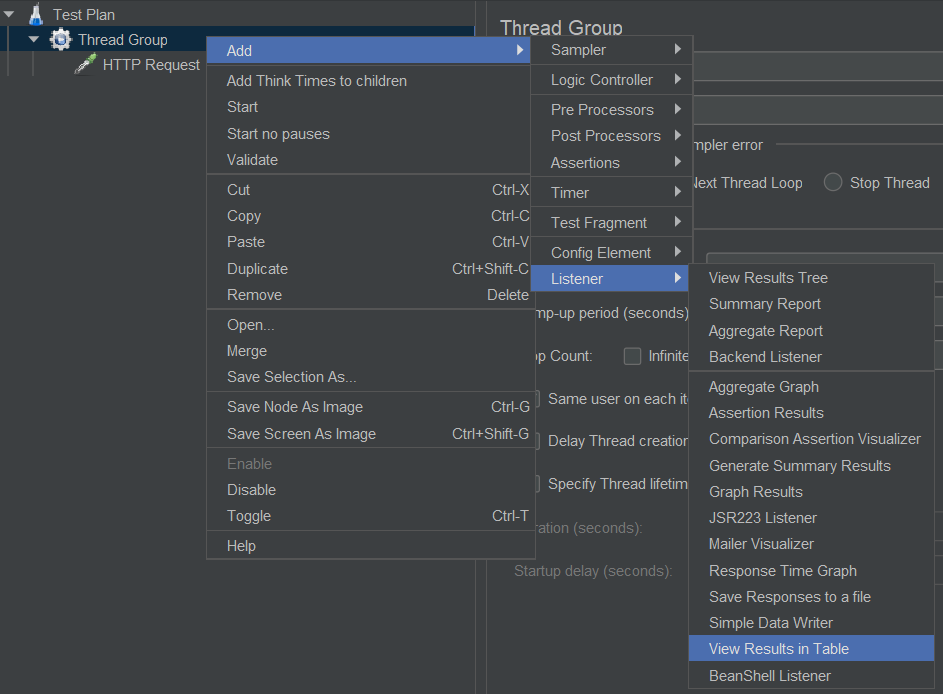


"C:\Users\VJ\Desktop\Switch\API\_Testing\JMETER\LogicController.jmx"

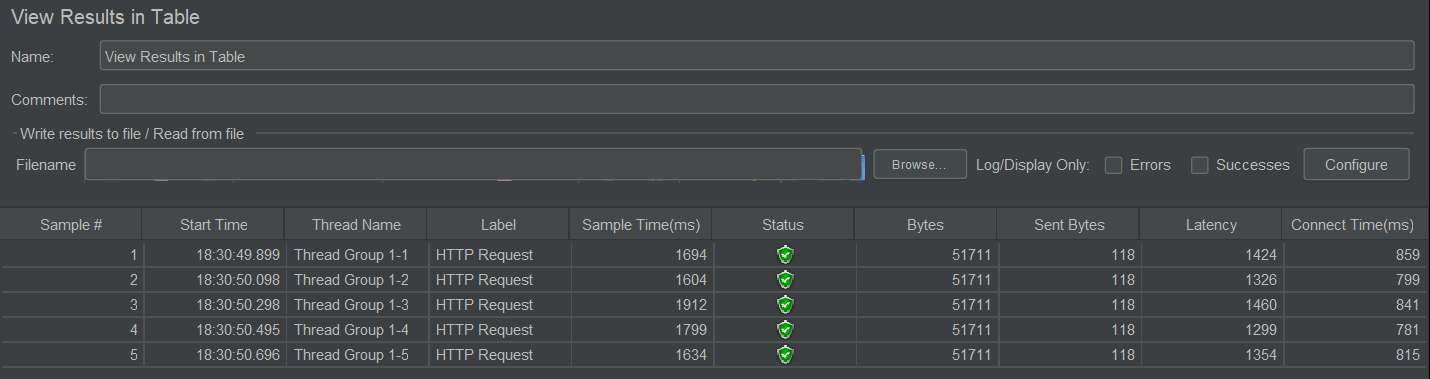
**Listeners 🡪** <https://www.youtube.com/watch?v=5FyVKVAqEJo&list=WL&index=10&t=40s>

"C:\Users\VJ\Desktop\Switch\API\_Testing\JMETER\Listeners.jmx"

Listeners are elements of JMeter which gather the information about the test which is getting executed.



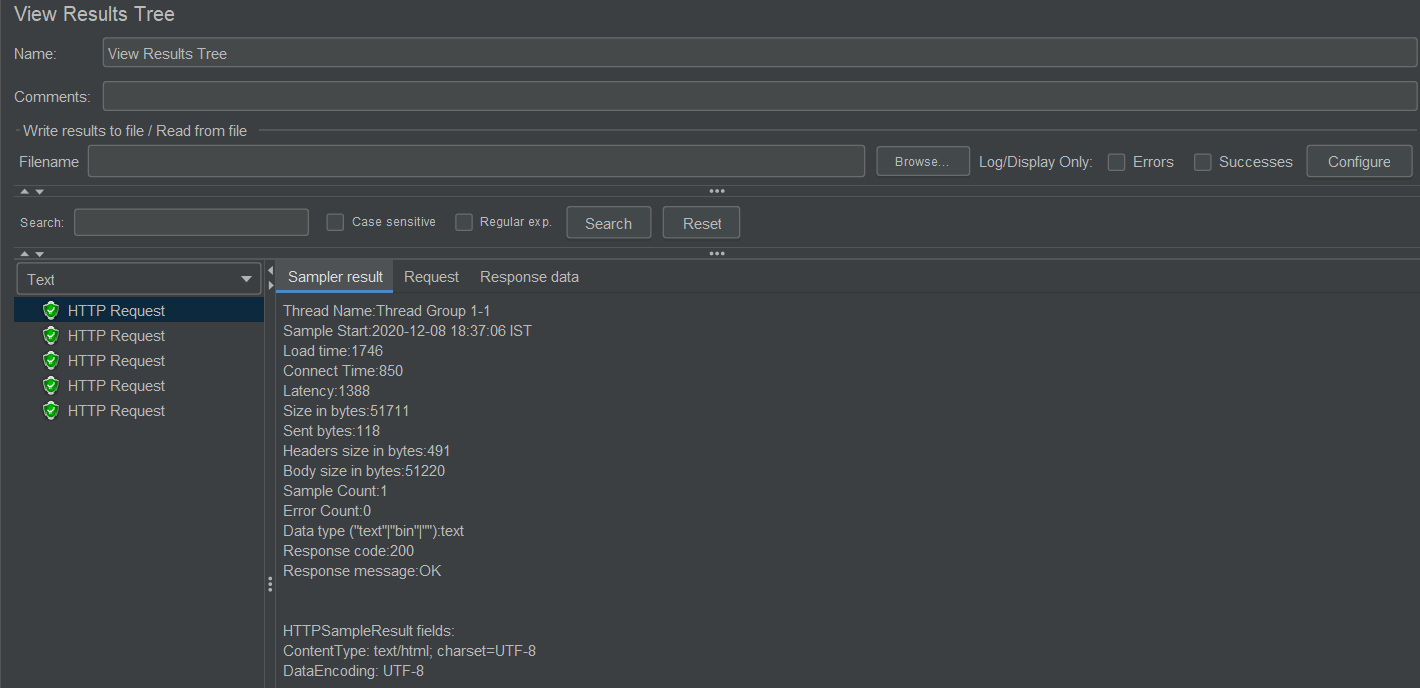
* View Results in Table -



It has columns like Sample time (response time), Status (which gives assertion status, pass / fail) latency (after requesting, time to receive first response byte from the server), connect time (time taken to connect to the server)

For every listener there is a Filename field where user can provide a file and all the result data displayed by that listener will get written in the provided file. It accepts xml & csv file.

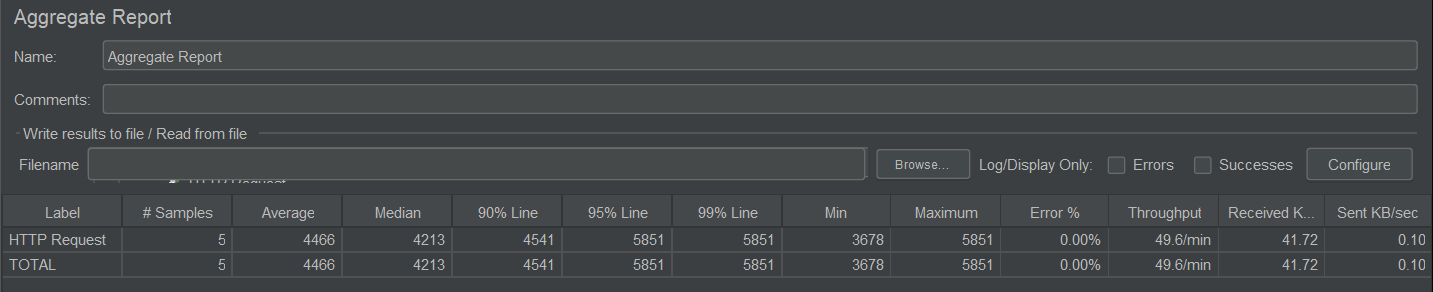
* View Results in Tree -



Here we can get info about the info about the request that we have passed, complete response data etc. However, View Results Tree listener require lot of memory, so it not preferred during performance testing (initially it can be used to check, is test plan working fine or not but later it should be dropped).

* Aggregate Report – It gives a single line report (per Sampler).

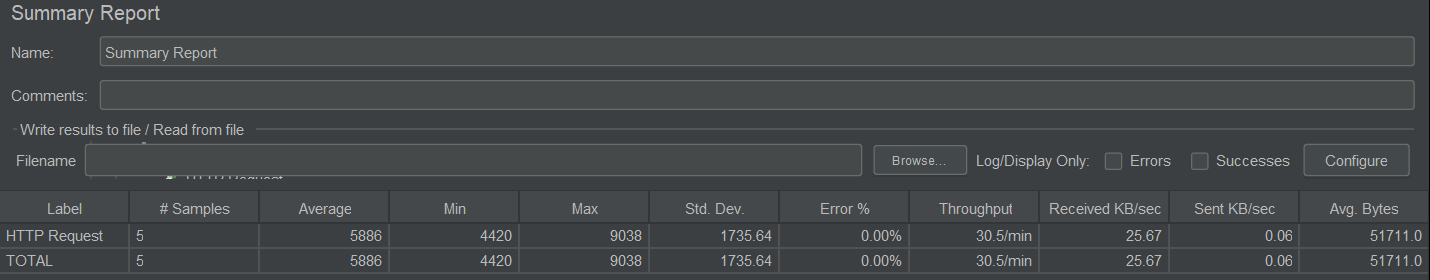
It mainly gives idea about response time with multiple parameters like average, median, min, max etc



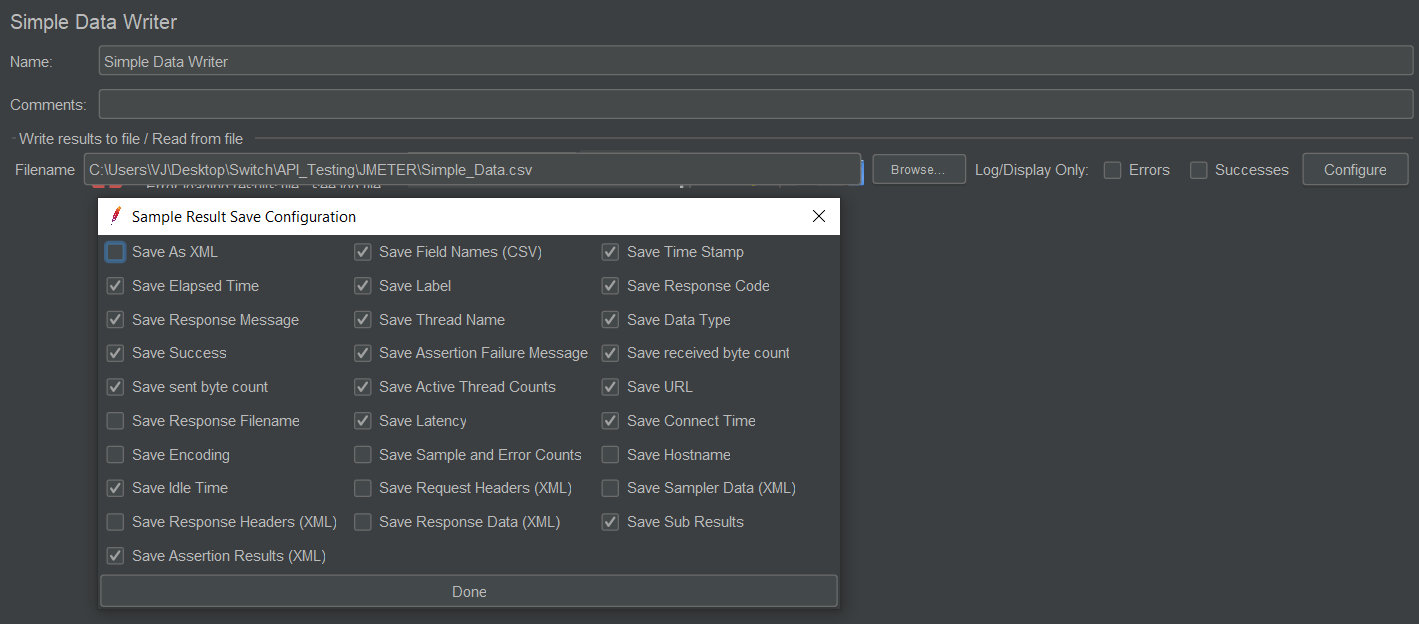
* Graph Results – It will generate graph at runtime in the form of colored points with Y axis as time in milliseconds and X axis as number of samples. Different colored points are for Average, Mediation, Deviation etc. It also consumes heavy memory so not recommended for performance testing.



* Summery Report – It is somewhat similar to Aggregate Report with some extra columns like standard deviation, Error % etc.



* Simple Data Writer – It simply writes report summery in provided file. Using Config option present in front of it, we can choose what type of data should be written in the file.



**Execution Flow of Different Elements in JMeter 🡪**

JMeter always uses below sequence of execution,

* + - Config Element
    - Pre Processors
    - Timers
    - Samplers
    - Post Processors
    - Assertions
    - Listeners

**What you have learnt / done in JMeter ?**

* Configuration for Load Test or Stress Test
* Test Plan
  + Thread Group
    - Sampler
      * HTTP Request
      * JDBC Request (for database validation)
      * FTP Request
    - Logic Controller
      * If Controller
      * Loop Controller
      * Module Controller
      * Throughput Controller
      * Simple Controller
      * Random Controller
      * Random Order Controller
      * Interleave Controller
      * Once Only Controller
    - Config Element
      * HTTP Authorization Manager
      * User Defined Variables (Parameterization)
      * CSV Data Set Config
      * HTTP Cookie Manager
      * HTTP Header Manager
      * JDBC Connection Configuration (for connecting to database)
    - Assertions
      * Response Assertion
      * Duration Assertion
      * Size Assertion
      * JSON Assertion
      * HTML Assertion
      * XML Assertion
      * XPATH Assertion
    - Timers
      * Constant Timer
      * Uniform Random Timer
    - Post Processors
      * Regular Expression Extractor (Correlation)
      * JSON Extractor
    - Listener
      * View Result Tree
      * View Results in Table
      * Aggregate Report
      * Graph Results
      * Summary Report
      * Simple Data Writer
  + Test Script Recorder

**Interview Questions**

Q) How we can run JMeter in non UI mode using command prompt cmd?

- It is always recommended to open JMeter in non UI mode for performance testing.

jmeter -n –t test.jmx -l testresults.jtl

-n: It specifies JMeter is to run in non-gui mode

-t: Name of JMX file that contains the Test Plan

-l: Name of JTL(JMeter text logs) file to log results

Q) What is distributed testing / Remote testing / Master Slave using JMeter ?

<https://www.youtube.com/watch?v=Ok8Cqc0wipk>

- JMeter consumes a lot of memory and its consumption increases with increase in number of virtual uses as JMeter uses a memory of local machine to generate the users. Generally load of 500 - 1000 virtual users from one machine should be applied. If we want to test with more users then we should apply with multiple machines by distributing number of virtual users (say 300 \* 3). This is distributed testing.

- Distributed testing can be performed if users to be tested from different locations / geographies.

- Couple of points should be taken care, all the systems should be on same network (means can connect to each other) and they have same version of JMeter installed.

- Steps to be followed,

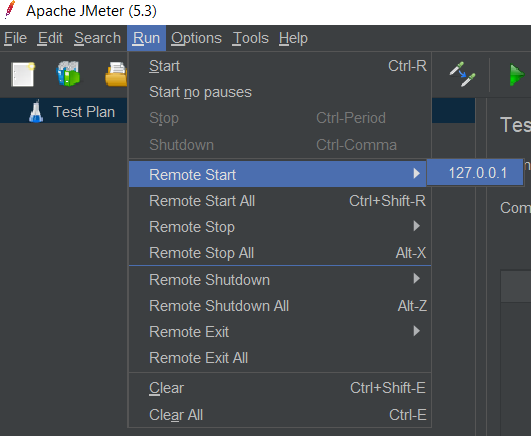
a) Setup Master – In jmeter.properties file (present in bin folder), give IP address of slave systems (multiple IP addresses can be given by separating with comma)

b) Create keystore file – by running ‘create-rmi-keystore.bat’ file.

c) Run ‘jmeter-server.bat’ file on slave / remote machine.

d) Run on remote system using Run -> Remote Start -> Select provided remote / slave IP address

e) If we want to start on all remote machines then select ‘Remote Start All’ option.



Q) How we will do end to end performance testing?

- Firstly need to understand all the requirements clearly like scenarios to be tested, expected load, do we need geographic distribution for load etc.

- Next is Performance Testing Environment as performance testing might hamper on going activities of existing environment and also if we do not use dedicated environment then results may differ.

- Now start creating Test Plan as per the scope of testing. First of all, run it with single user and make sure it works fine. Then increase the load gradually and record the results.

- After the execution compare the results with some baseline results like a specific result should respond in 0.15 sec etc.

Q) In JMeter, how will you increase the load gradually ?

<https://www.youtube.com/watch?v=_YsZn1VwZYY>

* To increase the load gradually, we need to use a plugin ‘Concurrency Thread Group’ instead of our normal ‘thread group’.
* We need to download and import this plugin OR need to install it from JMeter plugin manager.
* It has different options like,
  + Target Concurrency – Max number of users
  + Ramp up time – Total time to send the requests
  + Ramp up step count – In how many steps we have to increase the load
  + Hold Target Rate Time – After reaching max number, how much time further execution should be performed.

