**JMETER BLOG:**

**Table of contents:**

# What is Load Testing

# Open source tools available for the load testing

# Brief introduction to the Jmeter platform

# Installation

# Running jmeter

# How to Create A Thread group

# How to Add Samplers [http requests (User Login/get request)]

# Add a HTTP header manager

# How to add a json extractor (capture the access token provided)

# Add debug sampler (Assure the access token captured)

* **Adding Listeners to the thread groups**

# How to Use access token variable in any other http request

# User defined variable request and define URL on those variables

# Use variables on the server name field on the http requests

# Use of multiple Thread groups

# Add another request in the second thread group using the token variable of first thread group (initialize to make it fail)

# How to user login requests into test fragment group.

# Include module controller into every thread group and select test fragment

# How to Image Upload

# How to use CSV upload

# USE of Counter to add one plus value in each iteration

# Extract other values from the response like vehicle number, id and use them in another http request to edit

# Report Generation and running through command line

# Load Testing

# Load testing is performed to determine a system's behavior under both normal and anticipated peak load conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation.

# Open source Load testing tools available online

# Apache Jmeter(JMeter is the most popular open source load testing too)

# The Grinder

# Gatling

# Taurus

# Brief Introduction to the Apache jmeter:

# JMeter is an open source desktop Java application that is designed to load test and measure performance. It can be used to simulate loads of various scenarios and output performance data in several ways, including CSV and XML files, and graphs. Because it is 100% Java, it is available on every OS that supports Java 6 or later.

# Prerequisites for installation and launch

# Install the latest 64-bit JRE or JDK. This is needed as JMeter is a pure Java application.

# Launch jmeter through .bat file or jar file from inside the bin folder.

# You should now see an empty test plan (like in the image below):

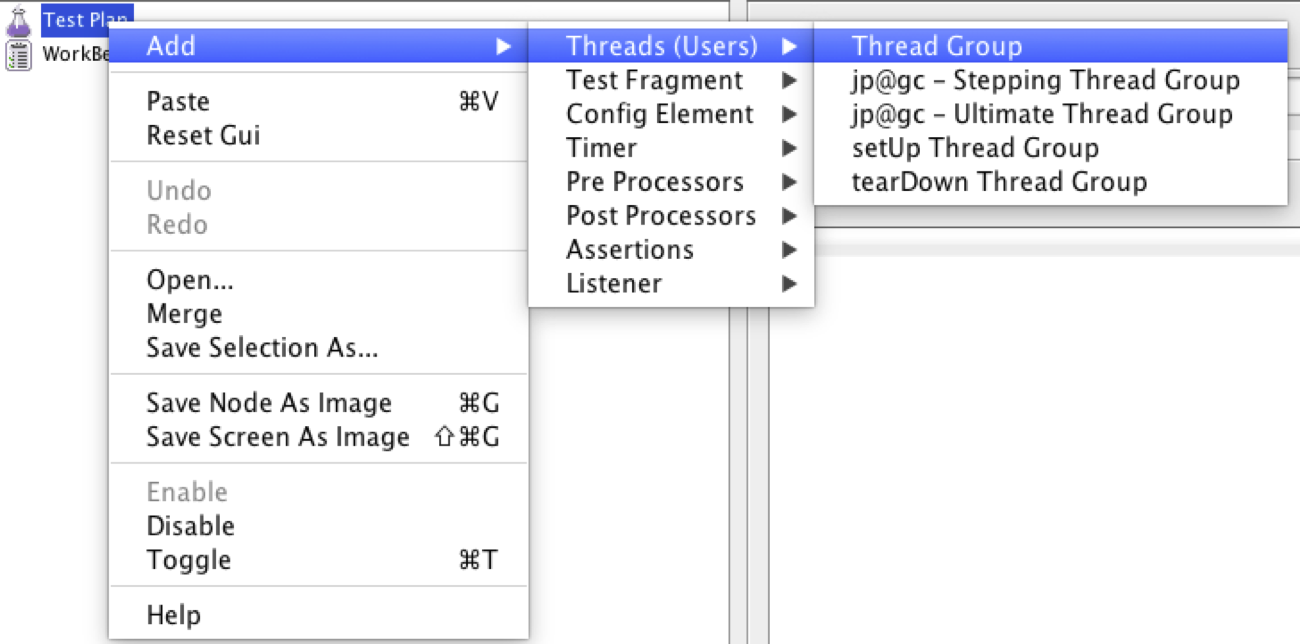
# 

# Test plan:

# In your test plan, you should specify the overall settings and outline the steps that you want JMeter to execute while it runs

# How to Create A Thread group

The first step for us to perform before sending any http request would be to create a Thread Group first. The Thread Group tells JMeter the number of users you want to simulate, how often the users should send requests, and how many requests they should send.

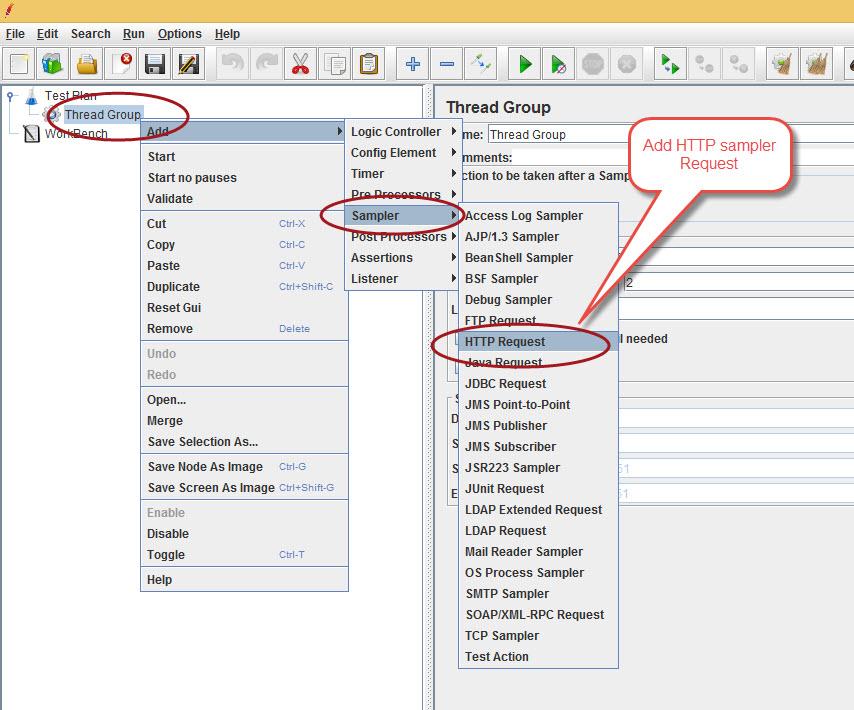


**How to Add Samplers**

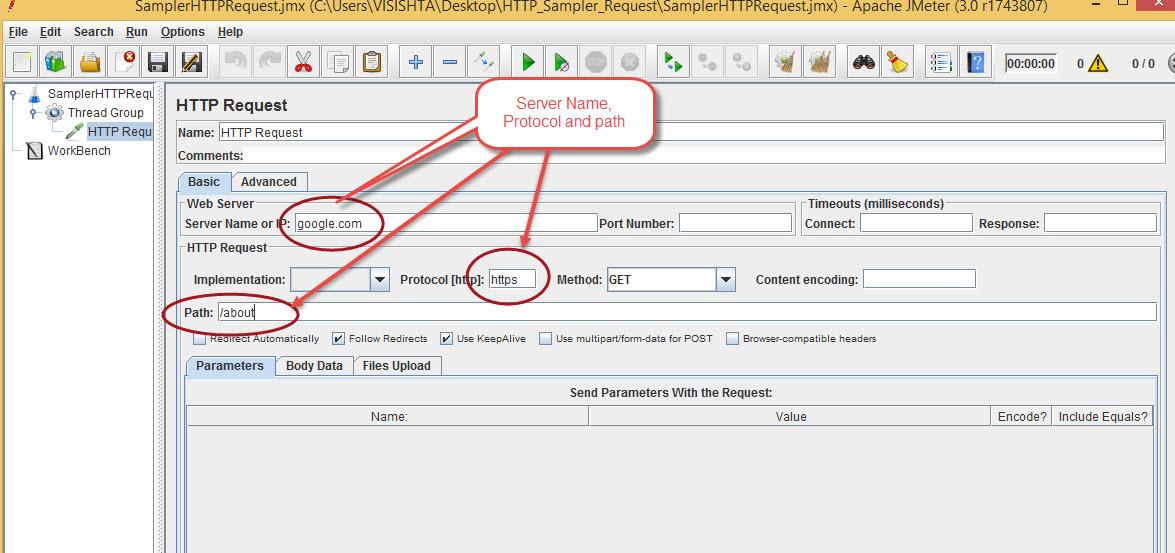
There are various samplers that we can use under the Thread group like as FTP request, HTTP request, Debug sampler, java request, JDBC request etc. and so on, we will be discussing about them as we will be using them in our test. The one we are going to be using right now is Http request.

**HTTP Request: HTTP request**

Add HTTP Request Sampler for this thread. Right Click on Thread Group - Add -> Sampler -> HTTP Request

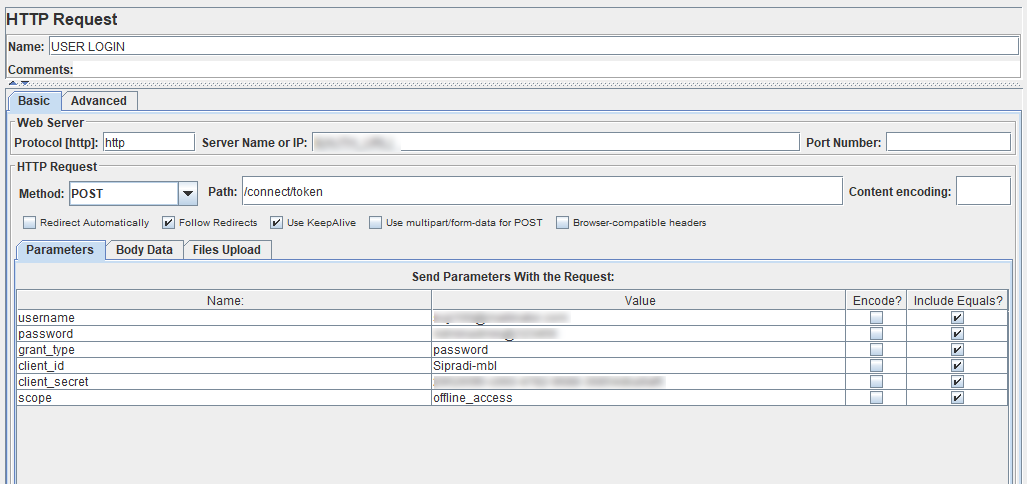


Now you will see an empty http request page, where you will be supposed to fill up the following data:



Let’s learn how to make a simple Login post request for those api that requires login credentials of any user for the api to be accessed.

Let’s look at the following Post request made to an website with all the parameters required:



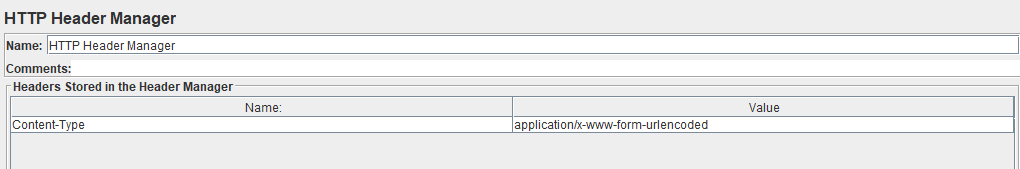
From the image above, what is to be noted is Fields like Protocol, server name or IP, Path. Method etc. are required fields to access any kind of rest api’s. The parameters being sent are the ones that are required fields to access the api.

**Add a HTTP Header Manager:**

In some cases, it's impossible to go on without HTTP Header Manager i.e. if application under test expects that request Content-Type will be "application/json" and if it is not the case the request won't be served. To add Http header manager, you need to follow the following procedure.

Thread Group-> Add->Config Element-> Http Header Manager

We have added HTTP header manager for our login api.



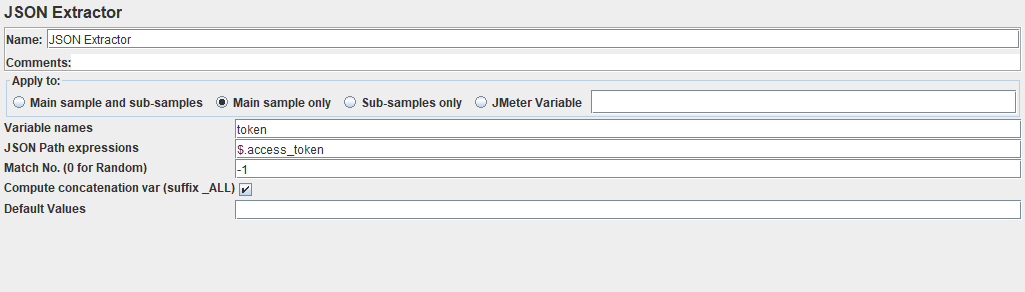
**Use of Json Extractor**

Logging into a website is good and but that is not it for every time we want to access few details of that specific user. Using the login post method every time we want to perform some activity is not a good idea. Here, what can be done simply is once you have Logged into a website, just capture the token provided back to you. And use that token as an authorization every time you would like to access data of the following user. How simple is that? Now let’s think about how to do that:

We have a sampler called JSON Extractor in Jmeter which will help us to capture the Token provided.

**How to Add that?**

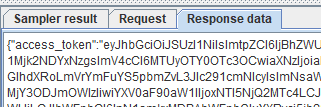
Thread Group->Add->Post Processor->JSON Extractor



All you have to fill inside the JSON Extractor Sampler is:

1. Name: You don’t need to change the name. You can just leave it as it is.
2. Variable Names: This implies the name of the variable you would like to access later. That would be your choice and could be anything. Note: You may also add multiple variable names for multiple data extraction from the response. Just add “;” in between them.
3. JSON Path expressions: Now this could be a critical stage for you to specify the exact name from the response. It depends upon in what way is your api providing you the response.

For a response like this,



The variable is to be set as: “**$.access\_token”.**

Note: The response like this could be checked by adding listener “View results tree” and running them under every thread group. We will be discussing about these later.

1. Match no (0 for Random): for now, leave it as -1.

So that will be all for a successful token extraction provided from any api.

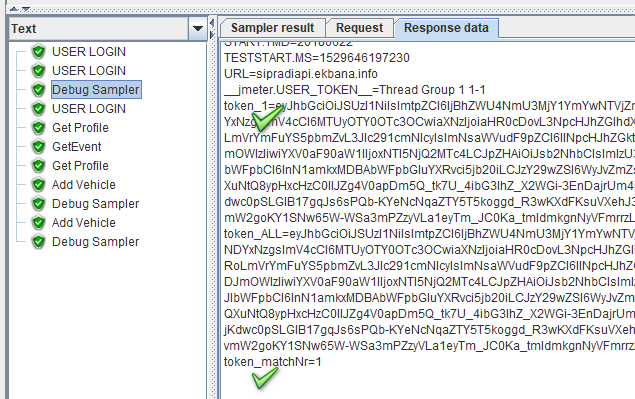
Do you want to make sure that you have successfully extracted the given token under your defined variable or not?

Jmeter has provided another sampler for you to be assured either. We can do this by using **Debug Sampler.**

**Thread Group-> Add-> Sampler-> Debug sampler**

**Note:** The debug sampler should be placed right below the User Login HTTP Request.

Your token has been extracted if you see the following response in the debug sampler:

****

In the image above, you can see the token has been placed inside the variable **token\_1.** The variable you did set before was **token**, but jmeter will add \_1 on any variable name been set on its own. You will have to access this token with the name **token\_1.**

The second pointed mark on the image assures that you have don’t the extraction successfully as it is displaying **token\_matchNr=1.**

Now we can use this variable token anywhere the authorization is required. Before that, lets learn how to add listeners to the thread groups first. It is very important one and required to view result in every thread group.

**Adding Listeners to the thread groups:**

Listeners are to verify the result of the test plan and they are of different kinds. The ones we will be using often are,

**Aggregate Report**

**View Results tree**

**View results in table**

**Aggregate Graph**

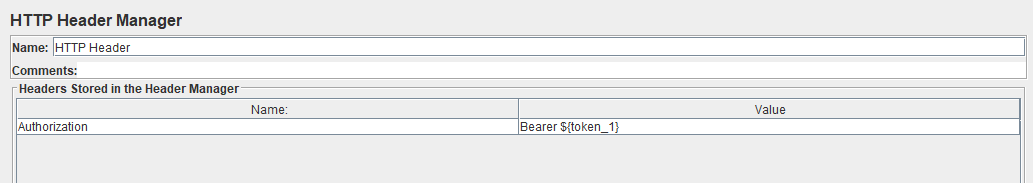
The listener we have been checking response was inside the **View Result Tree listener**. Responses will be available only after running the tests.

Now let’s get back towards how we use the extracted variable of the tokens. Now that we have extracted the token inside a variable we want, we are going to use it somewhere, right?

# How to Use access token variable in any other http request

For any other Http request which are required to have an authorization, we will be using the variable that has the token through the Http Header Manager.

Just add the HTTP header manager below the Http request that requires an Authorization and simply add a field to use the variable for the authorization field.



With this Variable Token\_1, the whole token that we have store before will be sent to the Authorization field. If you get the list of data from the respective http request, your token has been successfully sent, else you might have missed some point.

Note: The variable will always be accessed by using a $ sign in inside a pair of Curly braces in jmeter. i.e, **${variable\_name}**

**How to use User Defined Variable**

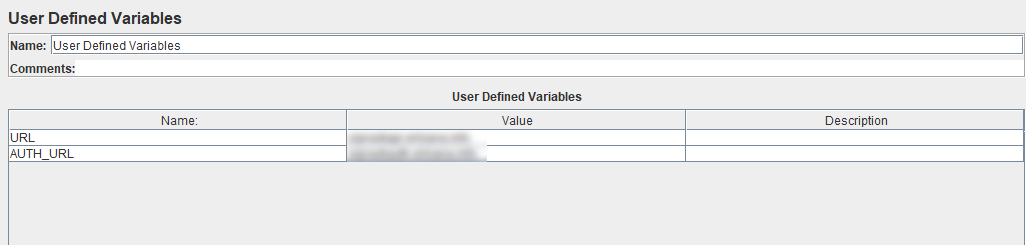
User defined variable is a config element that we can set under any a test plan or under any Thread Groups. What it is used for is we will simply define any URL’S or any other repeating values inside a variable name so that we don’t have to write those whole values every time and every where we want to use it. And also so that if the value changes, we don’t have to go to every request and change them one by one where ever we are using it. So this is just a smart way of doing it to store values inside a variable and use that variable against any them.

For instance, Let’s add a User Defined Variable Config Element and store URL’s inside the variable.

**To add a User Defined Variable:**

Test plan/Thread Groups-> Add-> Config element->User Defined Variables

You may add this inside the Testplan or thread group . Just the difference is, From the Testplan, it will work globally and from inside the Thread Group, it will work locally.

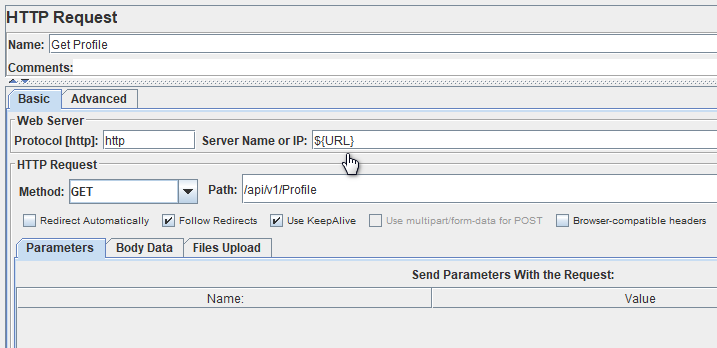


Add the URL you are using under the values. Now you may simply Access URL as ${URL} where ever you will be needing it.

You will now have to change the URL path on this page only, not everywhere you have used them.

# Use variables on the server name field on the http requests

Only change you need to make to use this variable will be:

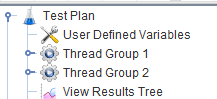


**How to Create multiple Thread Groups:**

You can create thread groups multiple times for few of the good reasons:

* To let every thread group work for different type of function
* To run each kind of functions (Thread groups) in an independent way.

For eg, I would like to perform the First thread group to get the user profile and the second Thread group to upload profile images of those users. Now If I want to get the user profile more often than to upload the image to it, then this is perfectly good idea to use the multiple thread groups. In that case, I would Put the thread of thread group 1 to 50 times and thread group 2 to 10 times.



Now, as we have created two thread groups, if we want to access the variable of first thread group from the second thread group, then it will fail, Because the variable was only set Locally and it cannot be used in another Thread group. For that case, we can add the User login and Token extractor in one module and then call that module in every Thread group instead.

Here we have a module that we can include in every thread group. That is a test fragment. Just follow the steps below to make it work.

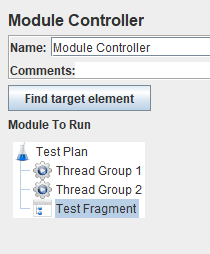
* Add a test fragment from the Test Plan.

**Test Plan-> Add-> Test Fragment->Test Fragment**

* Drag the User Login Http request of Thread 1 to the Test Fragment module along with the Http Header Manager and JSON Extractor inside it.
* Now how do we include this segment inside every thread Group is by using Module Controller inside each Thread Group.

Thread group->Logic Controller-> Module Controller

* And from inside the module controller, do select the test fragment from the available options.

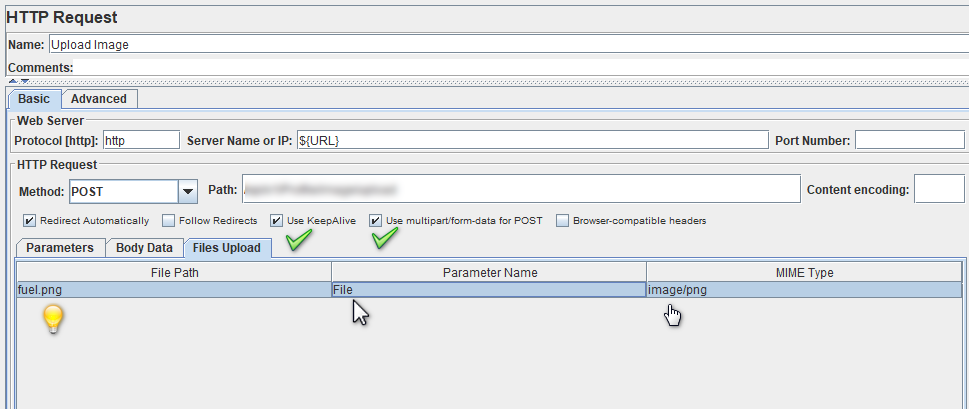


* Now we have enabled login and token variable inside every thread groups.

**How to UPLOAD IMAGE file**

While you need to test the upload image functionality of your app, you can simply do it through jmeter by following few simple steps:

First, make sure that your HTTP request is set to “POST” and the "Use multipart/form-data for POST" box is checked.



You need to consider few points before you send the file path for your file upload.

**File Path:** If you test locally, make sure you give the full file path, including the file name. If your jmx file and image file is in the same folder or location, you don’t need to specify the path.

**Parameter Name:** This must be the exact “name” attribute as it appears in the page source, otherwise the file won’t have a ‘destination.

**MIME Type:** It goes without saying, but make sure you enter the correct one. If not, the upload process won’t work properly.

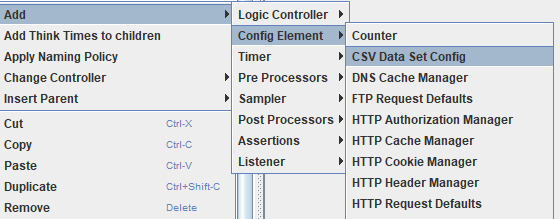
Here’s a list of common Image Mime Types:

* image/gif: GIF image
* image/jpeg: JPEG JFIF image
* image/png: PNG image
* image/svg+xml: SVG vector image
* image/tiff: TIF image
* image/vnd.djvu: DjVu image and multi-page document format

# How to use CSV upload

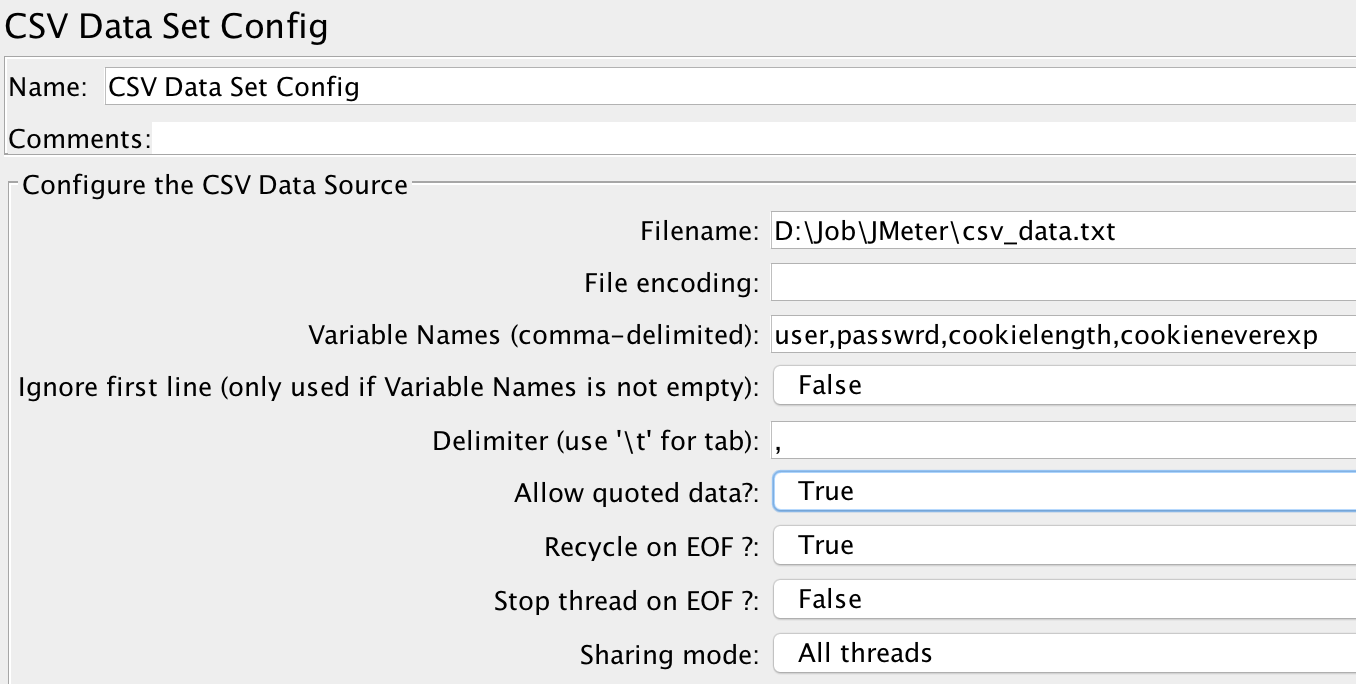
[JMeter](https://www.blazemeter.com/jmeter-load-testing?utm_source=knowledgebase&utm_medium=kb&utm_campaign=using-csv-data-set-config), an open source load testing tool, has an element that allows you to use external data sets in a CSV format. This element is called the “CSV Data Set Config”. The CSV Data Set Config is used to read lines from a file and to split them into variables.

You can add a “CSV Data Set Config” file by following the following steps:

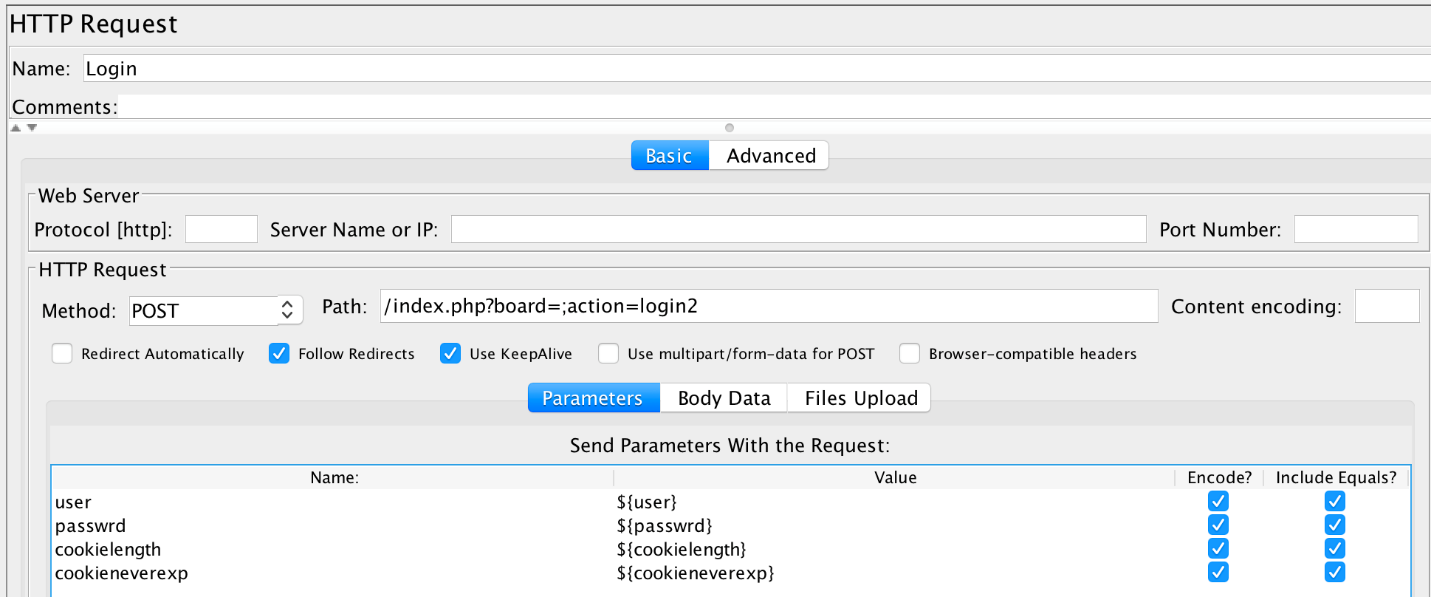


You will need to fill in at least three values in this screen below:

* **Filename:** if your file is in the /bin directory, this can just be the filename. If it’s somewhere else, use the full path to the file.
* **Variable names:** this usually refers to a “column name” in a spreadsheet.
* **Delimiter:** a comma is the default delimiter, but if your file uses tabs, this is the place to mention that.



You need the https request that will use this file and values.

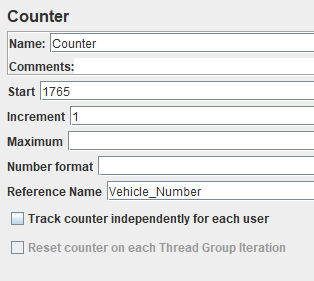


In the request, you can see the table with the variables. There are four variables, the same quantity as in 'CSV Data Set Config'. As you can notice, the value field has the same name as variable in the 'CSV Data Set'. The construction ${….} means that this is a variable and not an absolute value.

The values those will be passed on this http request will be visible on the request sample of “View Results Tree”.

**Using the counter for the increment of values**

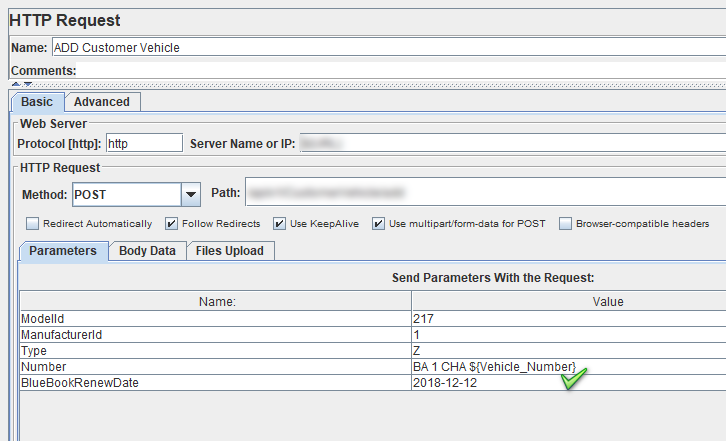
A Counter is an element to increase the required value (might be 1 or more) of a specific field for each iteration. Counter element is generally used for the increment of the integer value.



You must fill up at least 3 of the fields in this element to get the counter working as required.

* Start: Starting value of this field could be any integer value that you prefer
* Increment: In this field
* Reference Name: Here you may provide any variable name that you will be using while accessing its values.

For using the Reference Variable on the HTTP request, you may simply access as you access any other variables. An example is shown below to use a reference variable.



Now, discussing about the functionality of the counter hereby, it sends the exact provided value for the given variable on the first iteration and for every succeeding iteration the value will be increased by 1 as the increment filled has 1. (Note: It will increase by 2 if the increment 2 had 2 instead).

# Running the test through Command Line and Dashboard Report Generation

# Firstly, you will run the jmx test file externally from the jmeter window. But to have the line of command run run successfully, the jmx file you want to run, should be inside the Bin folder of jmeter. And you also need to be inside the bin folder through the command line. Then, you may run the following command.

# jmeter -n -t .\script\_filename.jmx -l filename.jtl -e -o .\ReportName

* -n - tells JMeter to run in non-GUI mode
* -t - specifies the path to source .jmx script to run
* -l - the file to log the samples to
* -e - generate report dashboard after load test
* -o - output folder for report generation



So, basically, your command starts from the keyword jmeter and the rest of the variables that has its own meaning as defined above.

# Dashboard Report Generation

# Dashboard report generation can be performed by running the tests from the command line.

The dashboard generator is a modular extension of JMeter. Its default behavior is to read and process samples from CSV files to generate HTML files containing graph views. It can generate the report at end of a load test or on demand.

This report provides the following metrics:

* [APDEX](https://en.wikipedia.org/wiki/Apdex) (Application Performance Index) table that computes for every transaction the APDEX based on configurable values for tolerated and satisfied thresholds
* A request summary graph showing the Success and failed requests (Transaction Controller Sample Results are not considered) percentage.

# There are various charts you can find under index.html to analyze behavior of performance test execution. You can configure these zoom-able charts to show/hide samples.