**Apache Jmeter:**

Apache JMeter is open-source software that is popular for performance testing. This tool is designed to load test functional behavior and measure performance.

This article is about how to use JMeter for Performance Testing of a RESTful API. In this, we will learn how to configure JMeter and test a REST API to know how efficiently it works and how many concurrent users the server can handle.

For this, I created a RESTful API that lets you do the CRUD operation on a “Product.” A Product is a simple object that has a name, description, and price.

The API was created using three components of the MEAN stack –  NodeJs, Express and MongoDB.

The API methods are as follows:

Using JMeter, we will design the test that will

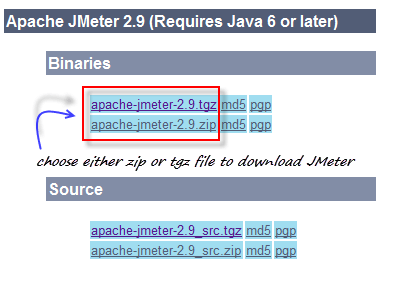
* Use HTTP GET to retrieve a list of all items
* Use HTTP POST to add a new item
* Use HTTP PUT to update a newly added item
* Use HTTP DELETE to delete the item added

### Step 1) Install Java

**Step 2) Download Jmeter**

As of this writing, the latest version of JMeter is **Apache JMeter 4.2**. You can download it [here](http://jmeter.apache.org/download_jmeter.cgi) But this tutorial demos installation of version 2.9, the install process remains the same.

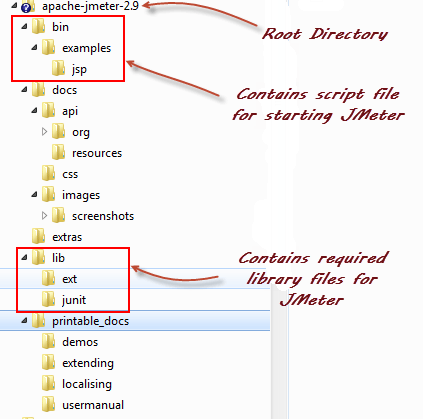
Choose the Binaries file (either zip or tgz) to download as shown in the figure below

[](https://www.guru99.com/images/ApacheJmeter.png)

**Step 3) Installation**

Installation of JMeter is extremely easy and simple. You simply unzip the zip/tar file into the directory where you want JMeter to be installed. There is no tedious installation screen to deal with! Simply unzip and you are done!

Once the unzipping is done installation directory structure should look like as figure below

[](https://www.guru99.com/images/ApacheJmeter2_9.png)

Given below is the description of the JMeter directories and its importance JMeter directory contains many files and directory

* /**bin**: Contains JMeter script file for starting JMeter
* /**docs**: JMeter documentation files
* **/extras**: ant related extra files
* /**lib**/: Contains the required Java library for JMeter
* **/lib/ext**: contains the core jar files for JMeter and the protocols
* **/lib/junit**:[Junit](https://www.guru99.com/junit-tutorial.html)library used for JMeter
* /**printable\_docs**:

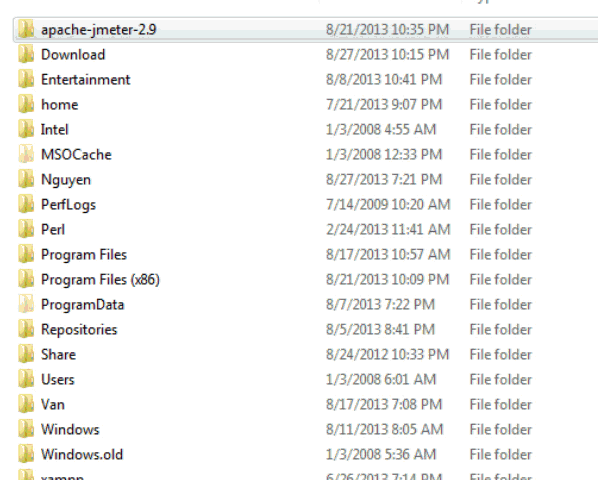
**Step 4) Launch JMeter**

You can start JMeter in 3 modes

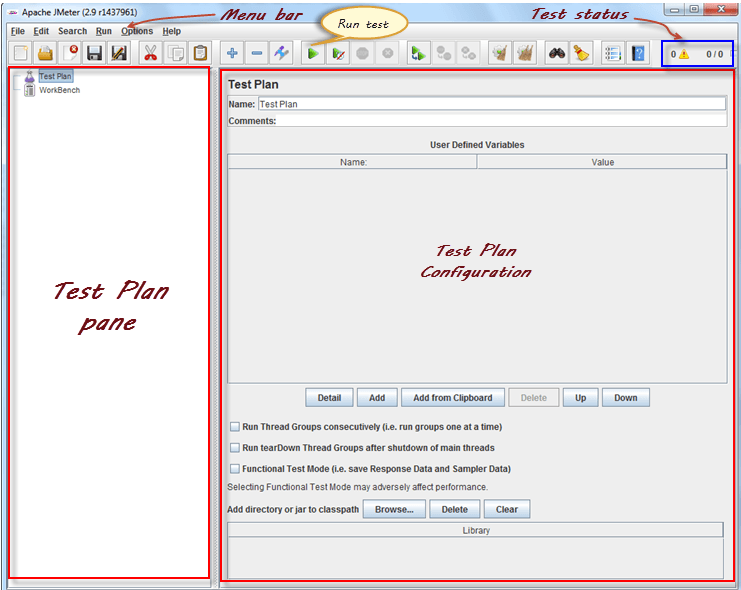
* GUI Mode
* Server Mode
* Command Line Mode

**Start JMeter in GUI Mode**

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

[](https://www.guru99.com/images/startjmeter.gif)

The following figure annotates the various components in the JMeter GUI

[](https://www.guru99.com/images/ApacheJmeterSnap.png)

**PERFORMANCE TESTING OF A RESTFUL API USING JMETER**

Apache JMeter is open-source software that is popular for performance testing. This tool is designed to load test functional behavior and measure performance.

This article is about how to use JMeter for Performance Testing of a RESTful API. In this, we will learn how to configure JMeter and test a REST API to know how efficiently it works and how many concurrent users the server can handle.

For this, I created a RESTful API that lets you do the CRUD operation on a “Product.” A Product is a simple object that has a name, description, and price.

The API was created using three components of the MEAN stack –  NodeJs, Express and MongoDB.

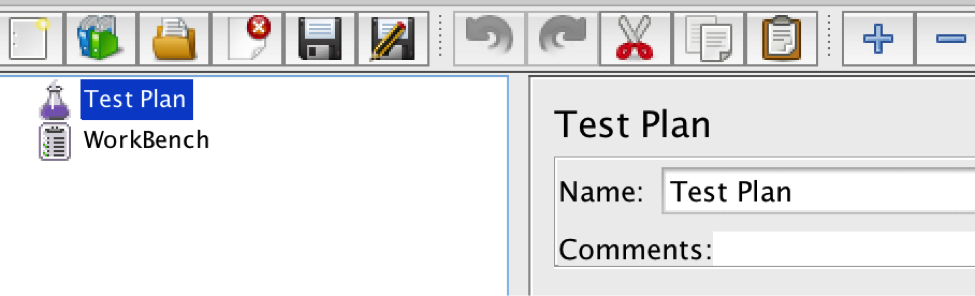
The API methods are as follows:

Using JMeter, we will design the test that will

* Use HTTP GET to retrieve a list of all items
* Use HTTP POST to add a new item
* Use HTTP PUT to update a newly added item
* Use HTTP DELETE to delete the item added

STEPS FOR SCRIPTING A REST API IN JMETER

As soon as you launch JMeter, you will see 2 elements: Test Plan and Workbench.

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_1.png)

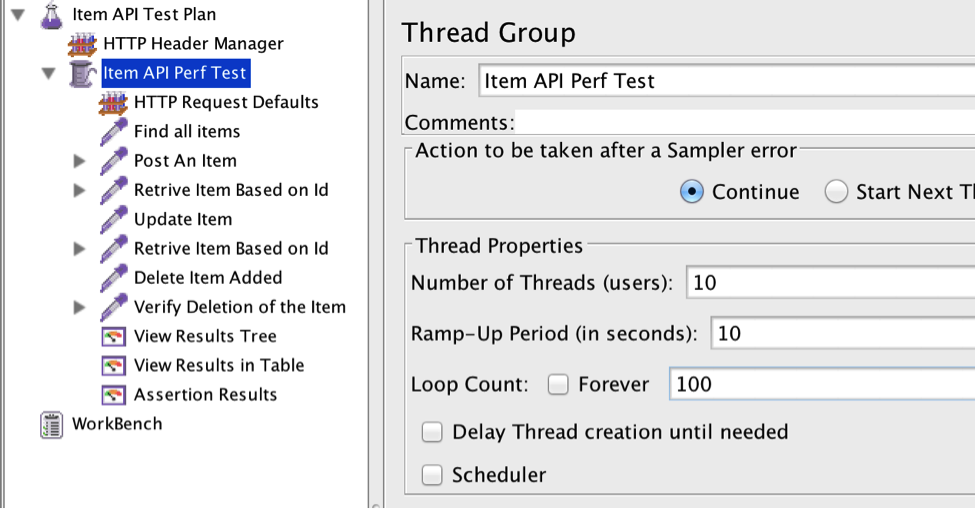
1. This is the REST API. We will posting data as a JSON object, so we need to set a Content-Type header. For this, right click Test Plan and add Config Element → Http Header Manager and add “Content-Type.” Set value to “application/json.”

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_2.png)

2. Right click Test Plan. Select Add → Threads (Users) → Thread Group. Add the thread group.

1. Set the name to “Item API Perf Test”
2. Set the number of threads (users) to 10
3. Set the Ramp-Up period (in seconds) to 10
4. Set the Loop count to 10

So we will have 10 users executing the test plan 100 times. The Ramp-Up Period tells JMeter how long to delay before starting the next user.

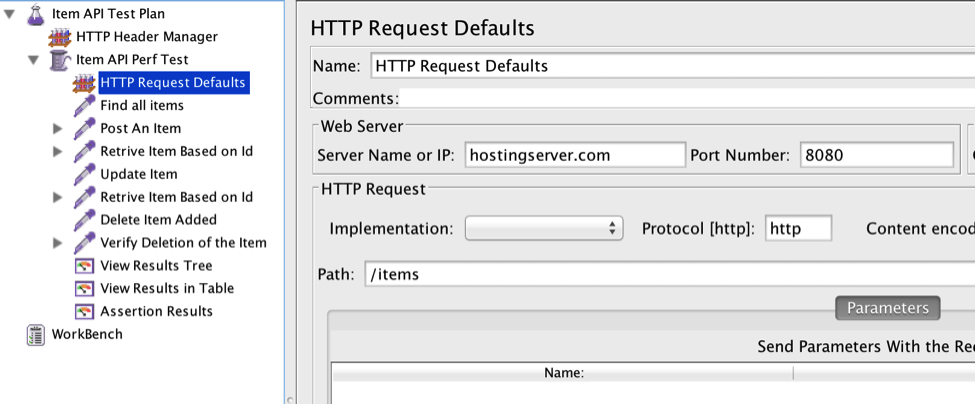
[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_3.png)

3. Right click Thread Group (Item API Perf Test) and add Config Elements → HTTP Request Defaults.

HTTP Request Defaults: This element lets you set default values that your HTTP Request controllers use. For example, if you are creating a Test Plan with 25 HTTP Request controllers and all of the requests are being sent to the same server, you could add a single HTTP Request Defaults element with the “Server Name or IP” field filled in.

So set the server name, port number, and the path to the resource.

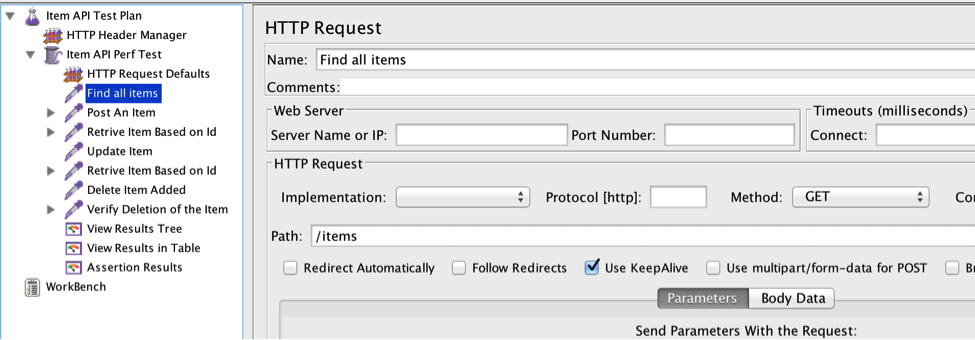
* Server Name: hostingserver.com (Server address where the RESTFul API is running)
* Port Number: 8080
* Path: /items

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_4.png)

4. Right click Thread Group (Item API Perf Test) and add Sampler → HTTP Request

Our first request will be to fetch all of the existing items. So set:

* Name: Fetch All items
* Method: GET

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_5.png)

5. For the second request to POST an item (aka create), let’s add another sampler. Right click Thread Group (Item API Perf Test) and add Sampler → HTTP Request

* Name: Post an Item
* Method: POST
* Body Data: JSON Item Object

Once the item is posted successfully, the response return will have a unique ID of the item.

{ "item": { "\_\_v": 0, "name": "MacBook Pro 15-inch with Retina display ",

"description": "MacBook Pro 15-inch with Retina display",

"price": 1699, "priceUnit": "$", "\_id": "5666907e6c5b4bb5525ba617"}

}

We need to extract that value so that we can use it in the subsequent requests to perform PUT (update) and DELETE operation.

For this, right click HTTP Request Sample (Post an Item) and add Post Processor → BSF PostProcessor. BSF PostProcessor provides read/write access to JMeter variables

Set the language to Javascript and add the following into the script:

var data = prev.getResponseDataAsString();

var object = JSON.parse(data);

vars.put("itemId", object.item.\_id);

WHAT THE ABOVE SCRIPT IS DOING:

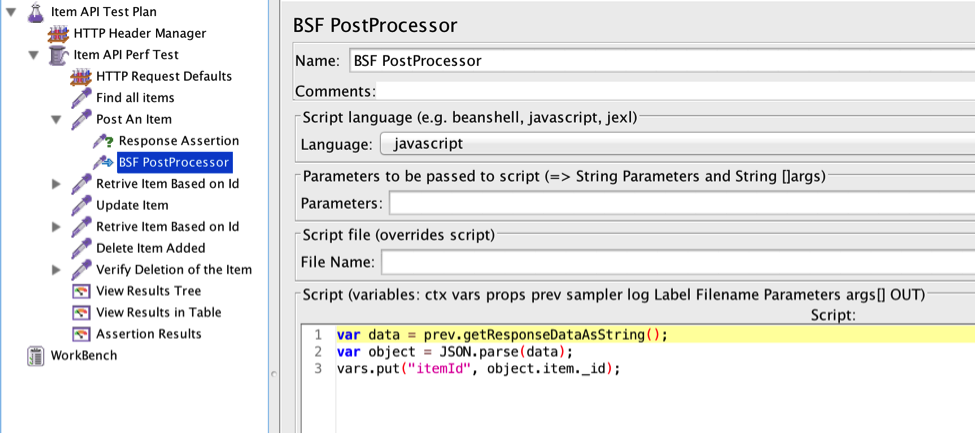
If you look into the “Script” section of the BSF Post Processor, you’ll see the following:

* Script (variables: ctx, vars, props, prev, sampler…)
* vars – stands for JMeterVariables. You can get or set variable values by using vars.
* prev – shorthand to previous SampleResult. In this scenario, it contains the response of the POST request.
* prev.getResponseDataAsString()  will extract the response as a string.

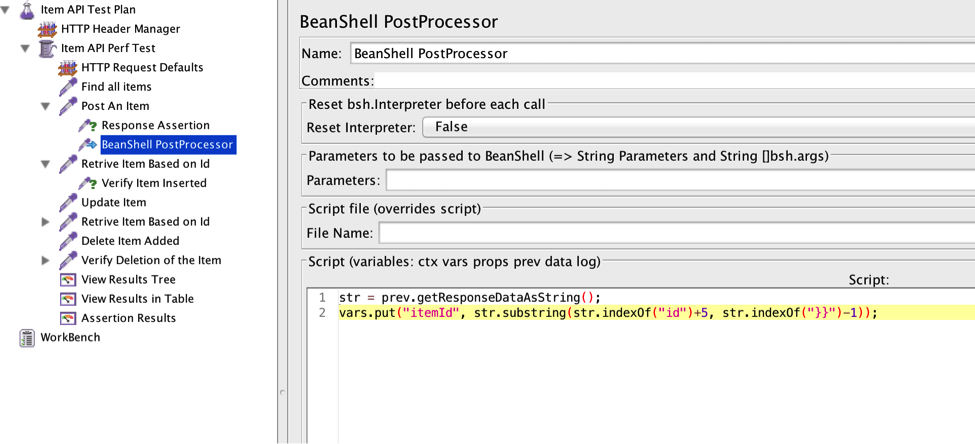
Use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object.

For the subsequent requests we need an ID, so we will create a variable called itemId:

vars.put("itemId", object.item.\_id);

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_7.png)

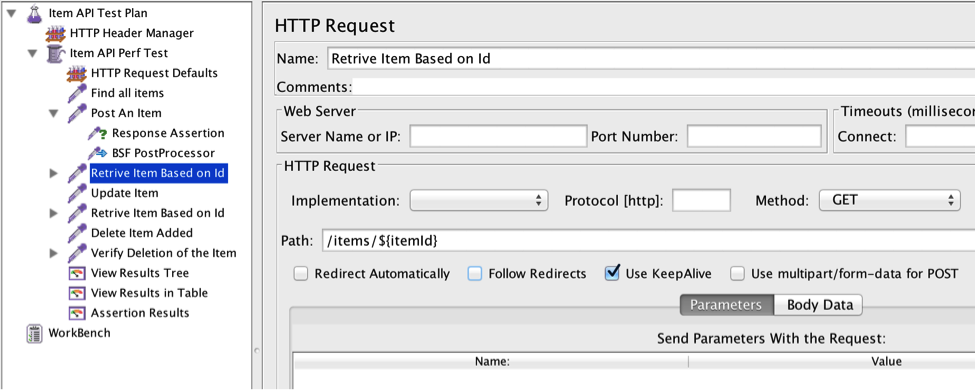
You can also use the BeanShell Post Processor to parse the JSON. We can easily invoke Java code in the Beanshell. For this scenario, here is simple way to extract the item ID.

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_8.png)

6. Now we will verify the insertion by retrieving the inserted item. For that, again right click Thread Group (Item API Perf Test) and dd Sampler → HTTP Request

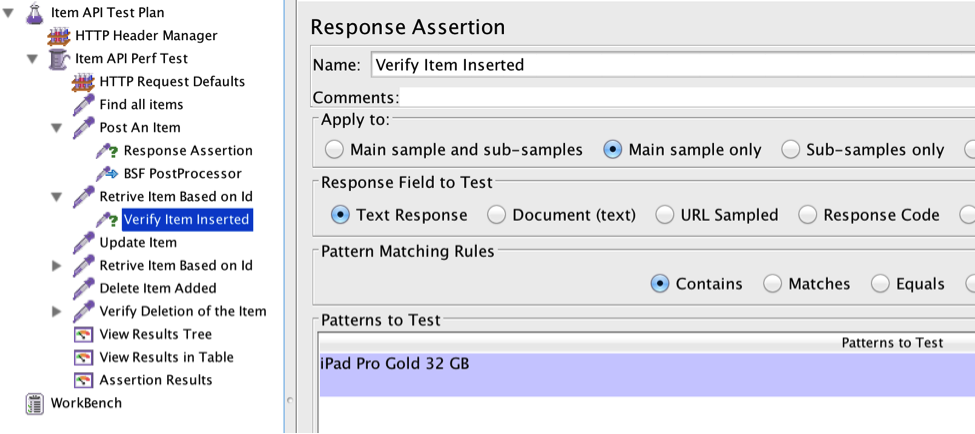
* Path: /items/${itenId}
* Method: GET

By adding ${itemId} to the path, the request will be retrieving the single item.

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_9.png)

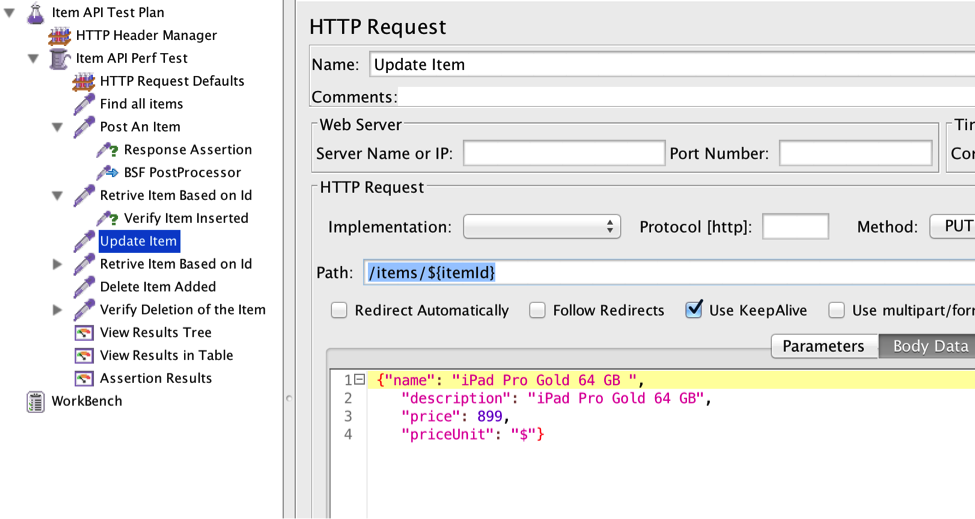
To verify the item inserted, add Response Assertion to the HTTP Request Sample (right click Assertions –> Response Assertion). Response Assertion lets you add pattern strings to be compared against various fields of the response.

For this, select the Text Response radio button for “Response Field to Test”. Set Pattern Matching Rules to “Contains” and add the item name in Patterns to Test.

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_10.png)

Please note that if the assertion fails, the test will stop and subsequent requests will be executed.

7. The next step is to update the item added. To do this, add another HTTP Request Sampler. Set the method to PUT and add the item’s new detail as Body Data. Also set the path to /items/${itemId}

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_11.png)

8. Now the last step is to delete the item added. For that, also add another HTTP Request Sampler. Set the method to Delete

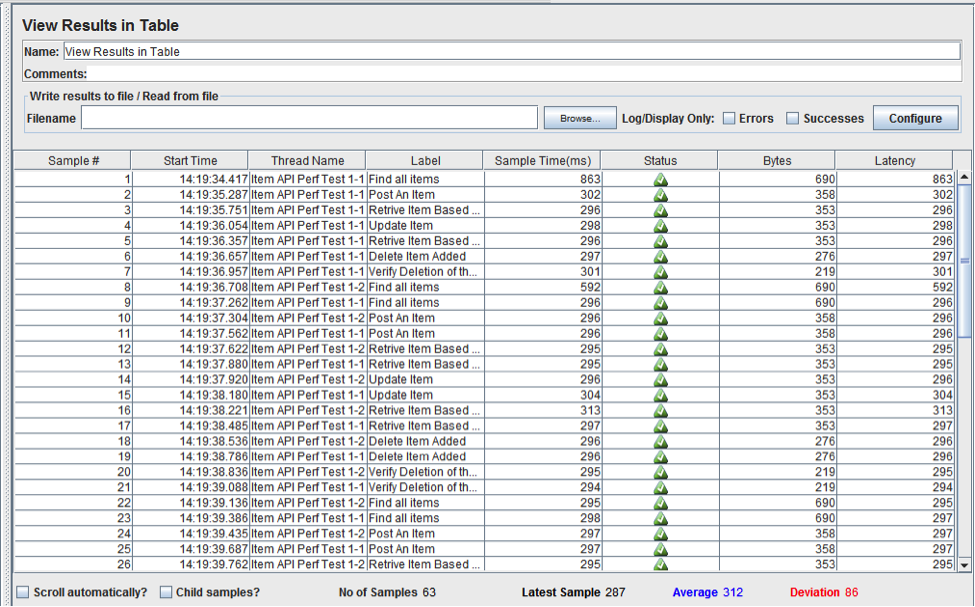
[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_12.png)

9. At last we can add different listeners like “View Results Tree” to conclude. To add, right click and select Listeners → View Results Tree.

In the View Results Tree, you can see the request and response data for each sample request.

So the last step is to save and execute the test by pressing the green arrow on the top menu bar.

Here is how the results look in View Results in the Table, which shows a row for every sample request.

[](http://www.3pillarglobal.com/wp-content/uploads/2016/02/jmeter_14.png)

In this way, we can do the performance as well as functional testing of a RESTful API.

By increasing the number of threads and loop count, we can increase the load on the server and measure various vitals of the server and API, such as CPU Utilization and average response time.