

Gurzu QA Tools

- [Web Testing Tools](#)
 - [Cypress - Introduction](#)
 - [Installation of Cypress](#)
- [Mobile App Testing Tools](#)
 - [Introduction to Appium](#)
 - [Installation Guide: Appium](#)
- [Non-functional Test Tools](#)
 - [Introduction of JMeter](#)
 - [Installation Steps of JMeter](#)
 - [Elements of Jmeter](#)
 - [Use JMeter for Performance and Load Testing](#)
- [Backend Testing Tools](#)

Web Testing Tools

Cypress - Introduction

Introduction

- It is the next generation front end testing tool built for the modern web application. It uses Javascript to write automated tests. Cypress addresses key points from other automation tools. In cypress, node.js is built and also comes packaged as an npm module. It is a faster, easier and more reliable testing tool.
- Before Cypress, all frameworks like Mocha, Jasmine, assertion libraries, and selenium had to be installed with their respective libraries. These all tools have to be integrated for end to end functionality.
- With the use of Cypress, one can do anything for end to end testing as it helps to test functionality and the API also controls HTTP requests. Therefore, Cypress helps in multiple integration as it consists of all tools.

Selenium vs Cypress

One of the main differences between [Cypress.io](https://www.cypress.io/) and [Selenium](https://www.selenium.dev/) is that Selenium executes in a process outside of the browser or device we are testing. Cypress executes in the browser and in the same run loop as the device under test.

Features of Cypress

- Time Travel
- Debuggability
- Automatic waiting
- Network Traffic control
- Consistent Results
- Screenshots and videos
- Cross browser testing

Limitations

- Support limited set of browsers- Chrome, Canary, Electron
- Page Object Model is not supported.
- Tough to read data from files.
- Third Party Reporting Tool integration is limited.

Installation of Cypress

Cypress Installation

- *Download node and npm(Node package manager)*
 - Go to <https://nodejs.org/en/download/> to install node.js and also setup npm

You will get the node.js website and here you can install your node.js as your operating system(Windows, Mac, Linux). Installing node.js by default npm is also in built in node by which on top of this node you can download any kind of javascript frameworks. So node will provide an environment where you can download javascript frameworks so we can work with them.

- *Set NODE_HOME environment variable.*
- *Create a working folder*
 - We have to create a working folder where we are going to create all our cypress test. You can create a cypress folder anywhere in your system. All cypress test cases will be written in the folder you created.
- *Generate package.json file*
 - We need to install cypress but before that we need to generate a package.json file. So on the node environment, if you want to download any software we need package.json so that node will read the package.json file and according to it, it will download required software. We need to specify what kind of software we are going to download in package.json.
 - For example, let's download Cypress where we need to specify the cypress dependencies in the package.json file and then we need to run this npm command which reads the package.json file and downloads the required software.
 - We can generate our package.json file where we can install our dependencies through the command prompt. So go to your cypress folder and open your command prompt.
 - We need to run a basic command called "npm init" which helps to generate package.json file. Once you run a command it will ask you for a package name you can give any name for the package. It will also ask for version, description, entry point, test command, git repository, keywords, author, license you can skip these

point.

- Once it is done, it generates one package file called package.json in the same file location and it's in json format.
- After this json format, it asks for permission to continue and type yes to continue. To view the json file you can simply run a command 'dir', it generates a package.json file which consists of default information. So we need to specify a required software or dependencies which we need to download. Then our node will read a package.json file then it will download the required software or dependencies.

- *Install Cypress*

- Go to cypress official document page (<https://docs.cypress.io/guides/getting-started/installing-cypress#System-requirements>). There are many ways to install cypress. Here in the official site, they are asking to go to the project location i.e. i created "Cypress Automation" as my project.

- `cd /your/project/path`

- `npm install cypress --save-dev`

The command npm install cypress will download cypress and make an entering in the package.json file. This command downloads cypress automatically through online and makes entry in the package.json and `--save-dev` will save that cypress in that package.json.

- If in case you move this project by default this package.json file consists of cypress and just you need to execute this command only. But for the first time you just need to execute a complete command but when you save it and move this project you can only execute `"npm install cypress --save-dev"` to install cypress.
- Once you execute this command then it will start downloading cypress and also it will install cypress by using the package.json file. So node will basically read this file then it will install. But currently we don't have package.json entry so this command automatically downloads cypress and keeps an entry in package.json so then node will read the package.json then it will install cypress.
- With this installation of cypress is completed and you can run cypress in your system.

- *Download Visual Studio Code Editor*

- If you have not installed visual studio code in your system yet then go to <https://code.visualstudio.com/download> and choose as per your operating system. This is the tool where you write cypress scripts. After installation of visual studio code, you can import your cypress working folder which you have created before (cypress automation). With this your project setup is

completed.

Test Runner Component

Once you have open your project in visual studio code, you can see dependencies used in cypress. To work with the test cases, we need test runner component. To launch test runner component, we need to start our cypress within visual studio code.

- Open terminal from visual studio code and run a command in windows
`node_modules/.bin/cypress open`
In Linux, you can use `./node_modules/.bin/cypress open`
- Once you run the command it will launch your test runner.
- After running the above command you get the test runner window where you can see sample test cases which are brought up by cypress.
- As soon as you run this command for the first time you will get some additional folder inside your project. You can see the cypress folder while expanding cypress. You can see some of the important folders which act as a framework provided by cypress by default.
- In the test runner window, we can see all the test cases whatever is already available in the integration/example folder. Once you create one test case in the integration folder, it will automatically popup in the test runner window. When you want to run your test cases you can only run from a test runner. We can create our test cases within the visual studio code and when you are running your test cases, the test runner window should be opened.
- Now lets run some test cases available in the example folder. Lets run the first test cases available in the test runner window. After clicking in this test case, it will start executing the test case opening in the new window.
It will load all the steps from the test case where all the steps are imported and then it will execute each and every step. You can see another window where execution will happen.

Mobile App Testing Tools

Introduction to Appium

Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms. **Native apps** are those written using the iOS, Android, or Windows SDKs. **Mobile web apps** are web apps accessed using a mobile browser (Appium supports Safari on iOS and Chrome or the built-in 'Browser' app on Android). **Hybrid apps** have a wrapper around a "web-view" -- a native control that enables interaction with web content.

Appium Philosophy

Appium was designed to meet mobile automation needs according to a philosophy outlined by the following four tenets:

1. You shouldn't have to recompile your app or modify it in any way in order to automate it.
2. You shouldn't be locked into a specific language or framework to write and run your tests.
3. A mobile automation framework shouldn't reinvent the wheel when it comes to automation APIs.
4. A mobile automation framework should be open source, in spirit and practice as well as in name!

Appium Concepts

Client/Server Architecture

Appium is at its heart a webserver that exposes a REST API. It receives connections from a client, listens for commands, executes those commands on a mobile device, and responds with an HTTP response representing the result of the command execution. The fact that we have a client/server architecture opens up a lot of possibilities: we can write our test code in any language that has a HTTP client API, but it is easier to use one of the Appium client libraries. We can put the server on a different machine than our tests are running on. We can write test code and rely on a cloud service like Sauce Labs & LambdaTest to receive and interpret the commands.

Session

Automation is always performed in the context of a session. Clients initiate a session with a server in ways specific to each library, but they all end up sending a POST /session request to the server, with a JSON object called the 'desired capabilities' object. At this point the server will start up the automation session and respond with a session ID which is used for sending further commands.

Desired Capabilities

Desired capabilities are a set of keys and values (i.e. a map or hash) sent to the Appium server to tell the server what kind of automation session we're interested in starting up. There are also various capabilities that can modify the behavior of the server during automation. For example, we might set the `platformName` capability to `iOS` to tell Appium that we want an iOS session, rather than an Android or Windows one. Or we might set the `safariAllowPopups` capability to `true` in order to ensure that, during a Safari automation session, we're allowed to use JavaScript to open up new windows. See the capabilities doc for the complete list of capabilities available for Appium.

Source: <https://appium.io/docs/en/about-appium/intro/>

Installation Guide: Appium

Installing Appium

Appium can be installed in one of two ways: via **NPM** or by downloading **Appium Desktop**, which is a graphical, desktop-based way to launch the Appium server.

Installation via NPM

If you want to run Appium via an `npm install`, you will need [Node.js and NPM](#) (use `nvm`, `n`, or `brew install node` to install Node.js).

The actual installation is as simple as:

```
npm install -g appium
```

Installation via Desktop App Download

Simply download the latest version of Appium Desktop from the [releases pages](#).

Before moving forward, make sure you have a client downloaded in your favorite language and ready to go.

Starting Appium

Now we can kick up an Appium server, either by running it from the command line like so (assuming the NPM install was successful):

```
appium
```

Or by clicking the huge Start Server button inside of Appium Desktop.

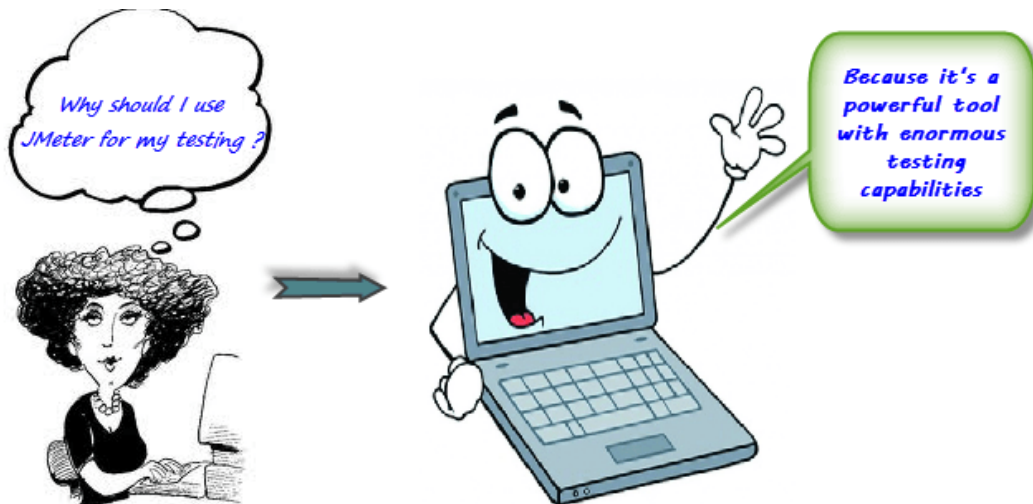
Appium will now show you a little welcome message showing the version of Appium you're running and what port it's listening on (the default is `4723`).

Non-functional Test Tools

Performance, UI, Load etc.

Introduction of JMeter

Apache Jmeter is a popular pure java open-source performance testing tool. We can use JMeter to analyze and measure the performance of web application or a variety of services. Performance Testing means testing a web application against heavy load, multiple and concurrent user traffic.



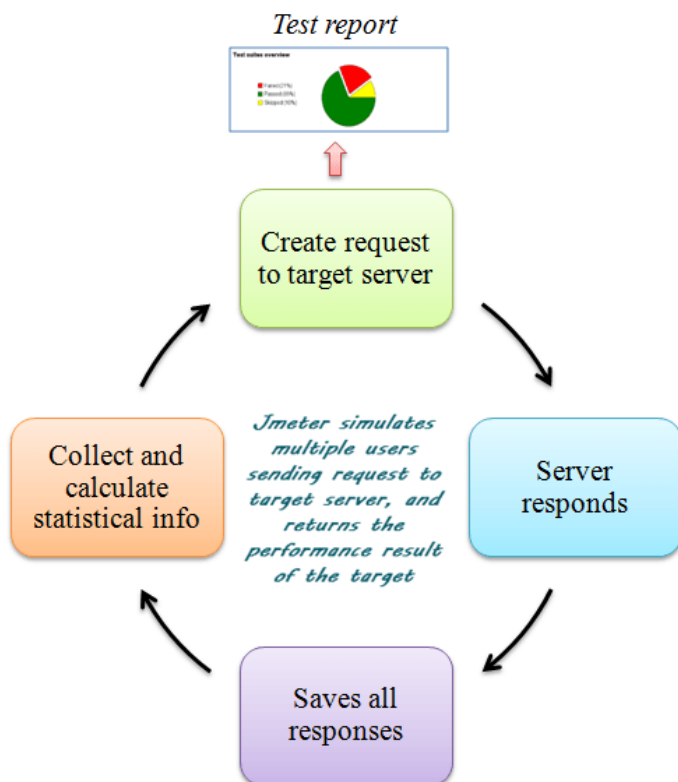
Advantages of JMeter

Jmeter advantages are listed below:

- **Open source license:** JMeter is totally free, allows developer use the source code for the development
- **Friendly GUI:** JMeter is extremely easy to use and doesn't take time to get familiar with it
- **Platform independent:** JMeter is 100% pure Java desktop application. So it can run on multiple platforms
- **Full multi-threading framework.** JMeter allows concurrent and simultaneous sampling of different functions by a separate thread group
- **Visualize Test Result:** Test result can be displayed in a different format such as chart, table, tree and log file
- **Easy installation:** You just copy and run the *.bat file to run JMeter. No installation needed.
- **Highly Extensible:** You can write your own tests. JMeter also supports visualization plugins allow you to extend your testing
- **Multiple testing strategy:** JMeter supports many testing strategies such as Load Testing, Distributed Testing, and Functional Testing.
- **Simulation:** JMeter can simulate multiple users with concurrent threads, create a heavy load against web application under test
- **Support multi-protocol:** JMeter does not only support web application testing but also evaluate database server performance. All basic protocols such as HTTP, JDBC, LDAP,

SOAP, JMS, and FTP are supported by JMeter

- **Record & Playback** - **Record** the user activity on the browser and simulate them in a web application using JMeter
- **Script Test**: Jmeter can be integrated with Bean Shell & Selenium for automated testing.



Installation Steps of JMeter

Steps to Install Jmeter

1. **Install Java :** The first step is to install java. We can download and install the latest version of Java SE Development Kit. [Download Java Platform \(JDK\)](#).

To check whether Java JDK is installed successfully in our system or not

- Go to Terminal and enter java-version

If the Java runtime environment is installed successfully, we will see the output as the figure below

[How to install Jmeter in easy steps](#)

If nothing displays, we have to re-install Java SE runtime environment

2. **Download Jmeter :** We can download Jmeter latest version from [here](#).

Choose the Binaries file (either zip or tgz) to download as shown in the figure below

Apache JMeter 5.4.1 (Requires Java 8+)

Binaries

[apache-jmeter-5.4.1.tgz sha512 pgp](#)
[apache-jmeter-5.4.1.zip sha512 pgp](#)

Source

[apache-jmeter-5.4.1_src.tgz sha512 pgp](#)
[apache-jmeter-5.4.1_src.zip sha512 pgp](#)

3. **Installation :**

Unzip the zip/tar file into the directory where we want JMeter to be installed.

[How to install Jmeter in easy steps](#)

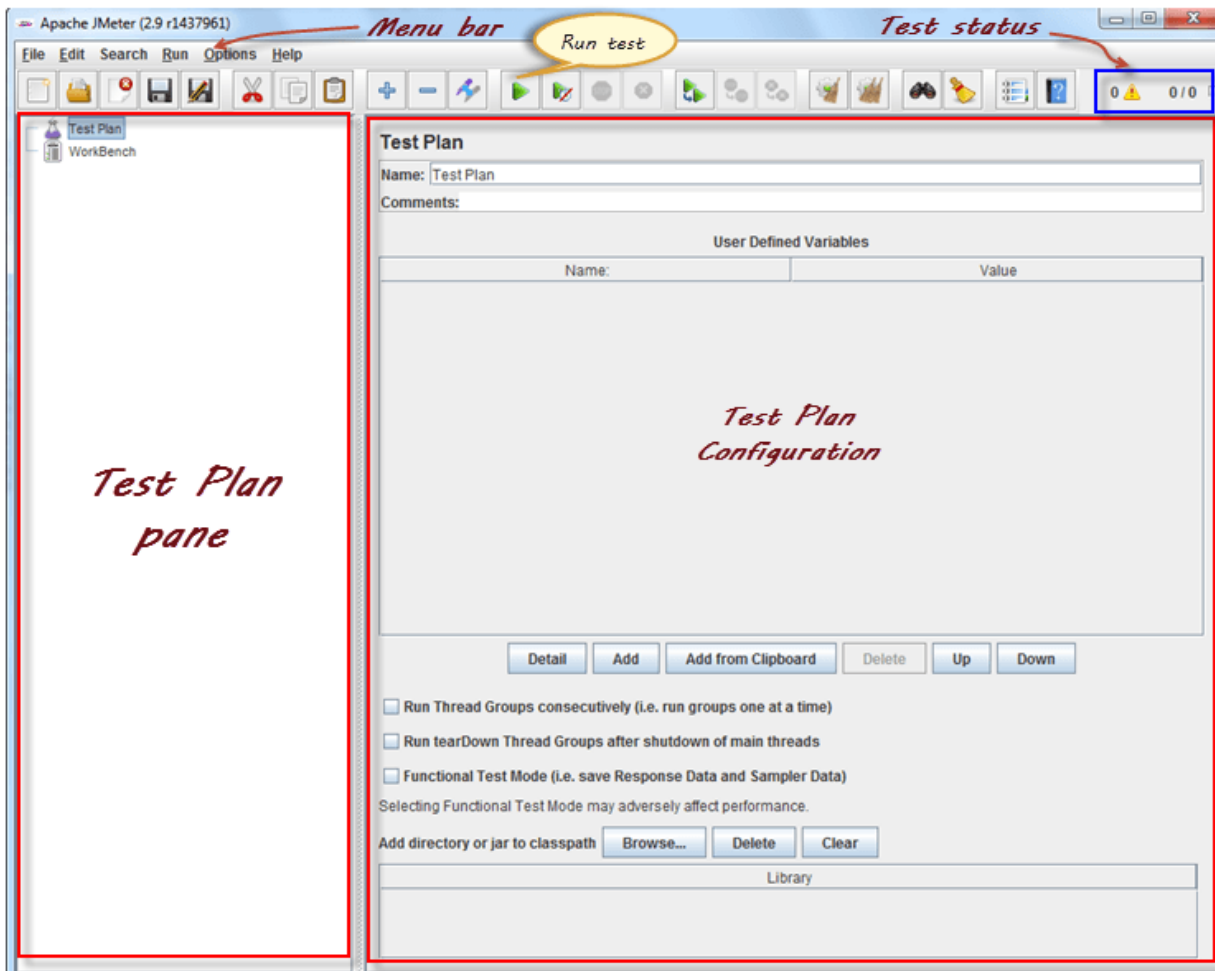
4. **Launch Jmeter :**

We can start JMeter in 3 modes

- GUI Mode
- Server Mode
- Command Line Mode

Start JMeter in GUI Mode

If we are using Window, just run the file **/bin/jmeter.bat** to start JMeter in GUI mode.



Start JMeter in Non-GUI Mode

Start Jmeter in Server Mode

To start the server mode, we run the bat file `bin\jmeter-server.bat` as below figure



Start Jmeter in command line mode

JMeter in GUI mode consumes much computer memory. For saving the resource, we may choose to run JMeter without the GUI. To do so, use the following command options

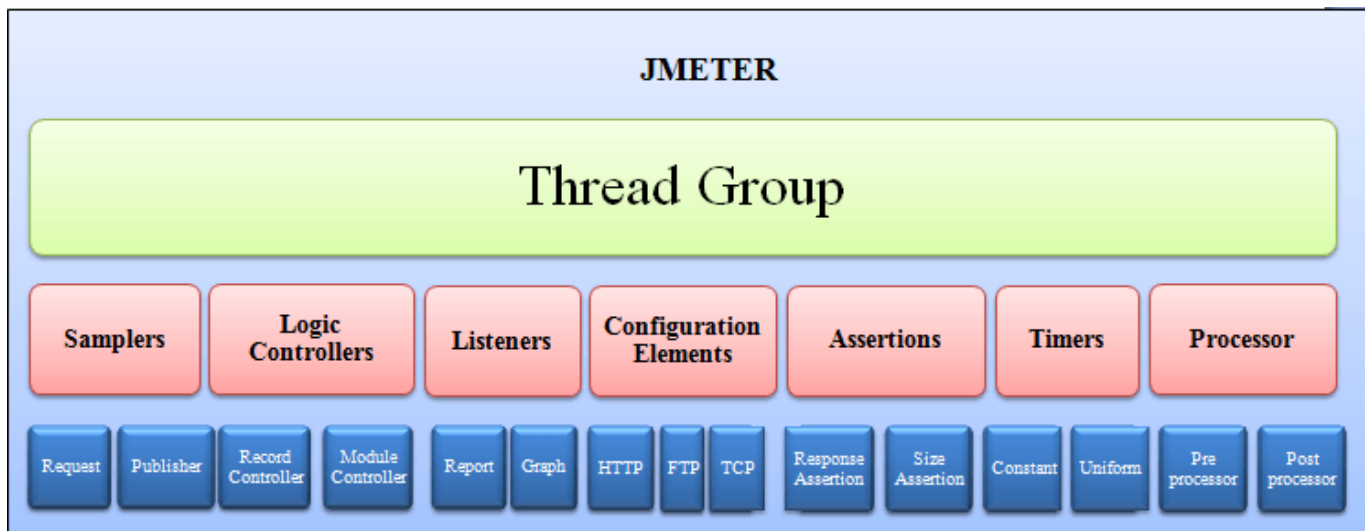
This is a command line example

```
$jmeter -n -t testPlan.jmx -l log.jtl -H 127.0.0.1 -P 8000
```

we simply use `./jmeter.sh` syntax to run `jmeter` in `Linux`.

Elements of Jmeter

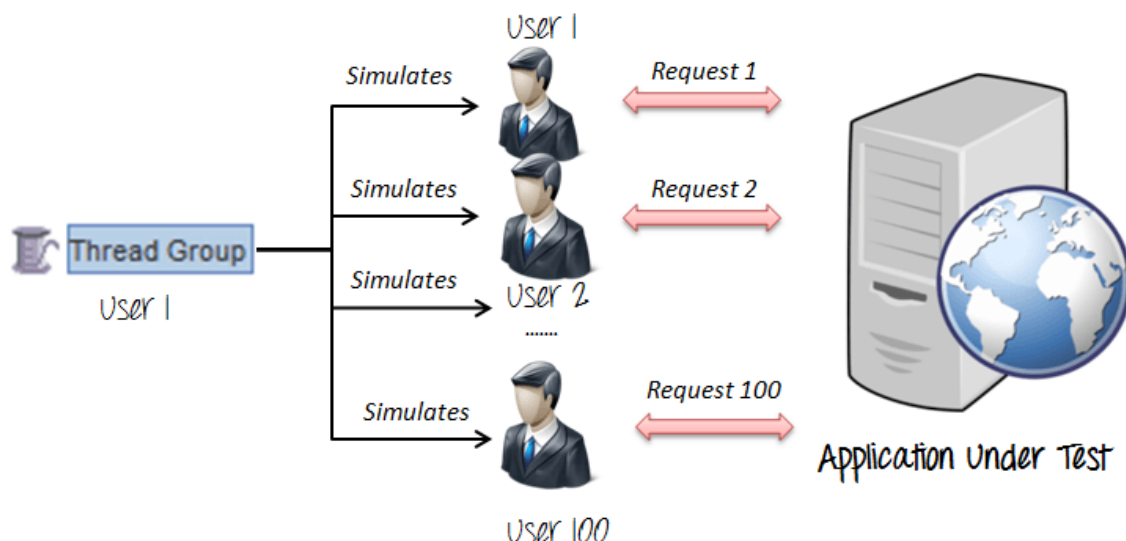
The different components of JMeter are called Elements. Each Element is designed for a specific purpose.



Thread Group

Thread Groups is a collection of Threads. Each thread represents one user using the application under test.

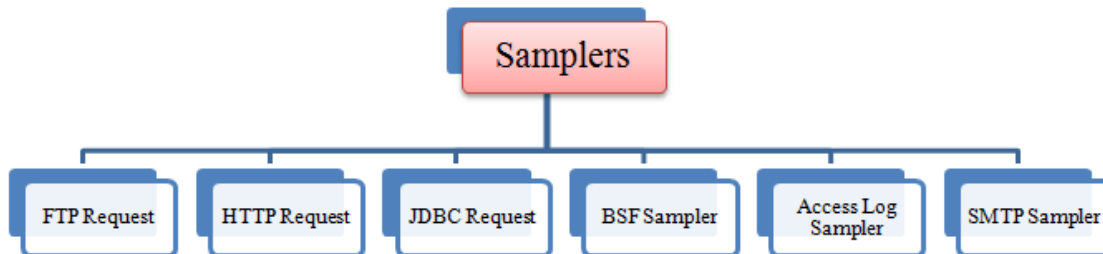
For example, if we set the number of threads as 100; JMeter will create and simulate 100 user requests to the server under test



Sampler

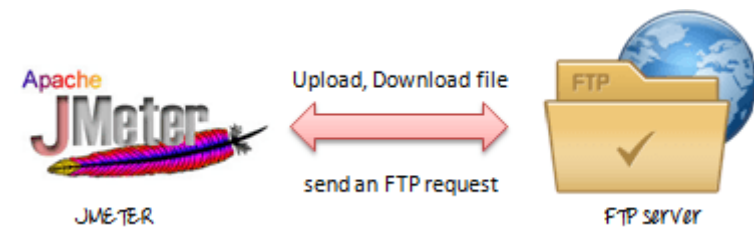
Samplers in JMeter allows JMeter to send different types of requests to a server.

The user request could be FTP Request, HTTP Request, JDBC Request...Etc.



FTP Request

We can send an "download file" or "upload file" request to an FTP server.



For example, if we want to download a file "Test.txt" from an FTP server under test, we need to configure some parameters in JMeter as the figure below

FTP Request

Name:

Comments:

Server Name or IP: *server under test* Port Number: *port to use*

Remote File: *File to download*

Local File:

Local File Contents:

☒ get(RETR) ☐ put(STOR) ☐ Use Binary mode ? ☐ Save File in Response ?

Login Configuration

Username: *FTP account*

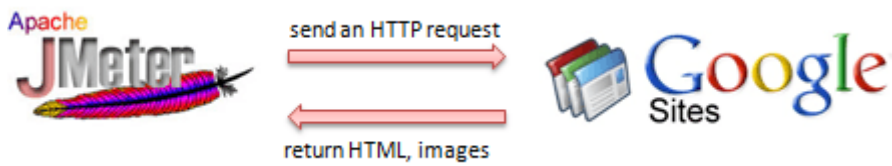
Password:

HTTP Request

One of the most used samplers when configuring a test plan in JMeter is the HTTP Request.

A sampler that lets us send an HTTP/HTTPS request to a web server for load testing. There are different methods the sampler is able to use, like:

- GET
- POST
- PUT
- HEAD
- DELETE
- PATCH
- OPTIONS



JDBC Request

This sampler lets us execute Database Performance Testing. It sends a JDBC Request (an SQL query) to a database.

[Complete Element reference for Jmeter](#)

JDBC Request	
Name:	JDBC Request
Comments:	
Variable Name Bound to Pool:	
Variable Name:	
SQL Query	
Query Type:	Select Statement
Query:	<div>select test_result from test_tbl where id = 1</div> <div><i>sql query</i></div>
Parameter values:	
Parameter types:	
Variable names:	
Result variable name:	

BFS Sampler

This sampler allows you to write a sampler using BFS scripting language.

BSF Sampler

Name: BSF Sampler *Name of sampler*

Comments:

Script language (e.g. beanshell, javascript, jexl)
Language: *Name of scripting language*

Parameters to be passed to script (=> String Parameters and String []args)
Parameters: *List of parameters to be passed the script*

Script file (overrides script)
File Name: *Name of script file* Browse...

Script (variables: ctx vars props SampleResult sampler log Label FileName Parameters args[] OUT)
Script:

Access Log Sampler

This sampler allows us to read access logs and generate HTTP requests. The log could be image, Html, CSS..

Access Log Sampler

Name: Access Log Sampler *Name of sampler*

Comments:

Default Test Values
Server: *Server IP address*
Port: *and port*

Parse Images: False

Plugin Classes
Parser: org.apache.jmeter.protocol.http.util.accesslog.TCLogParser

Filter (Optional): Undefined

Log File Location
Log File: *Location of log file*

SMTP Sampler

This sampler is used to send email messages using the SMTP protocol. If we want to test a mail server, we can use SMTP sampler.

SMTP Sampler

Name: SMTP Sampler

Comments:

Server settings

Server: *Mail server address and port*

Port: (Defaults: SMTP:25, SSL:465, StartTLS:587)

Mail settings

Address From:

Address To: *Mail settings*

Address To CC: *to send an email*

Address To BCC:

Address Reply-To:

Auth settings

☐ Use Auth *security setting for sending mail* Username:

Password:

Security settings

☒ Use no security features ☐ Use SSL ☐ Use StartTLS

☐ Trust all certificates ☐ Use local truststore ☐ Enforce StartTLS

Local truststore:

Message settings

Subject: ☐ Suppress Subject Header

☐ Include timestamp in subject

Message: *Message settings: email subject + message body + file attachment* ☐ Send plain body (i.e. not multipart/mixed)

Attach file(s):

Listeners

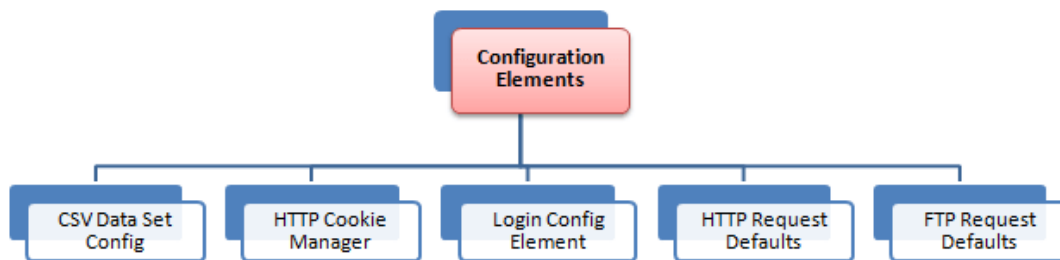
A listener is a component that shows the results of the samples. The results can be shown in a tree, tables, graphs or simply written to a log file. The following is the list of all Listeners in Jmeter

- Graph Results
- Spline Visualizer
- Assertion Results
- Simple Data Writer
- Monitor Results
- Distribution Graph (alpha)
- Aggregate Graph
- Mailer Visualizer
- BeanShell Listener
- Summary Report
- Sample Result Save Configuration
- Graph Full Results

- View Results Tree
- Aggregate Report
- View Results in Table

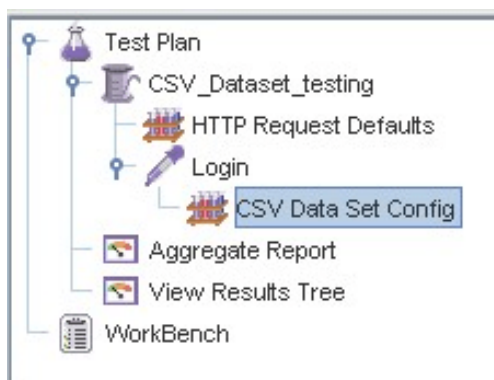
Configuration Elements

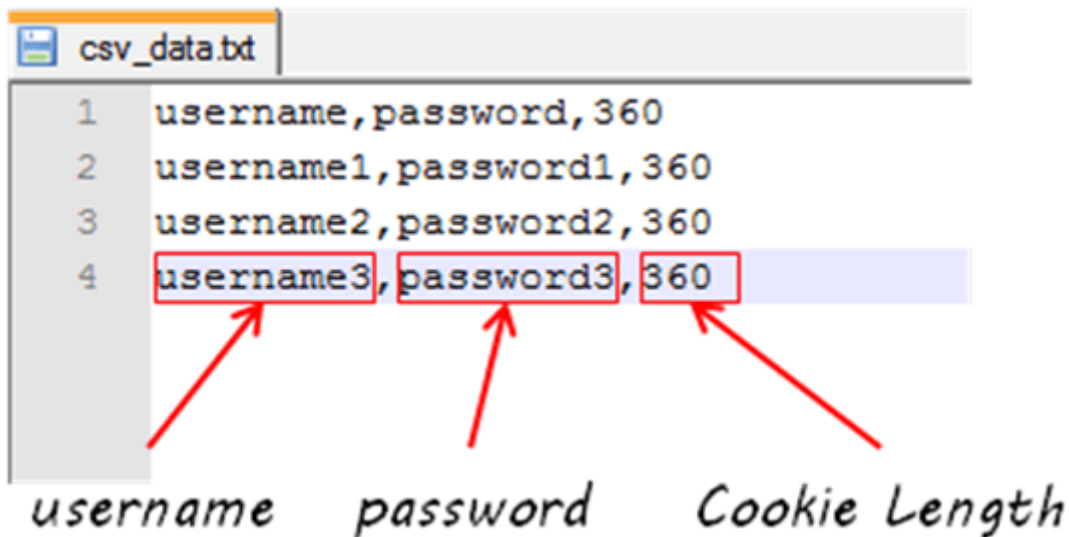
Configuration Element enable us to declare variables, so Samplers can use data through these variables.



CSV Data Set Config

The “CSV Data Set Config” enables **using CSV files as an external data source**, where we can keep unique user data like names, emails and passwords. With the help of this config element, JMeter is able to read the CSV file line by line, and then use split parameters to allocate different values for different threads. It is CSV Data Set Config which is used to read lines from a file, and split them into variables.





HTTP Cookie Manager

The cookie manager stores and sends cookies just like a web browser. If we have an HTTP Request and the response contains a cookie, the Cookie Manager automatically stores that cookie and will use it for all future requests to that particular web site. Each JMeter thread has its own "cookie storage area"

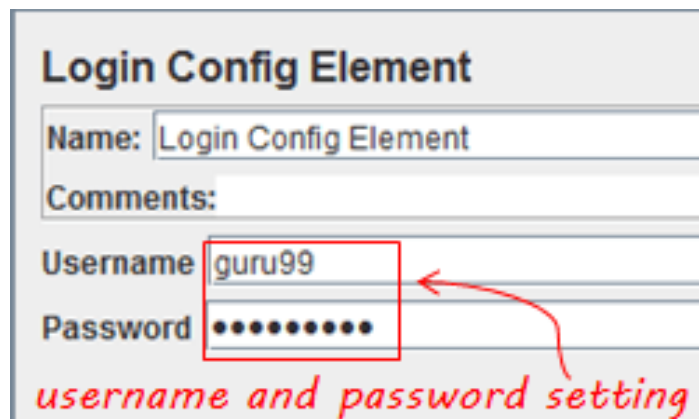
HTTP Request Defaults

'HTTP Request Defaults' is a **very basic and key element of JMeter**. This config element is used when all requests in the JMeter script are sent to the same server. You can add a single HTTP Request Defaults element under Test Plan with the proper server name or IP address in the field 'Server Name or IP'.

HTTP Request Defaults			
Name: HTTP Request Defaults			
Comments:			
Web Server		Timeouts (milliseconds)	
Server Name or IP:	google.com	Port Number:	80
		Connect:	1000
		Response:	2000
HTTP Request			
Implementation:		Protocol [http]:	
		Content encoding:	

Login Config Element

The simple definition of 'Login Config Element' component in JMeter is the element which is **used as a global component to add or override the credential** in the following Samplers added to the test plan. This element is specially used for authentication.



Login Config Element

Name: Login Config Element

Comments:

Username guru99

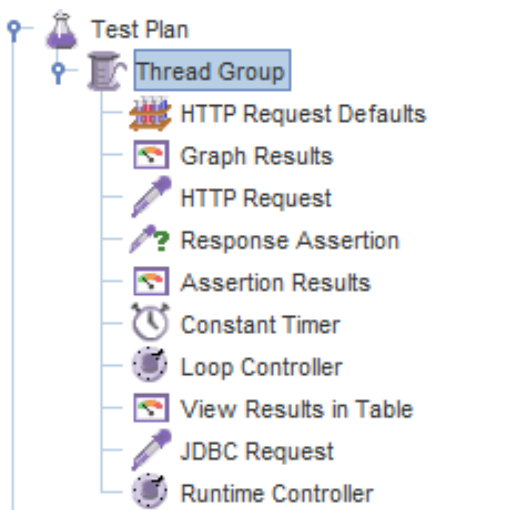
Password ●●●●●●●●

username and password setting

Use JMeter for Performance and Load Testing

Test Plan

A test plan describes **a series of steps JMeter will execute when run**. It stores all the elements (like Thread Group, Timers etc)



Load Testing

Jmeter for load testing is a crucial tool that determines whether the web application under test can satisfy high load requirements or not. It helps to analyze overall server under heavy load.

Performance Testing

Jmeter for performance testing helps to test both static and dynamic resources, helps to discover concurrent users on website and provides variety of graphical analysis for performance testing. Jmeter performance testing includes load test and stress test of web application.

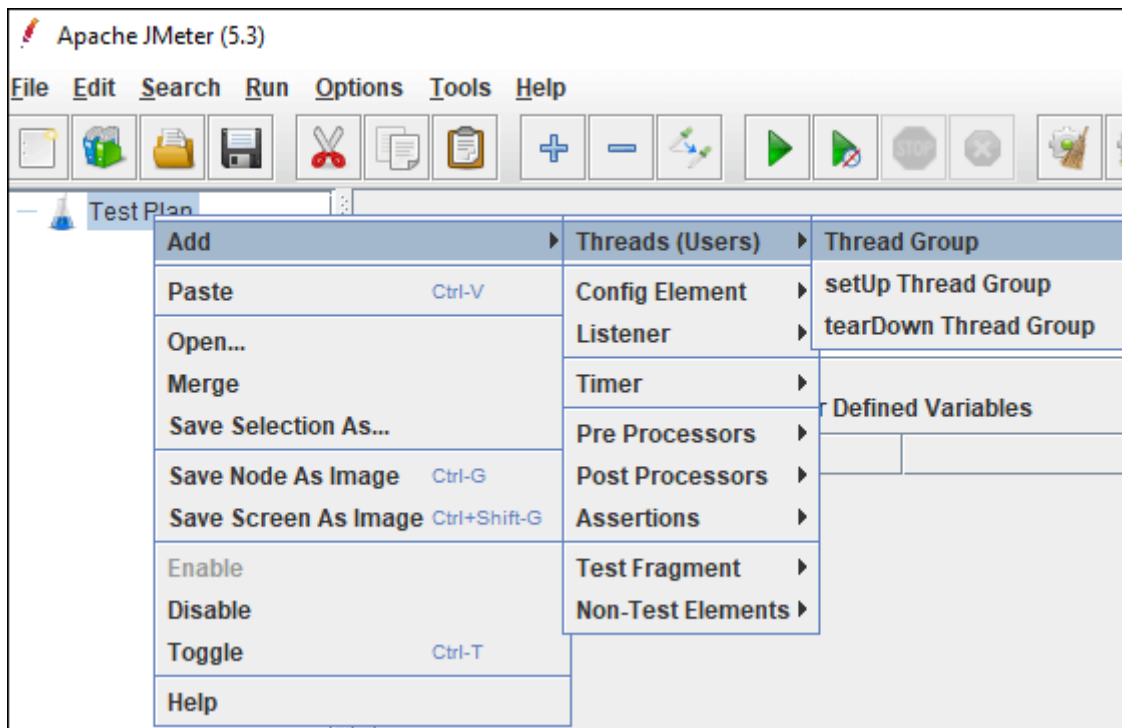
Stress Testing

Every web server has a maximum load capacity. When the load goes beyond the limit, the web server starts responding slowly and produce errors. The purpose of the Stress Testing is to find the maximum load the web server can handle.

Create a Performance Test Plan in JMeter

Add Thread Group

1. Launch JMeter
2. Select test Plan on the tree
3. Right Click on the Test Plan and add a Thread Group.



Enter Thread properties:

Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

☒ Continue

Thread Properties

Number of Threads (users): 100

Ramp-Up Period (in seconds): 100

Loop Count: ☐ Forever 10

☐ Delay Thread creation until needed

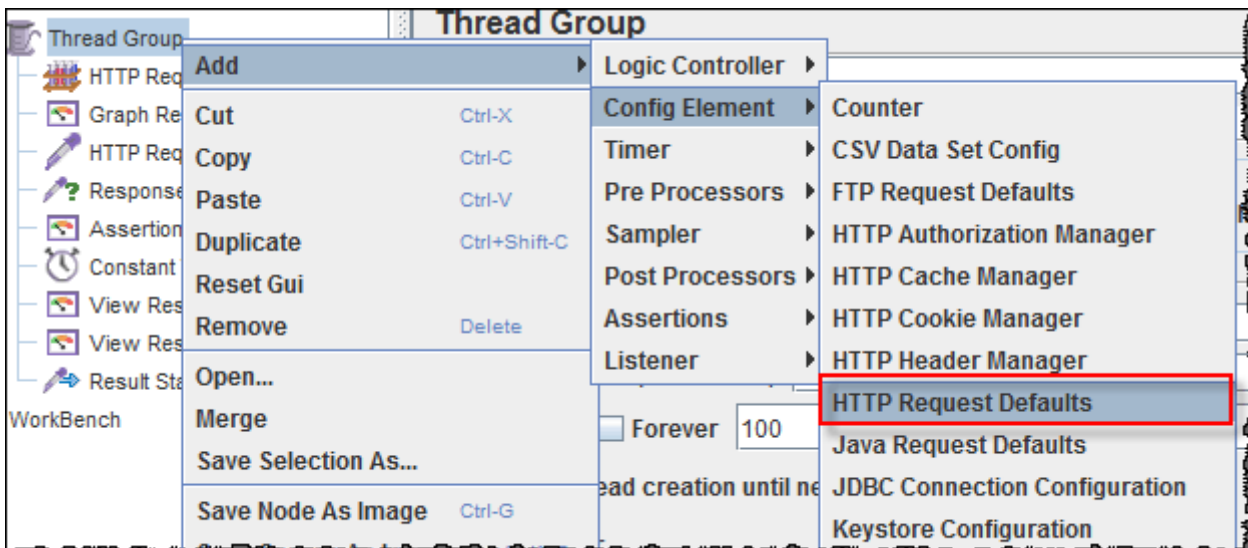
☐ Scheduler

- **Number of Threads:** 100 (Number of users connects to the target website: 100)
- **Loop Count:** 10 (Number of time to execute testing)
- **Ramp-Up Period:** 100

Adding Sampler

- **HTTP Request Default**

Right clicking on the Thread Group and selecting: **Add-> Config Element -> HTTP Request Defaults**



HTTP Request Defaults

Name: HTTP Request Defaults

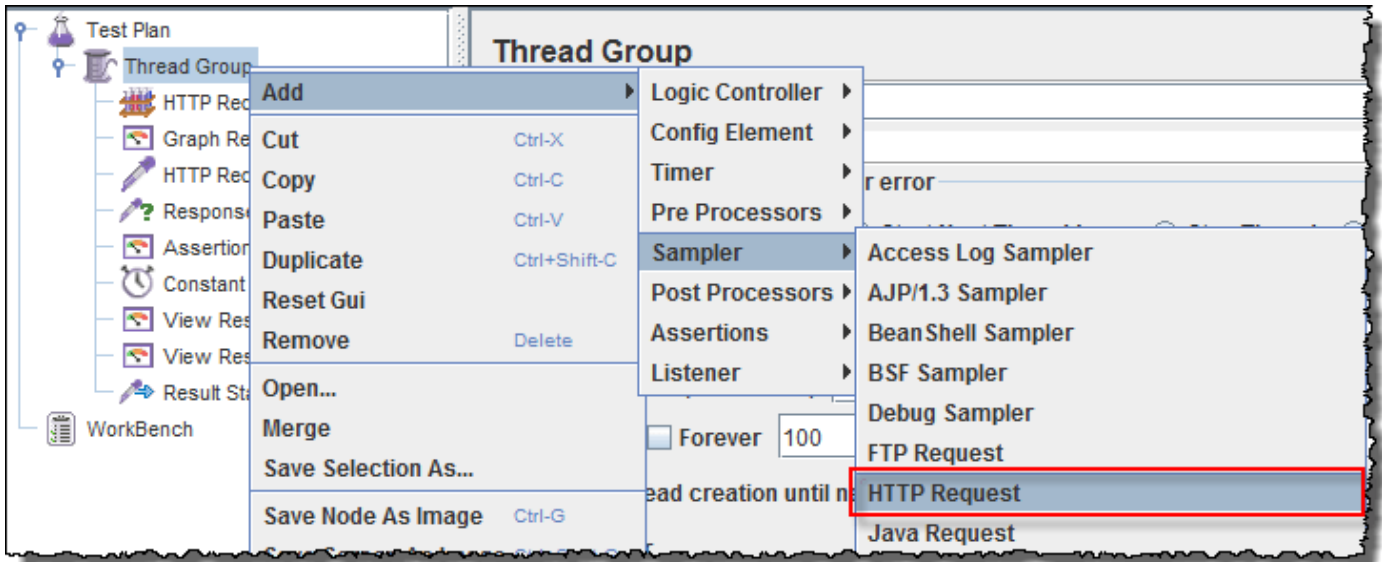
Comments:

Web Server

Server Name or IP: www.google.com Port Number: 80

- **HTTP Request**

Right-click on Thread Group and select: **Add -> Sampler -> HTTP Request**.

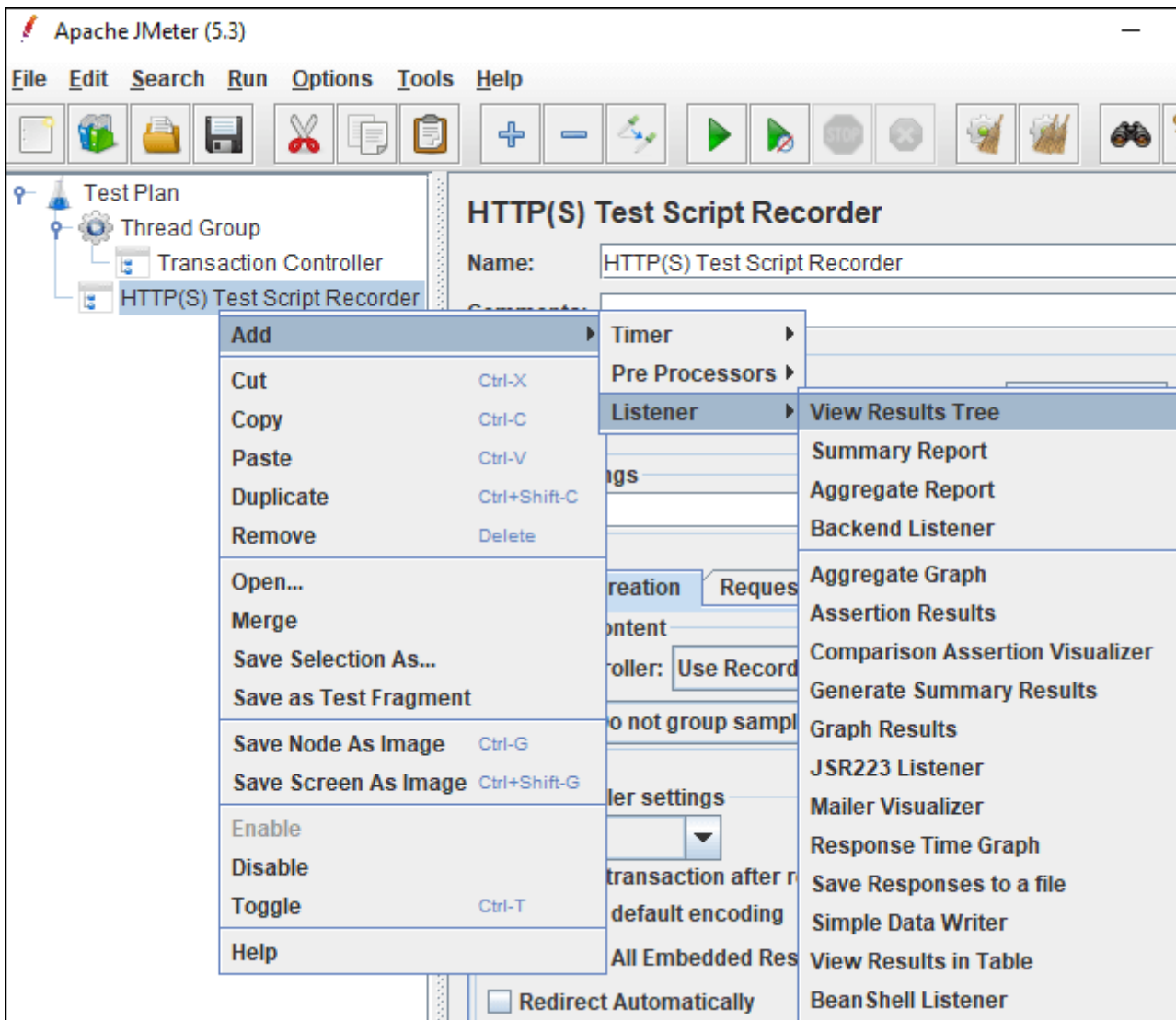


The path filed indicates which URL request we want to send to the google server

A screenshot of the 'HTTP Request' configuration dialog box in JMeter. The dialog has several sections: 'Name' (set to 'HTTP Request'), 'Comments', 'Web Server' (with 'Server Name or IP' field), and 'HTTP Request' (with 'Implementation' dropdown and 'Protocol [http]' field). The 'Path' field is highlighted with a red rectangular border. Below the 'Path' field are three checkboxes: 'Redirect Automatically' (unchecked), 'Follow Redirects' (checked), and 'Use KeepAlive' (checked). At the bottom, there are tabs for 'Parameters' and 'Post Body'. The 'Parameters' tab is active, showing a table with columns for 'Name' and 'Value'. The 'Send Parameters' checkbox is also visible.

- **Adding View Result Tree as Listener**

Right click Test Plan, **Add -> Listener -> Graph Results**



• Run Test and get the Test Result

Press **the Run** button (Ctrl + R) on the Toolbar to start the software testing process. You will see the test result display on Graph in the real time.

Backend Testing Tools

Tools used for Backend Testing.