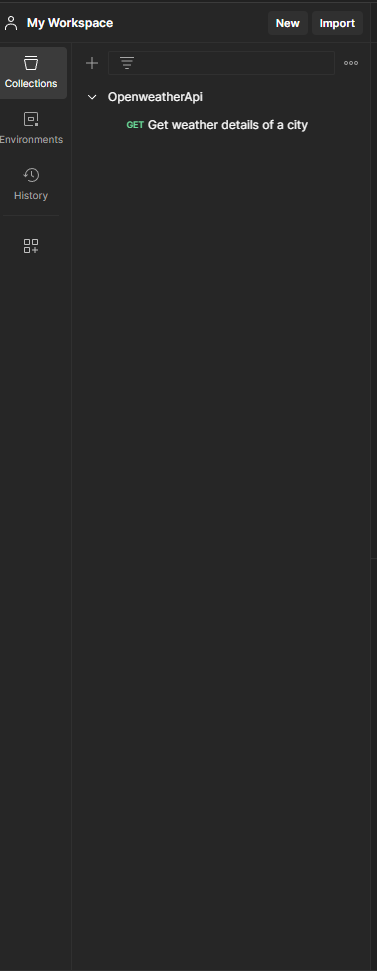
**Project**: API Testing and Performance Testing

Source code link:

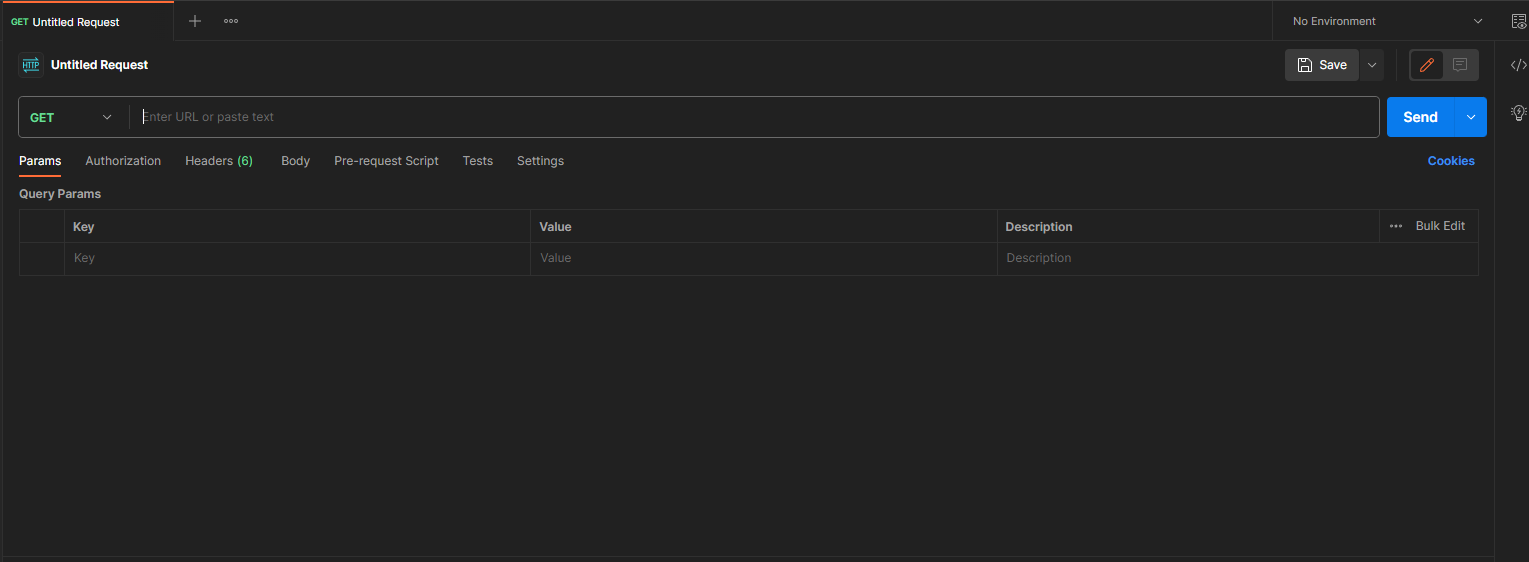
**Project Description**: API Testing using Postman, Rest-Assured and performance testing using Jmeter.

Secnario-1:

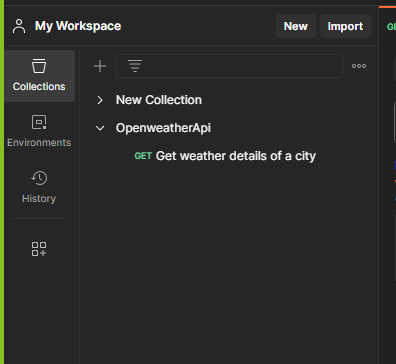


Open postman and click on + icon to create a collection

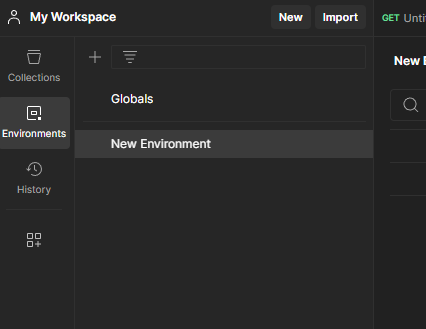
Click on + on workspace environment to create a Request.



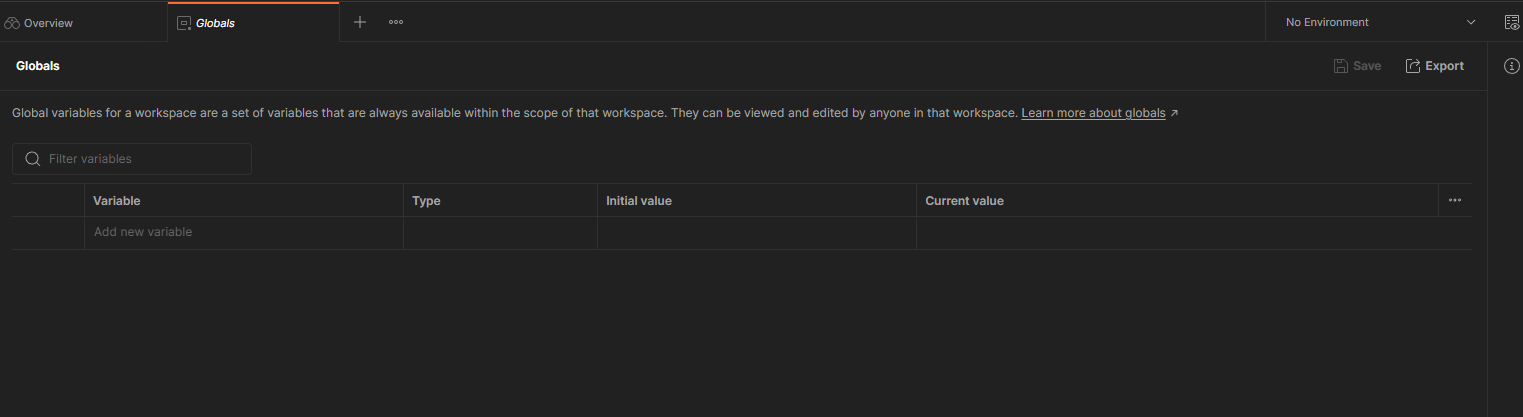
To create variables click on environment option



Then click on Globals to a global variable



Add variable as URL and enter the value in fields



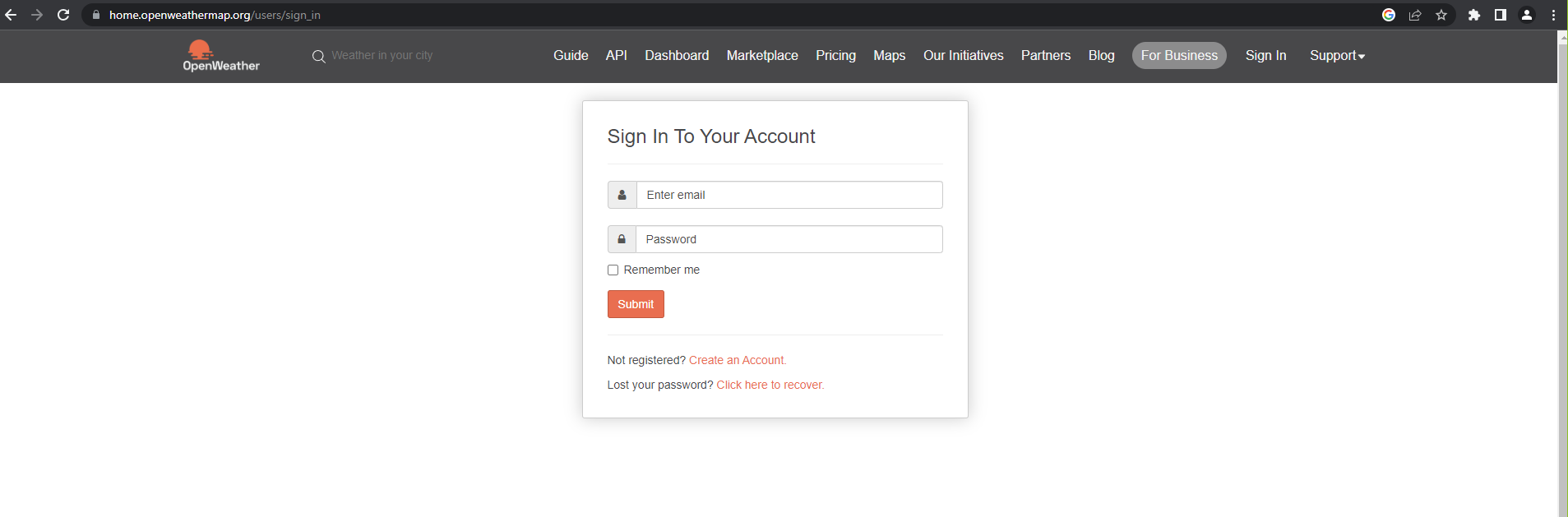
Name: URL

Type: Default

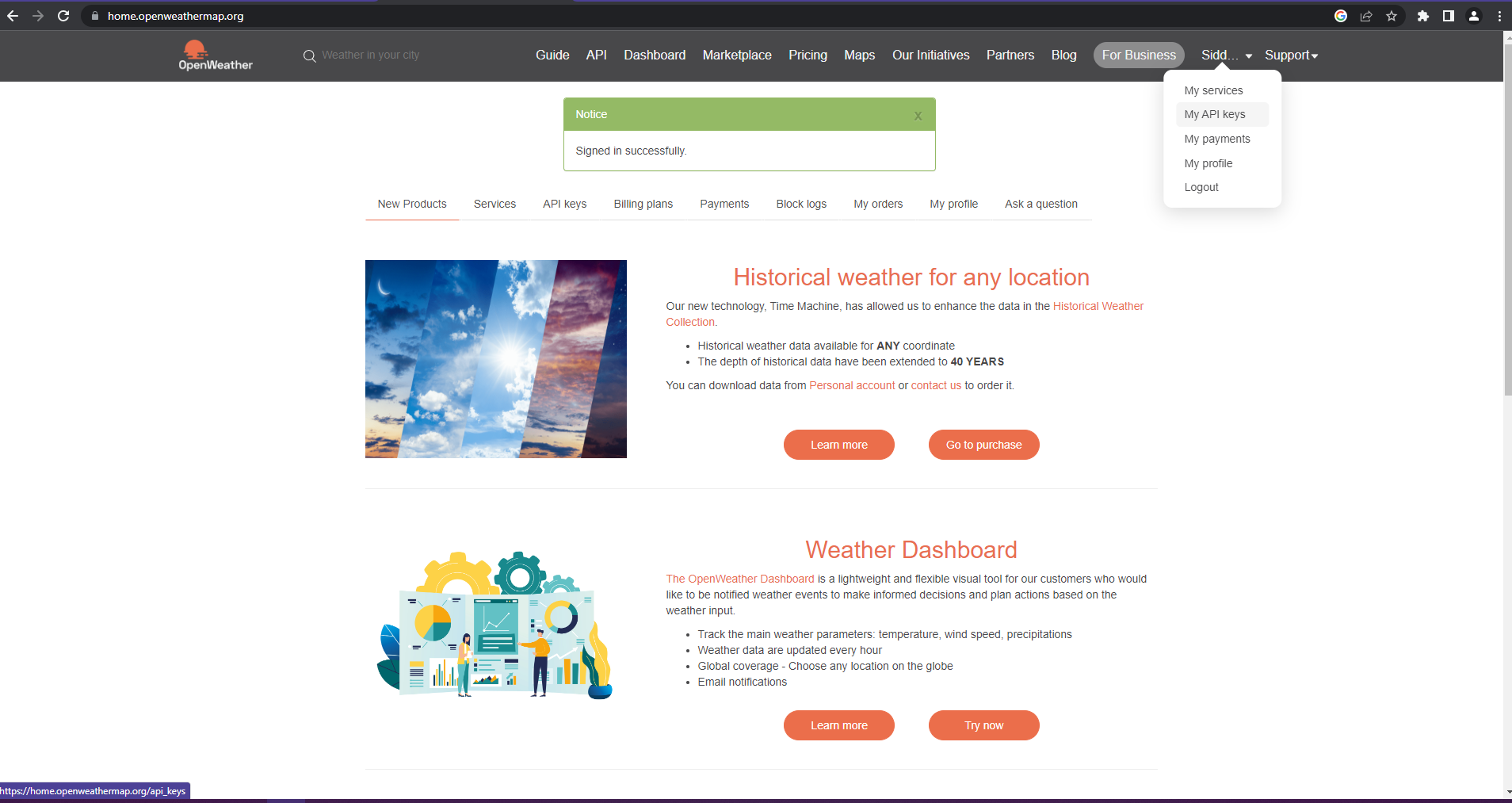
Value: <https://api.openweathermap.org/data/2.5/weather>?

Then Create an API Key in open weather application to access the weather information.

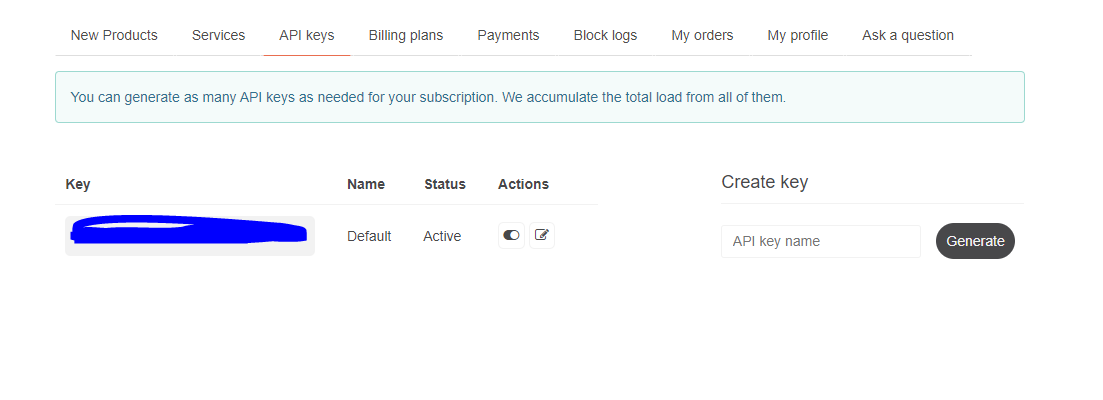
Go to <https://openweathermap.org/> and create an account yourself.



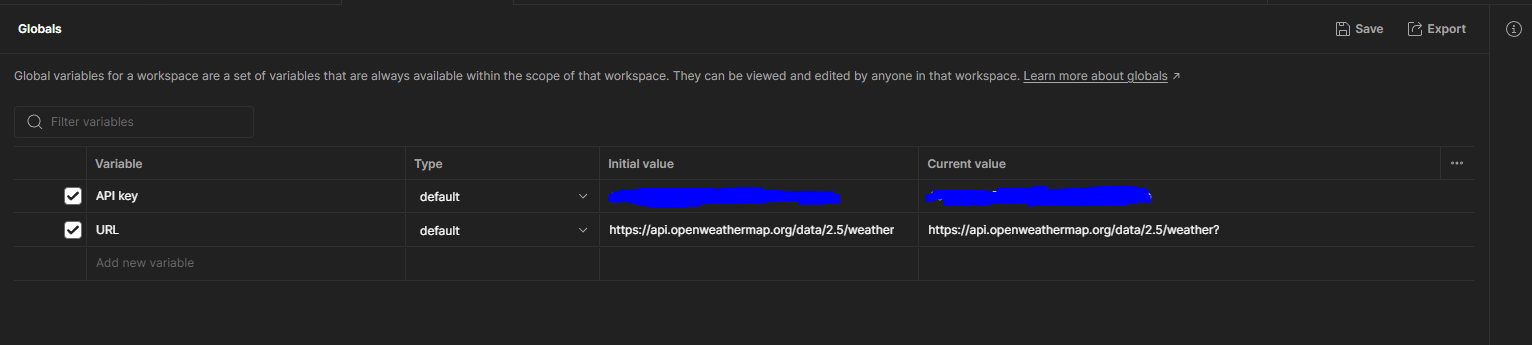
Go to account and find My API’s



By default an API Key is generated. Copy the key name and value.

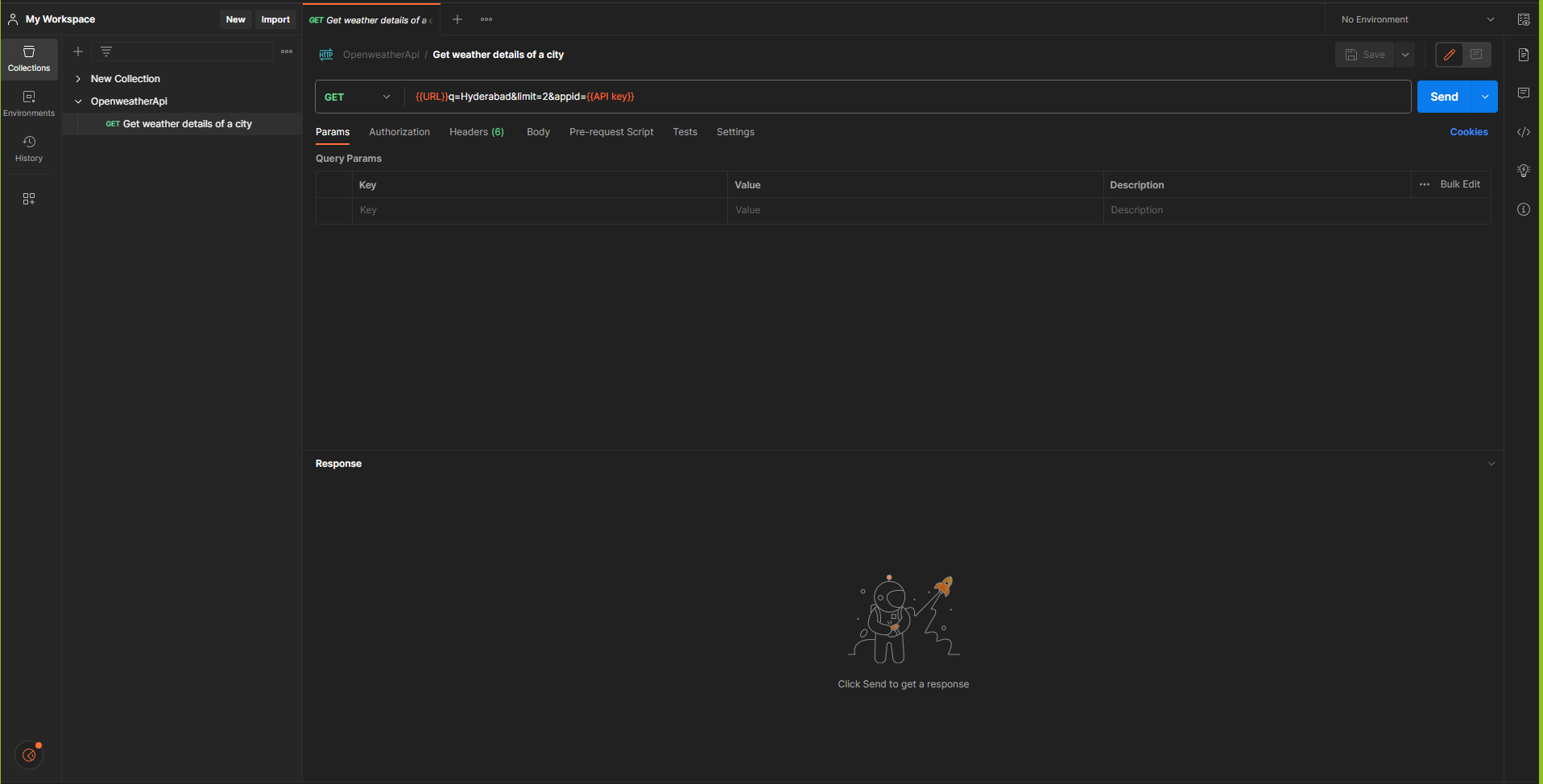


Then come to Postman and create a variable for API like we created for URL



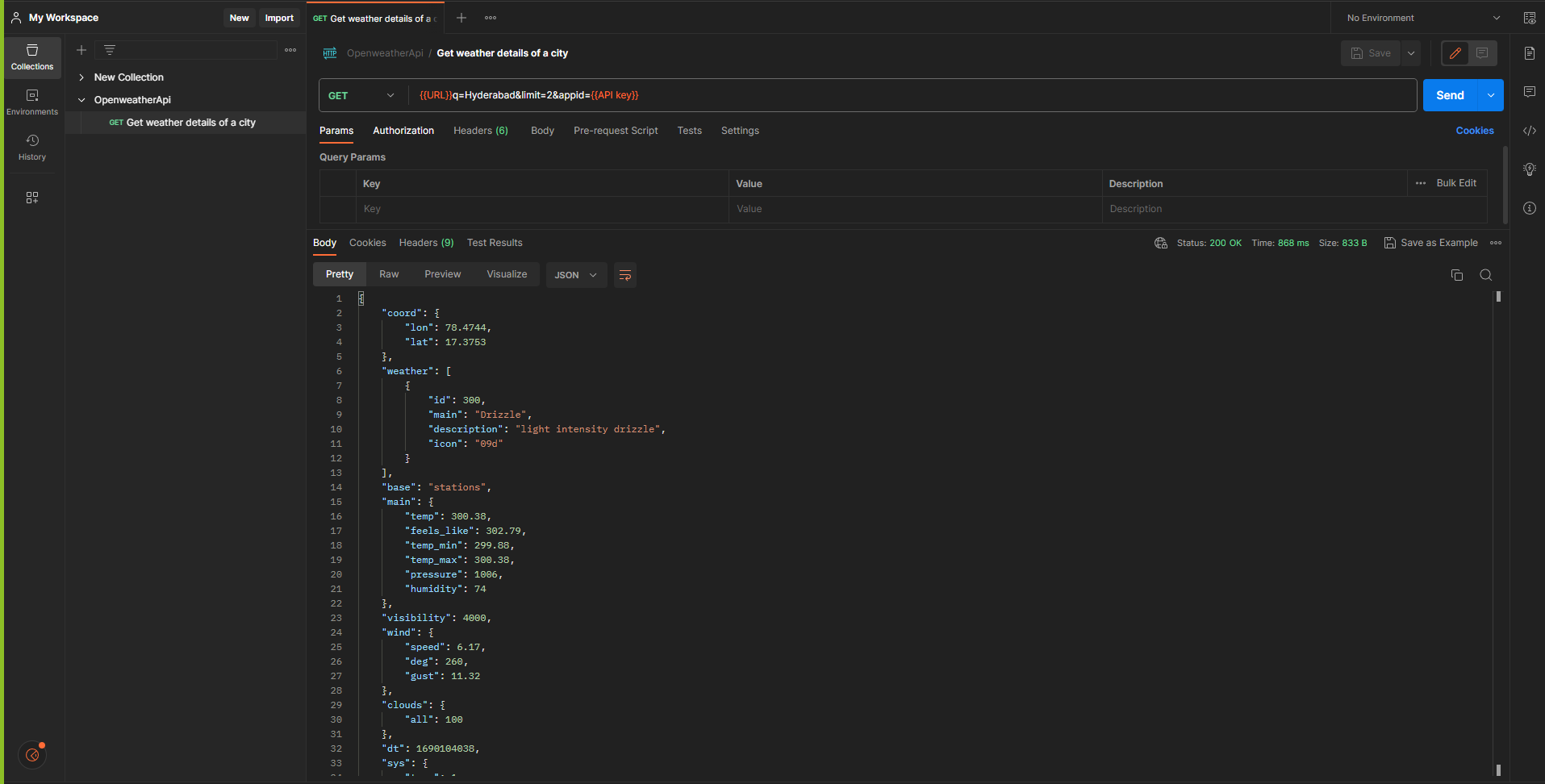
Click on save button to the variables.

Then go to Request tab and call the variables and pass the parameters of the city and state to get the weather detail. Save the request under openweatherAPI Collection.

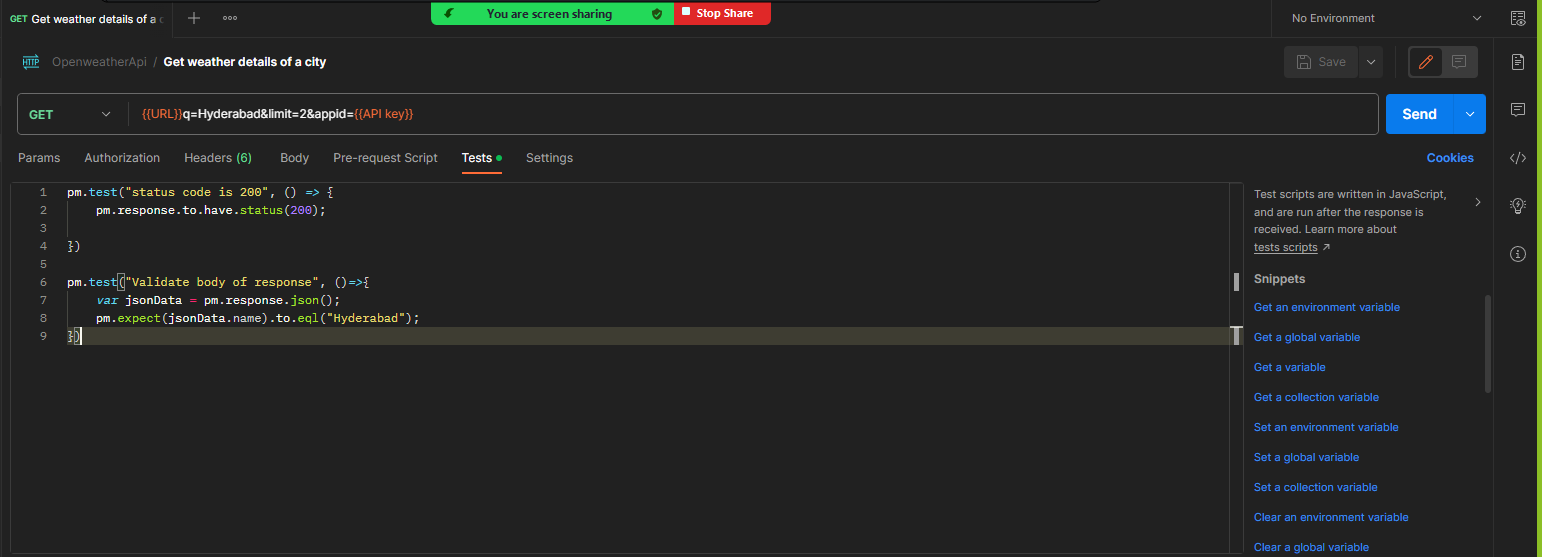


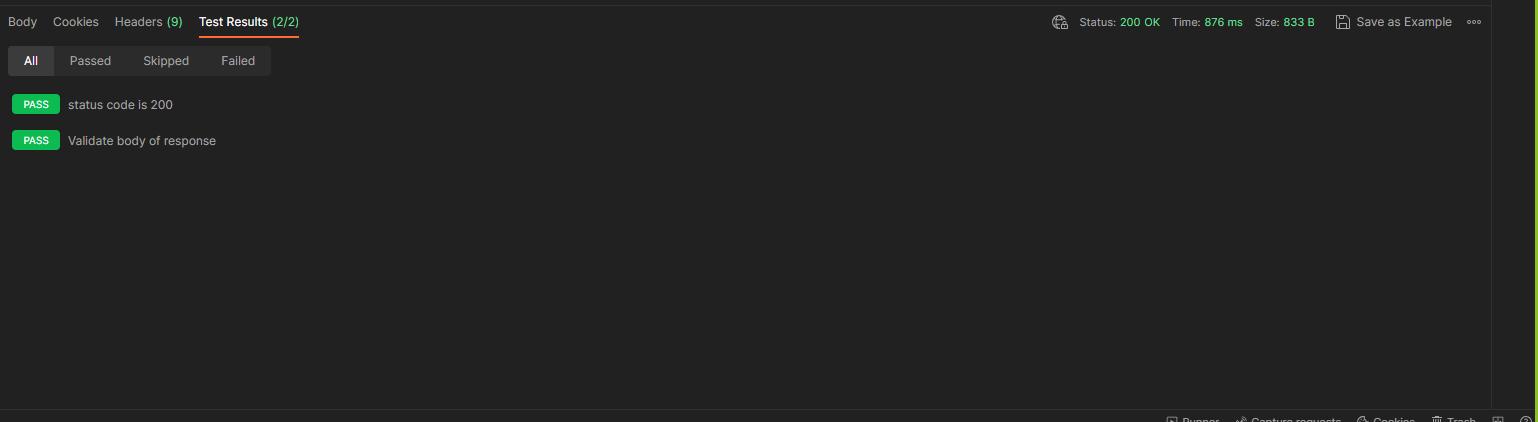
Then Send the request and wait for the response from the server.

Response from the server in JSON format (default). We can change the response format (XML/HTML/Text)



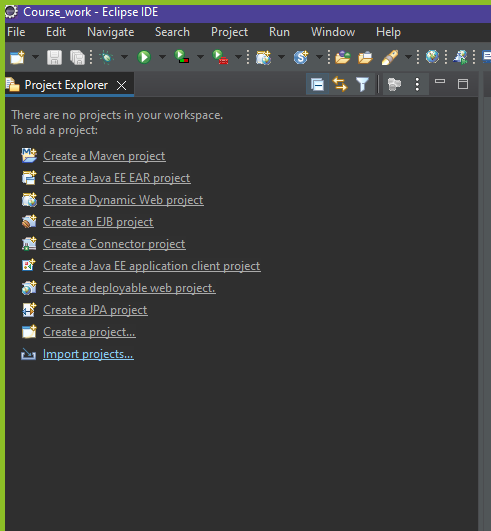
To validate the response, we can add the test scripts under “Test tab”



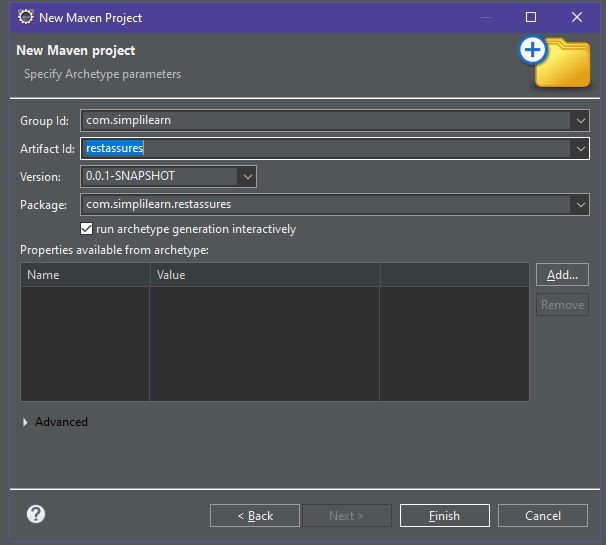


Secnario-2:

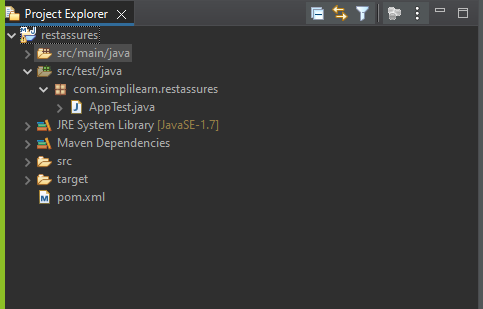
Open Eclipse and create a Maven project

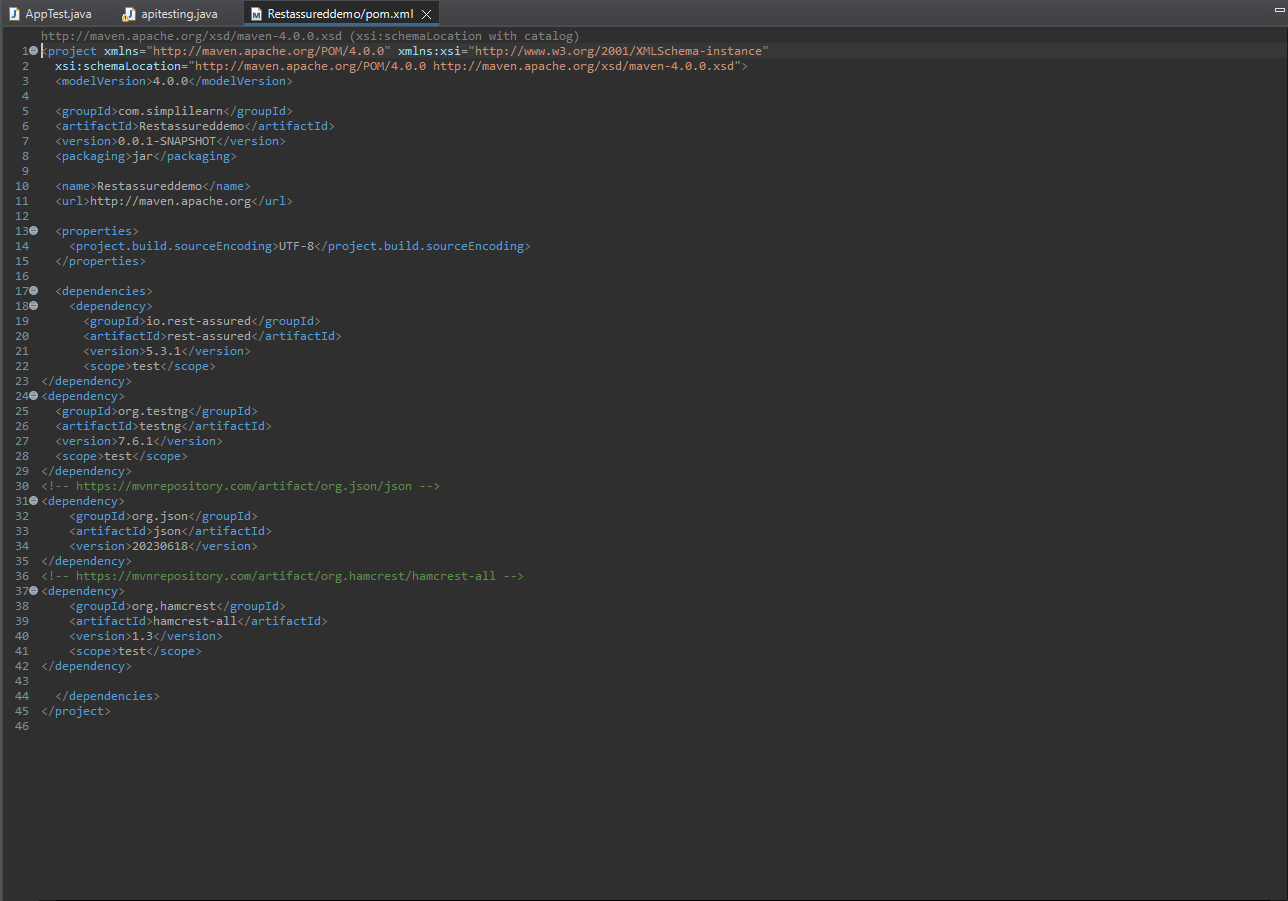


Enter the Group ID, Artifact like below then click on finish button to create the project.

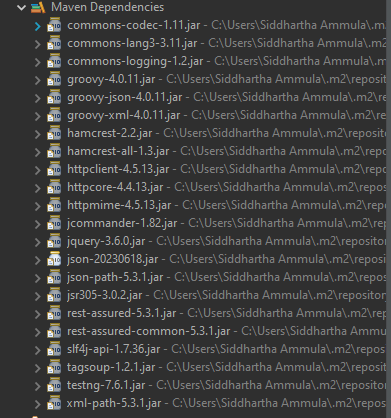


Open pom.xml and add the dependencies

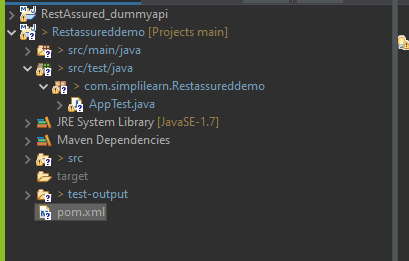




Save the pom.xml and wait for Eclipse to load the dependencies into the project.

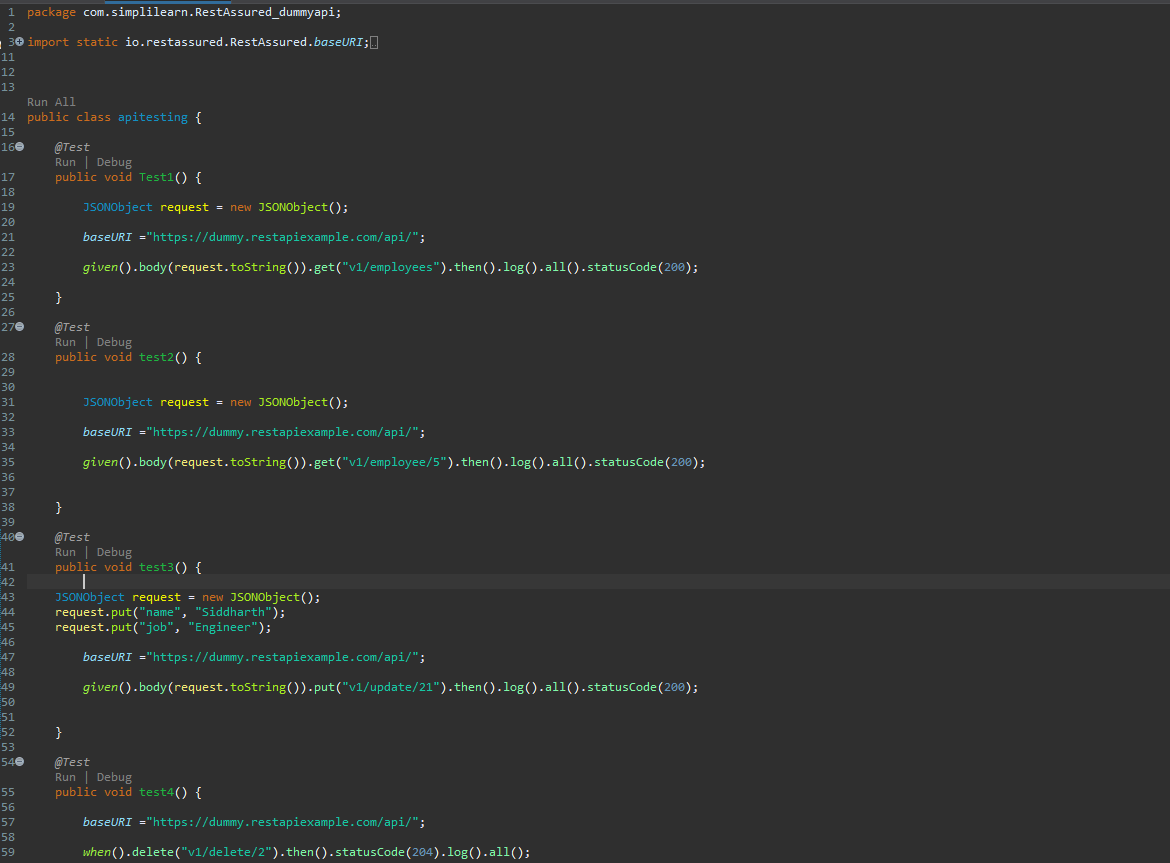


Click on src/test/java to create a new class file



Go to <https://dummy.restapiexample.com/> and test the API request mention in the portal.

And call the API’s in the test file

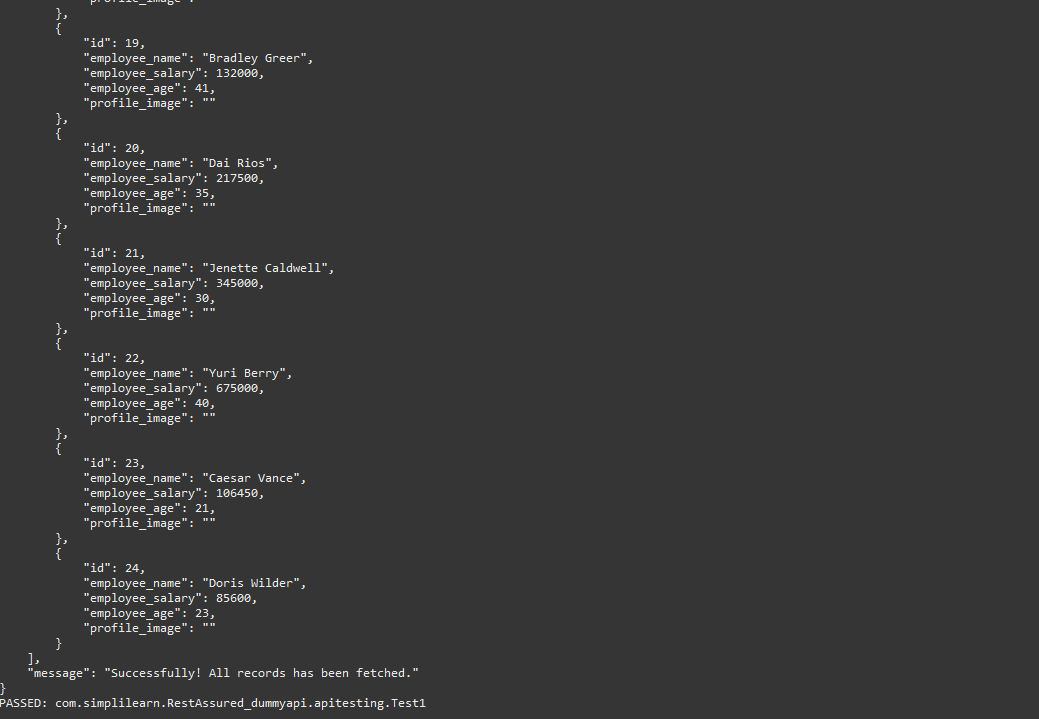


And log the response from the API request to the console.

Log Data Results:

Get all Employees data:

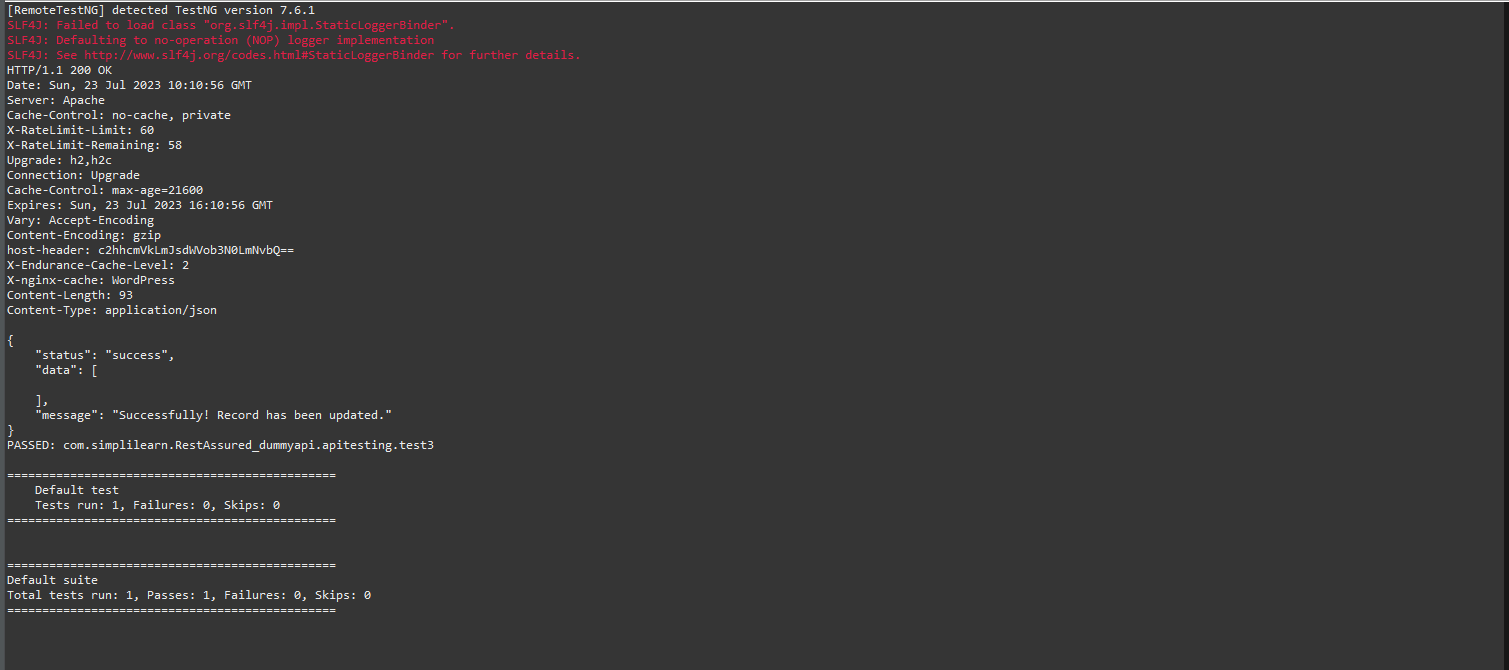




Get Employee data by Id:

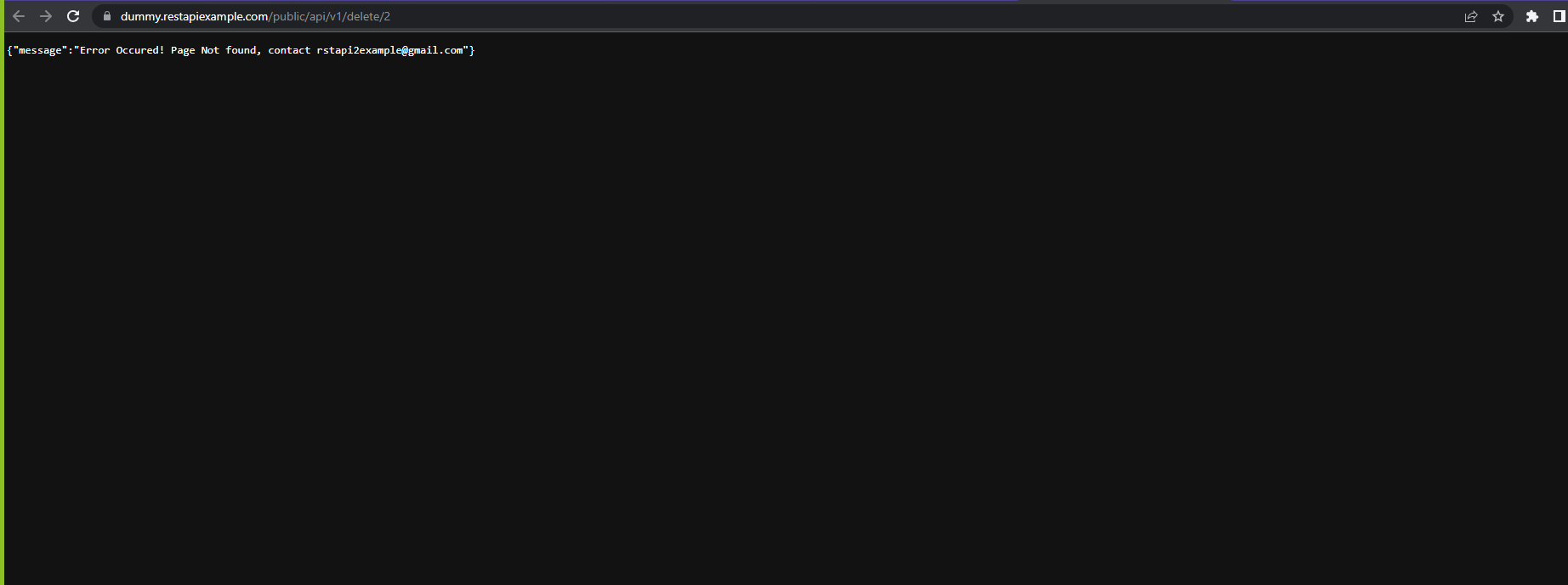


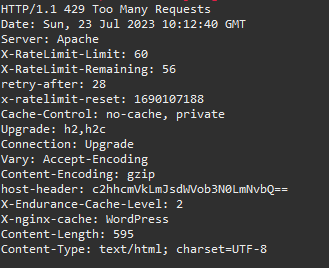
Update the Employee data using Put request:



Delete Employee data using delete Request:

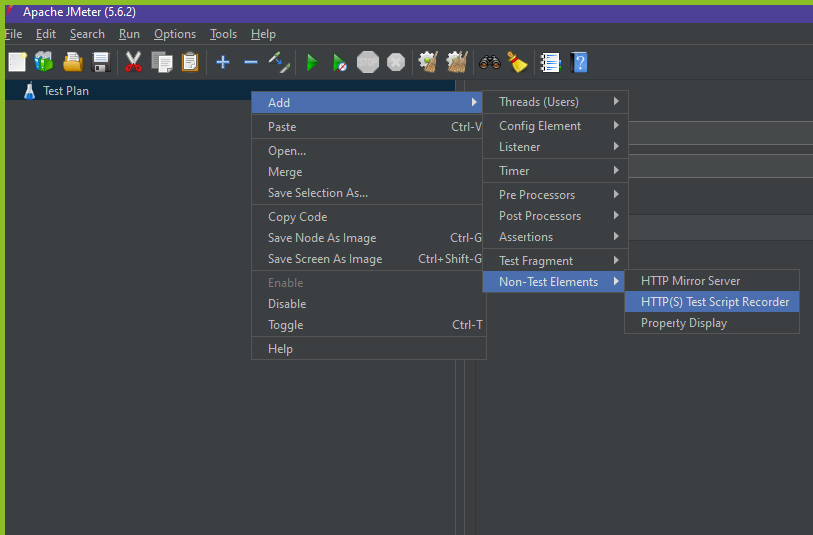
We tried to call the API request but page showing error “Page not found” it showing. Also we tried manually check the API, still it showing same error



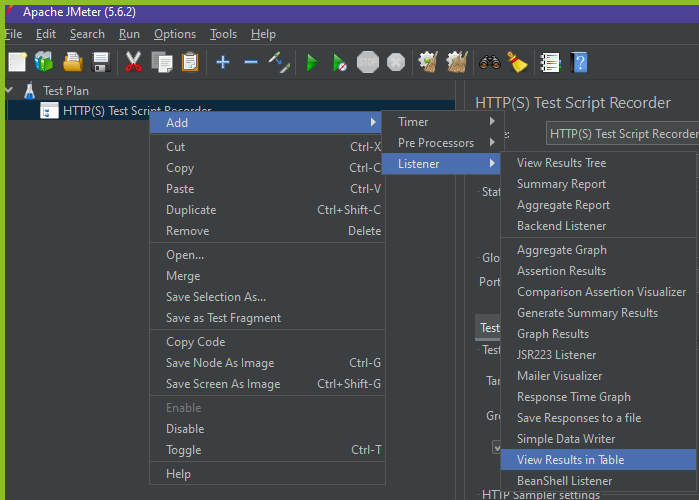


Secnario-3:

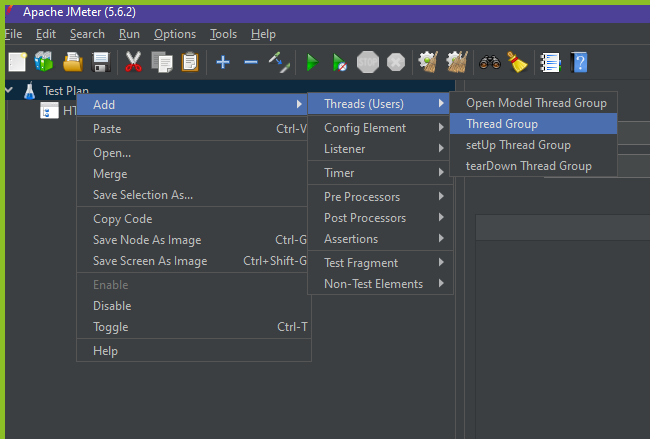
Open Jmeter and Right Click on the Test plan to add “Test Script Recorder”



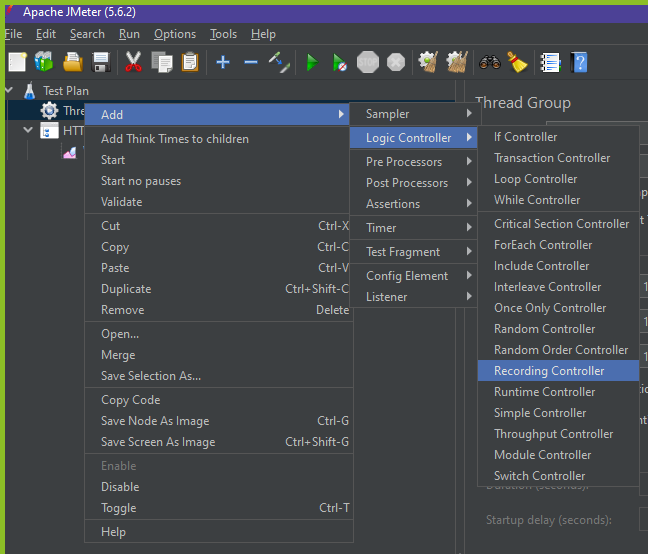
Right click on the Test Script recorder and add the Listener (view Results in table).



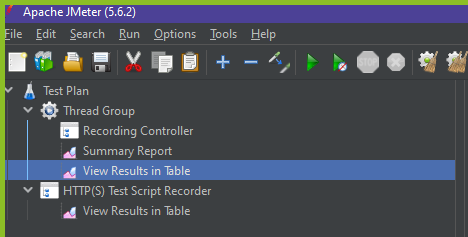
Next create a thread Group by right click on the Test plan



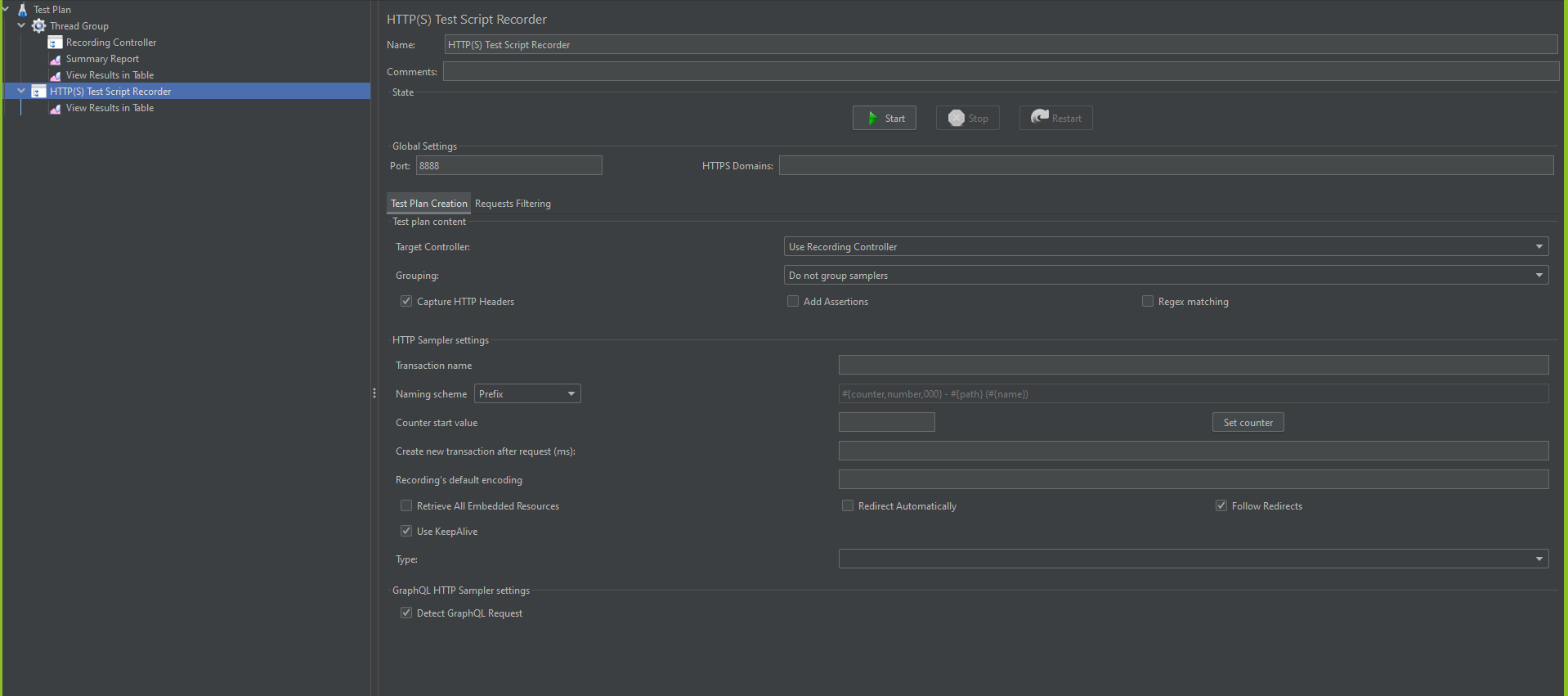
Add Recording Controller under Thread group

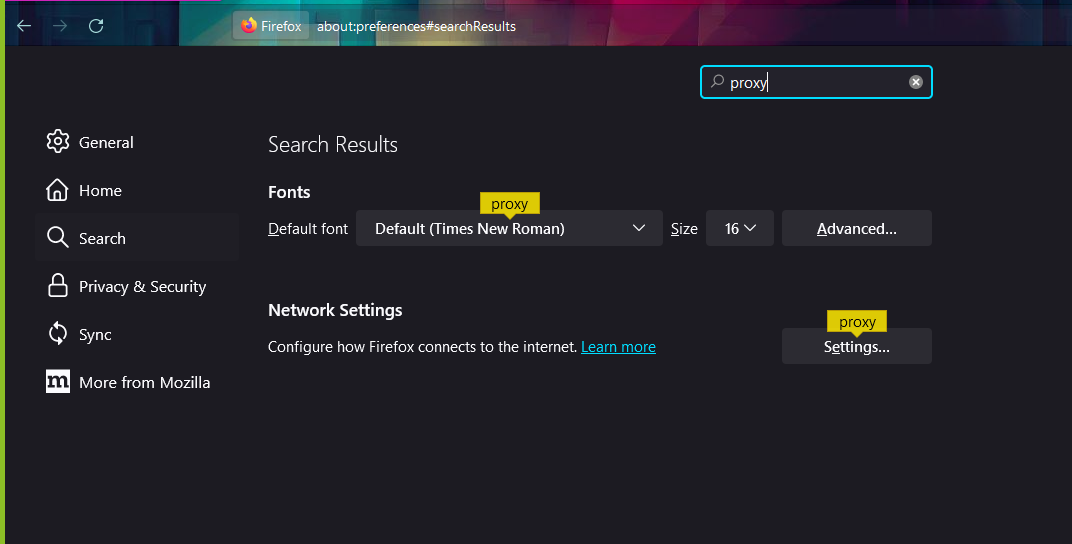


Add the Listener under Thread Group

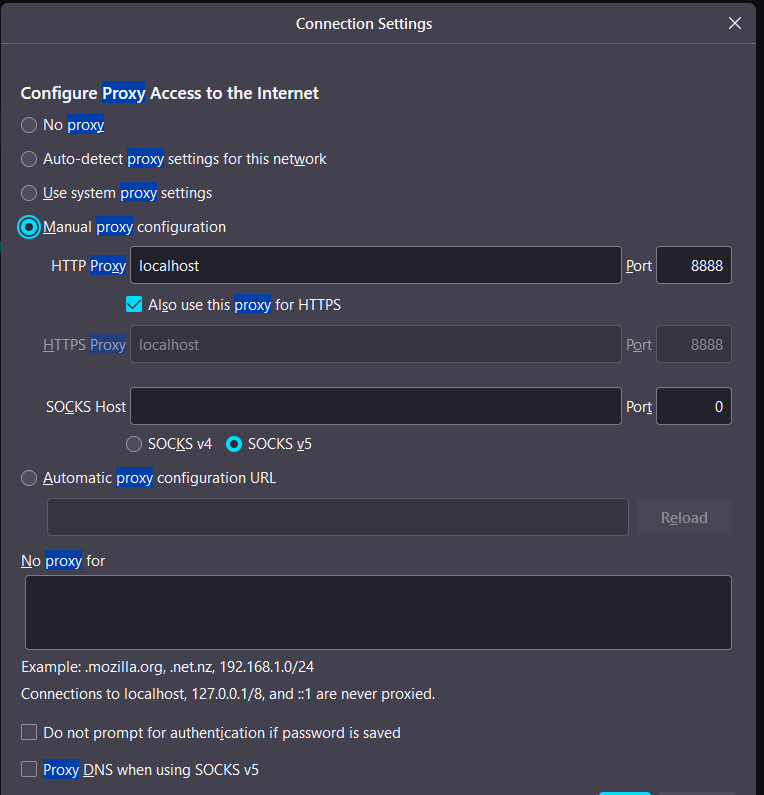


Now open the test script recorder and copy the port number. Then Open Firefox and goto Proxy settings under network options.

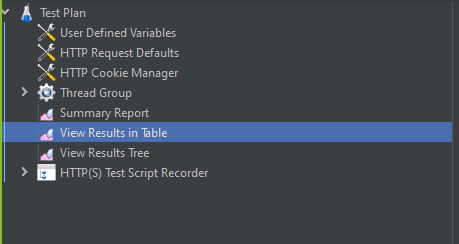




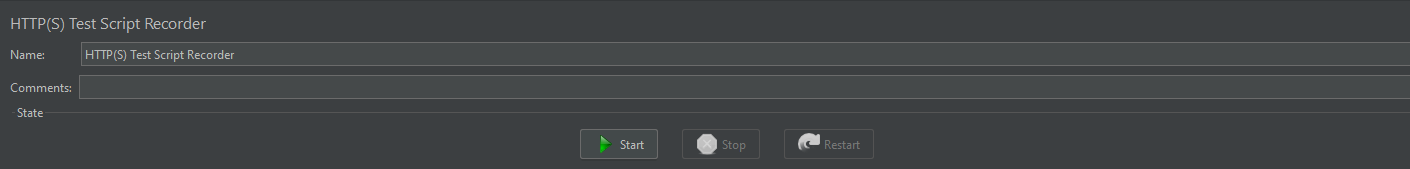
Choose manual proxy and server name as “localhost” and enter the port number as shown in “Test script recorder”.



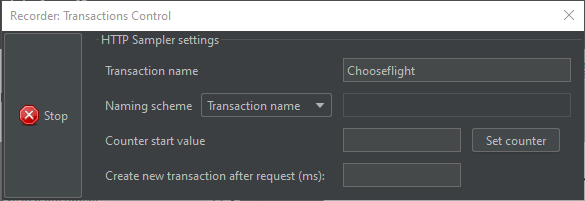
Final structure of the Test plan

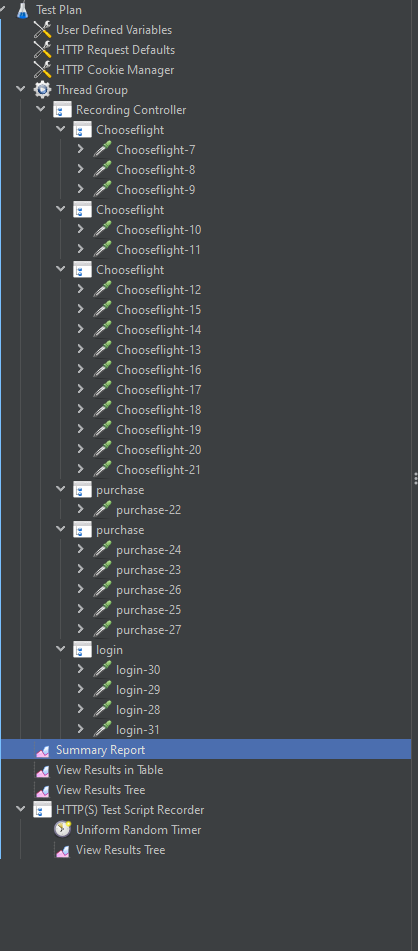


Click on the start button to record the user action the webpage.



Set the transaction as “chooseflights” in the pop-up





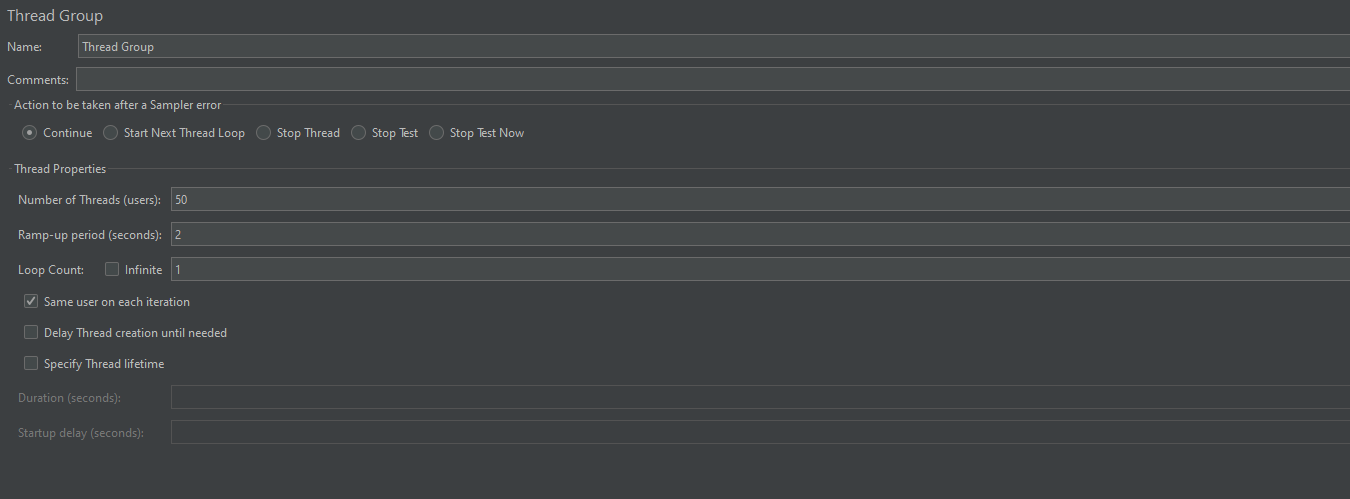
Transaction name: Login

Transaction name: Purchase

Transaction name: choose flights

After completing and stop the recorder.

Go to thread add virtual users and Ramp-up period



Save the test plan and click on start button to run the test.



Wait for Jmeter to execute the results

Observe the summary results of each transaction.

