Magazine

OpenSta – OpenSource for Web Load, HTTP Stress & Performance testing

Author: Łukasz Smolarski

About the Author:

Łukasz Smolarski :

Graduated from Higher School of Business-National Louis University – faculty: “Computer Science” and Leon Koźminski Academy – faculty: “Management”. During his studies he won a scholarship for leaders funded by GE Foundation and Institute of International Education. He currently works for Gtech Polska on the position of Quality Software Engineer as a Team Leader and person responsible for test automation. In 2007 he passed ISTQB Foundation Level, and in 2010 become AIS Certificated Specialist in HP Mercury Quality Center and Mercury QuickTestPro. Member of SJSI. Contact: smolar2@op.pl

Intermediate

Level

1

Magazine Number

Software testing

Section in the magazine

Introduction

Every single day in our work we can notice that all types of tests are run repetitiously – which obviously is wasting a lot of valuable time. It's a hard work when every single scenario has to be repeated several times. There's a selection of free tools which can help in test automation process – in this article I will describe one of them. The tool named OpenSta is mainly designed for measuring performance testing, however it can also be used for other purposes, such as automation of certain actions performed in testing activities.

OpenSta is continuously developed free tool which can be downloaded from http://opensta.org website. Moreover, in case of problems or questions, we can use the forum for users of this tool and search or ask for help.

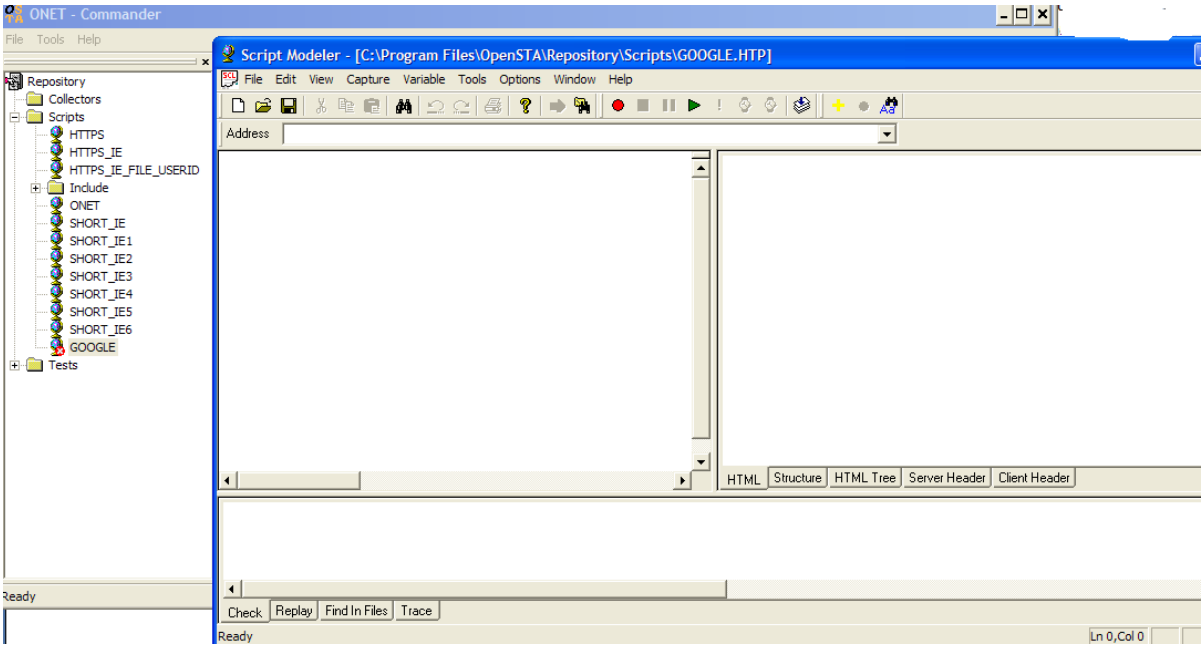
First steps with OpenSta

After installation, OpenSta Commander should be run. It's the main screen of the application and contains a tree, which is our repository and place where we can keep our scripts and tests.

The structure consists of three folders: Collectors, Scripts and Tests. We will focus on two folders. The first one is Scripts, where, using SCL programming language, we can create our scripts. The second one is Tests, where, using scripts created earlier, tests are created. Additionally, in the upper part of the application, there's a menu containing several useful options, as well as extensive Help.

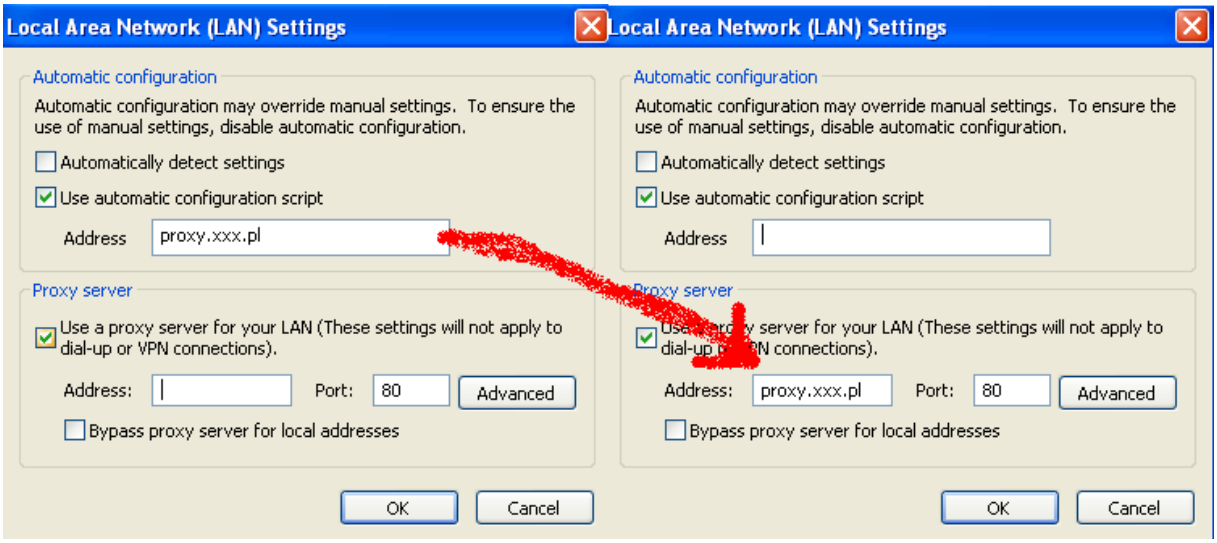
Picture 1 Main Screen of OpenSta

To create a new test, first we have to create new scripts – it can be made by choosing **File ->New Script** from top Menu. When we double click on the script, a recording window will open.



Picture 2. Creating Script in OpenSta

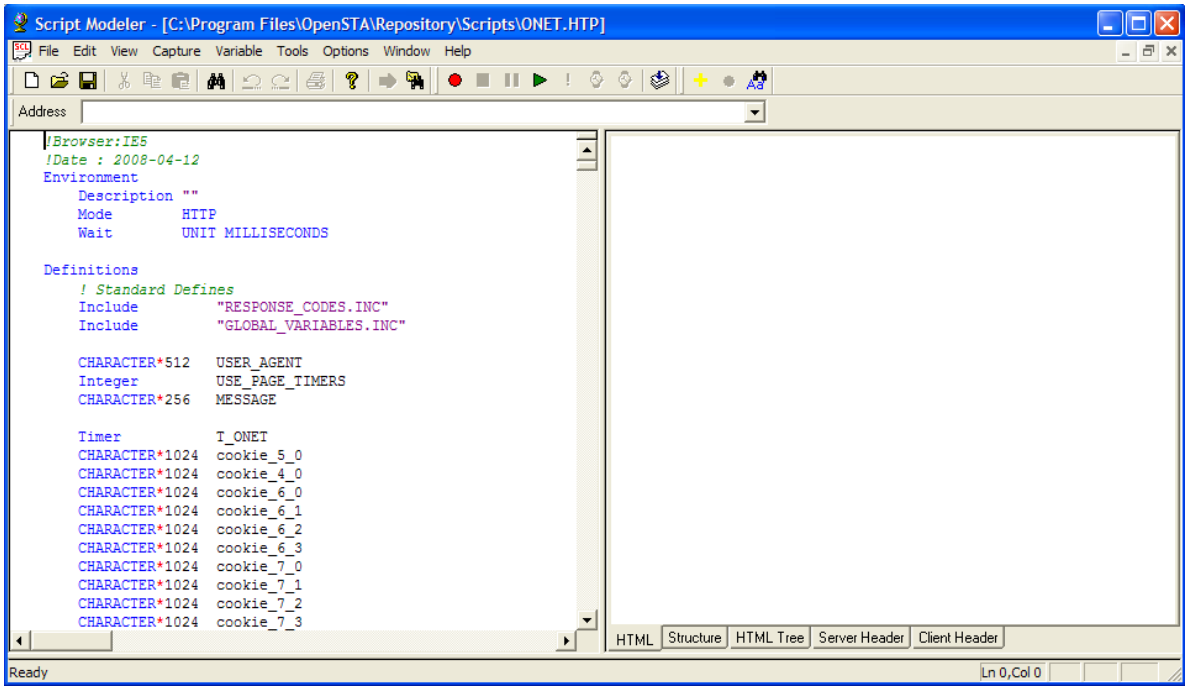
OpenSta supports both HTTP and HTTPS protocols what distinguishes it from other tools – like JMeter - and is one of its main advantages. If we are using Proxy server for internet connection, we have to properly configure the browser. In order to do it, click **Options -> Browser** from the menu (applies to IE 8).



Picture 3. Proxy configuration

If we want to connect to remote Server, it is possible by using settings from **Options->Gateway** menu. Unfortunately, you have to set up Proxy like shown on picture 3. In other case OpenSta will not be able to run.

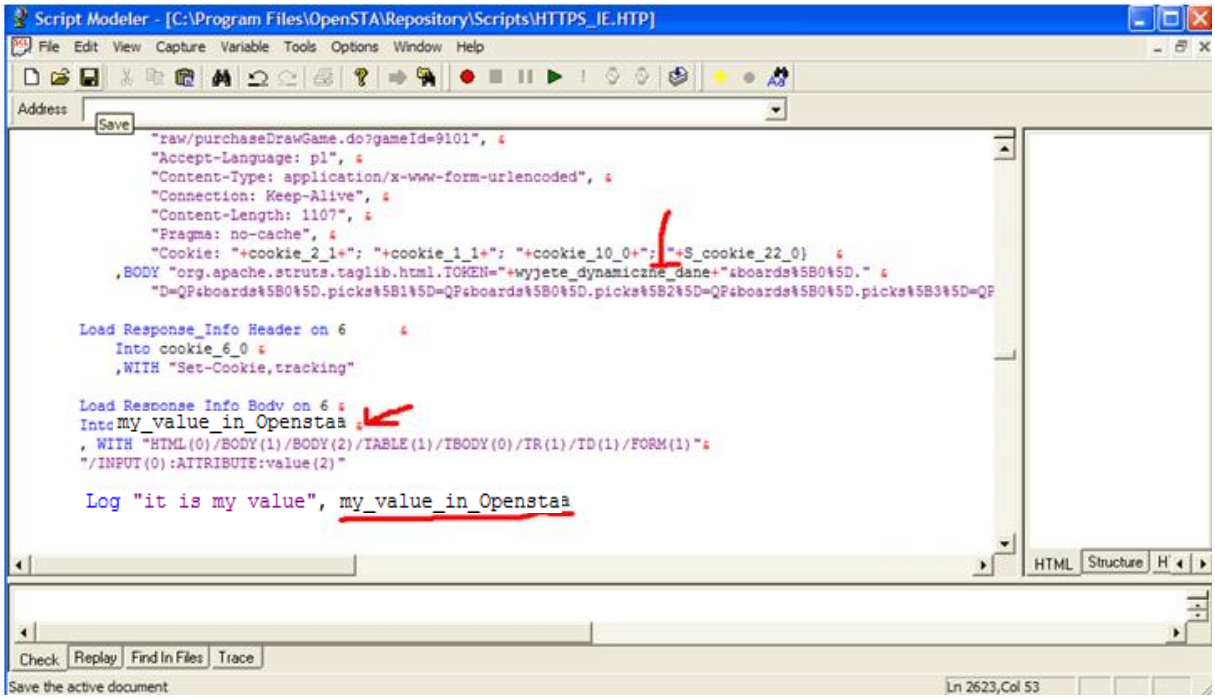
Another function which is worth to mention, is the possibility to declare variables from **Variable ->Create** menu. It allows to prepare variables used for storing important values before recording script will take place. In order to start recording, click red button or choose **Capture->Record** from the menu. After that a browser will open and we can go through our planned test scenario. To finish recording, simply close the browser or click on the **Stop** button.



Picture 4. Source code view in OpenSta commander

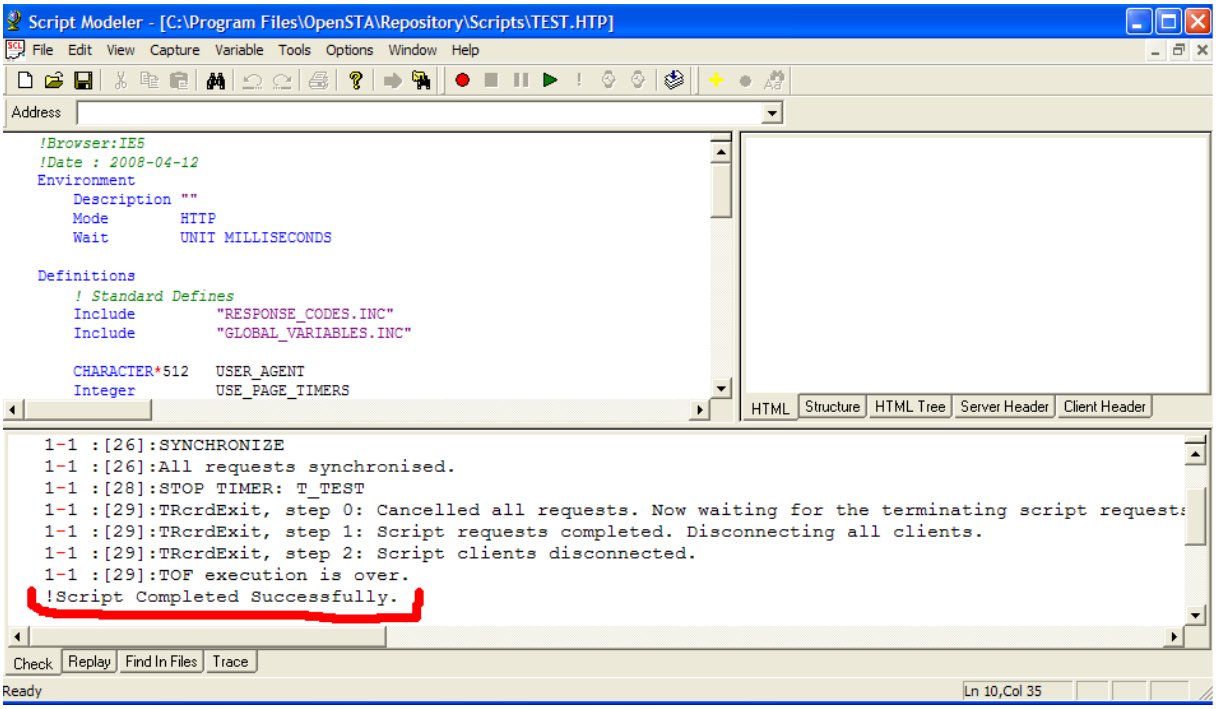
On the left side we can see the source code of recorded scenario, declared variables, environment information and other data captured by OpenSta. As it was mentioned before, source code is written in SCL (Structured Control Language). If any modifications need to be made after recording, it's possible by modifying the script. After finishing the code must be compiled, and, if there are no errors, we can run the script by clicking green **Play** button.

SCL language is not the simplest or easiest one, but if we take a closer look, we can notice some dependencies, such as adding data to variables. On the next picture we can see a piece of code which was modified in order to retrieve data from HTML generated by JavaScript – OpenSta has recorded it as a permanent value although it is generated dynamically. Running the script caused errors as the "old" value does not conform with the current one, generated by the page – hence it's necessary to put the value as variable.



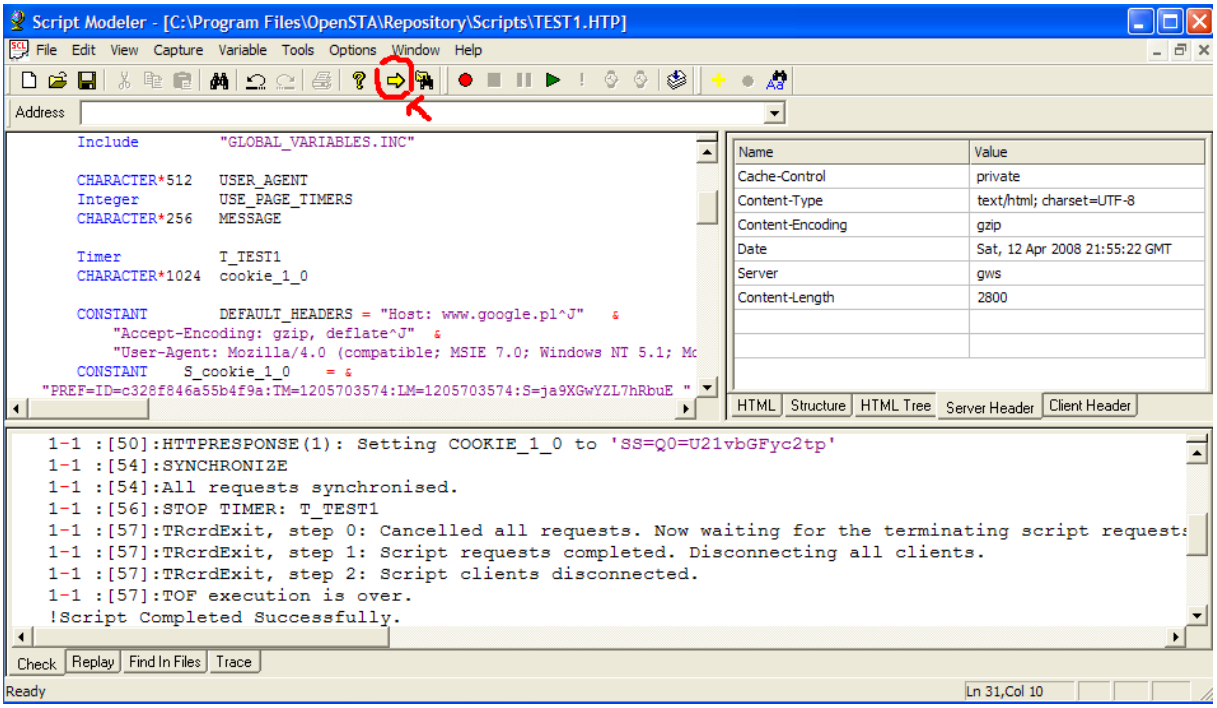
Picture 5. Modyfing source code

If the script contains no errors, the following message should be displayed:



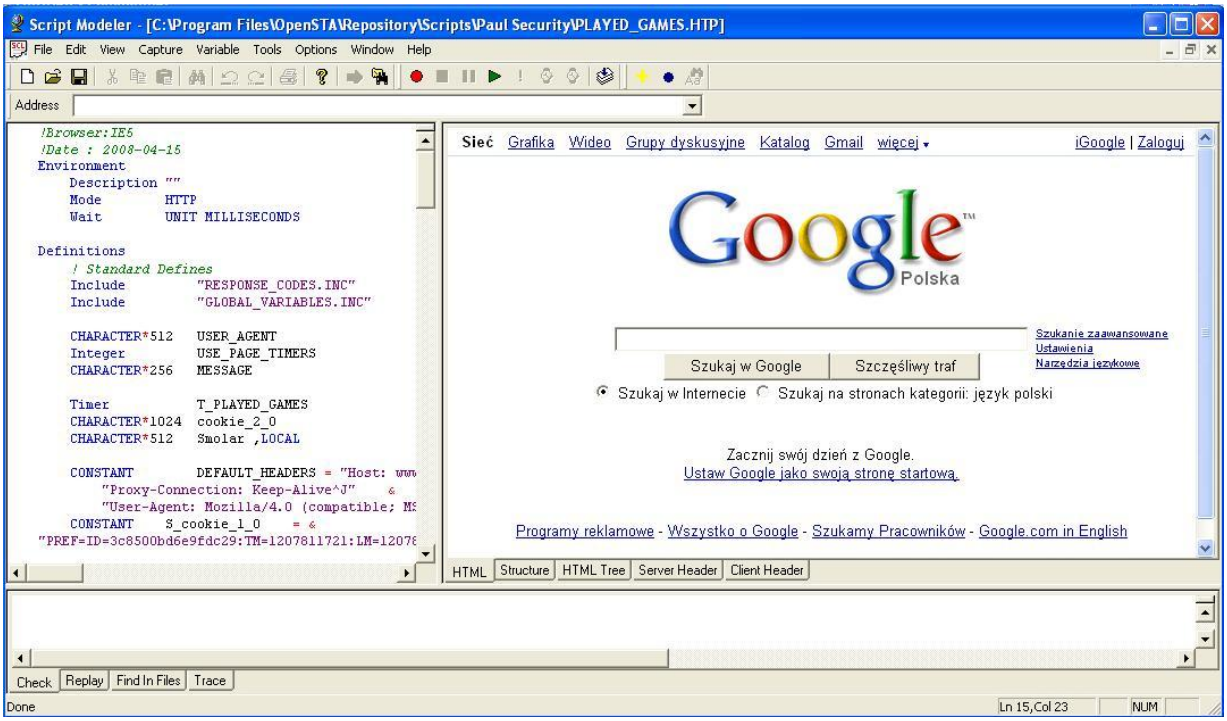
Picture 6. Compiling code

After compilation, we can see „Get” function in the code – it can be highlighted using cursor – and when yellow arrow appears in top menu, click on it. Then the preview of recorded page will be shown.



Picture 7. Get function

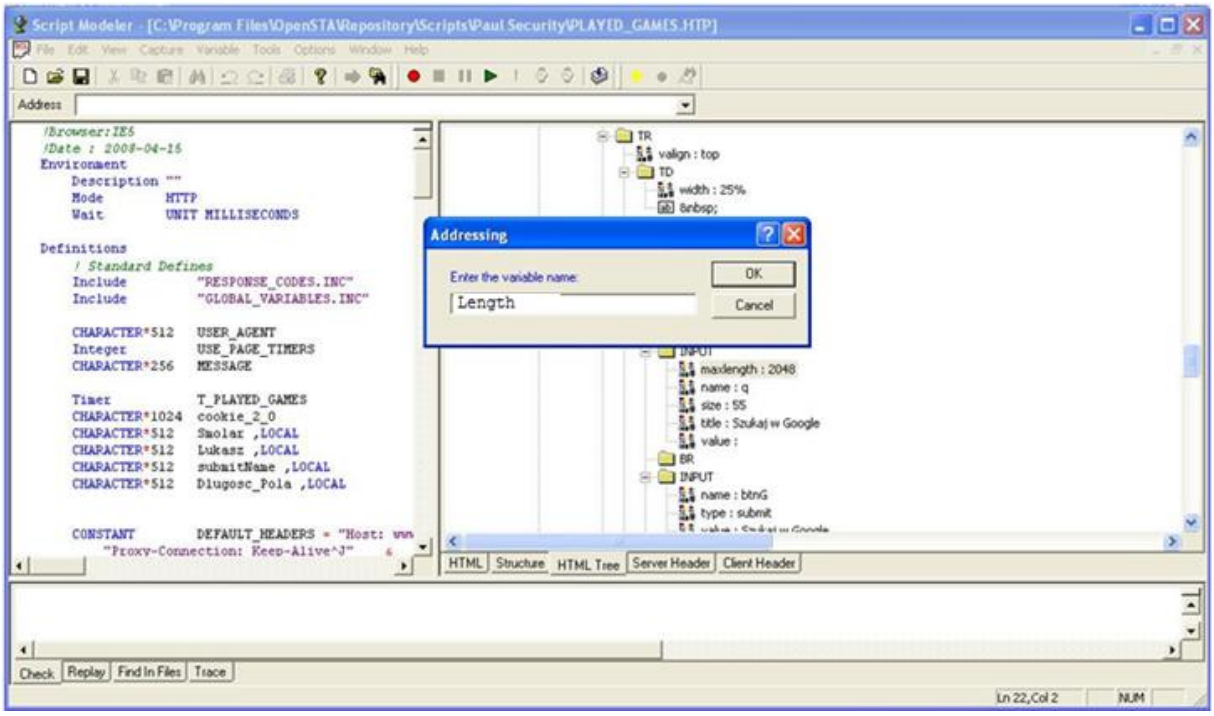
Now we can see HTML structure, server data and other useful information. We can also go to HTML code by right clicking on the interesting part of code and create variables, which can be used to retrieve data from DOM module (it can be used to retrieve information such as details created by JavaScript).



Picture 8. Page preview

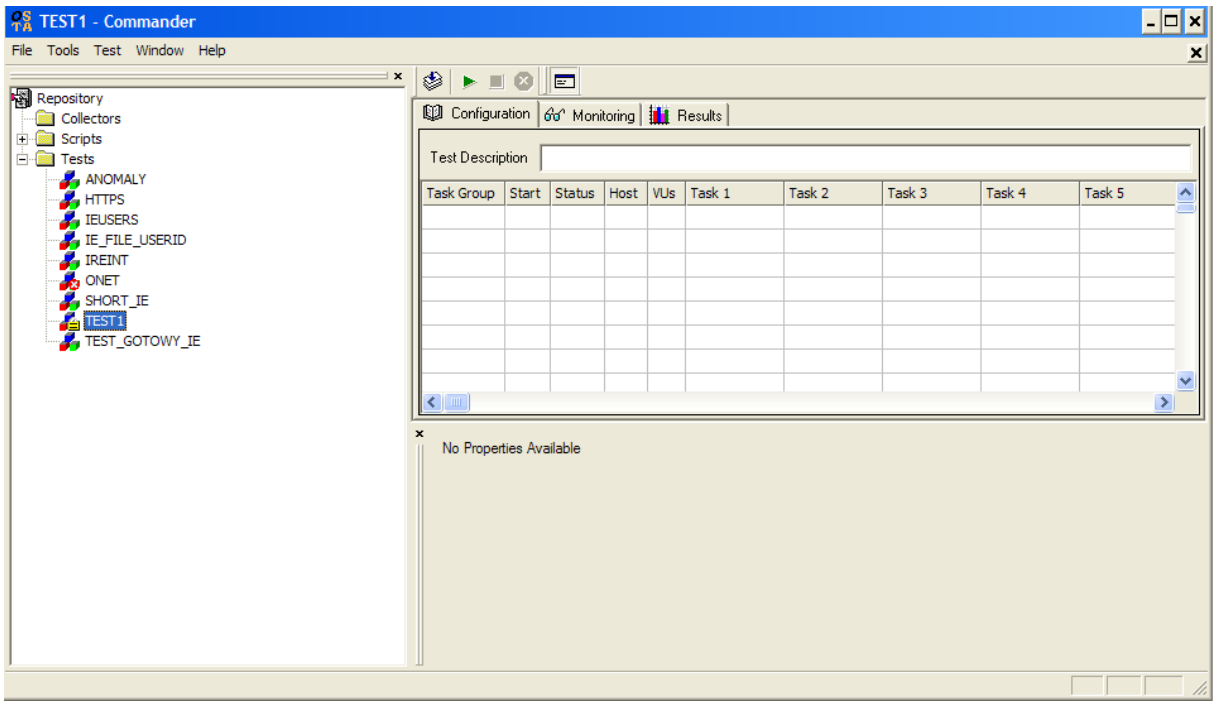
By clicking on **structure** tab, we can preview the page to see the entire structure and all related elements and their values.

Additionally, by clicking HTML tab, we can find a specific value in the code, and after that, with right click, create variable containing this value and put it in the source code.



Picture 10. Creating variable with value taken from HTML tree

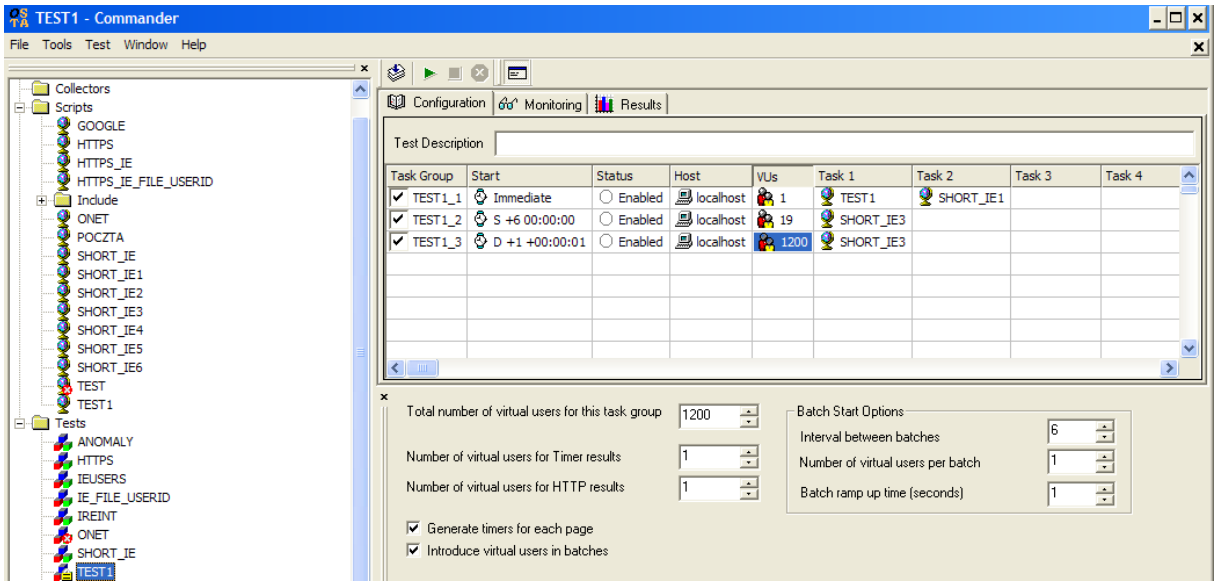
Now let's focus on test creation. Return to main page and choose **File-> New Test-> Tests** from the top menu**.** After that double click on the "Test" icon shown in the tree - testing menu will be shown.



Picture 11. Testing menu

Next, choose the script you have just created and drag it to the **Task** area. This area is split to columns, where we can add several scripts. It means that if we add some scripts to the same row, but

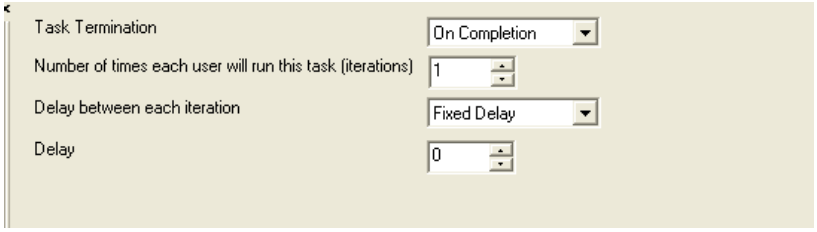
to different columns, they all will be executed at the same time. If we add some scripts to different rows, they will be executed in certain order (Ascending).



Picture 12. Test configuration

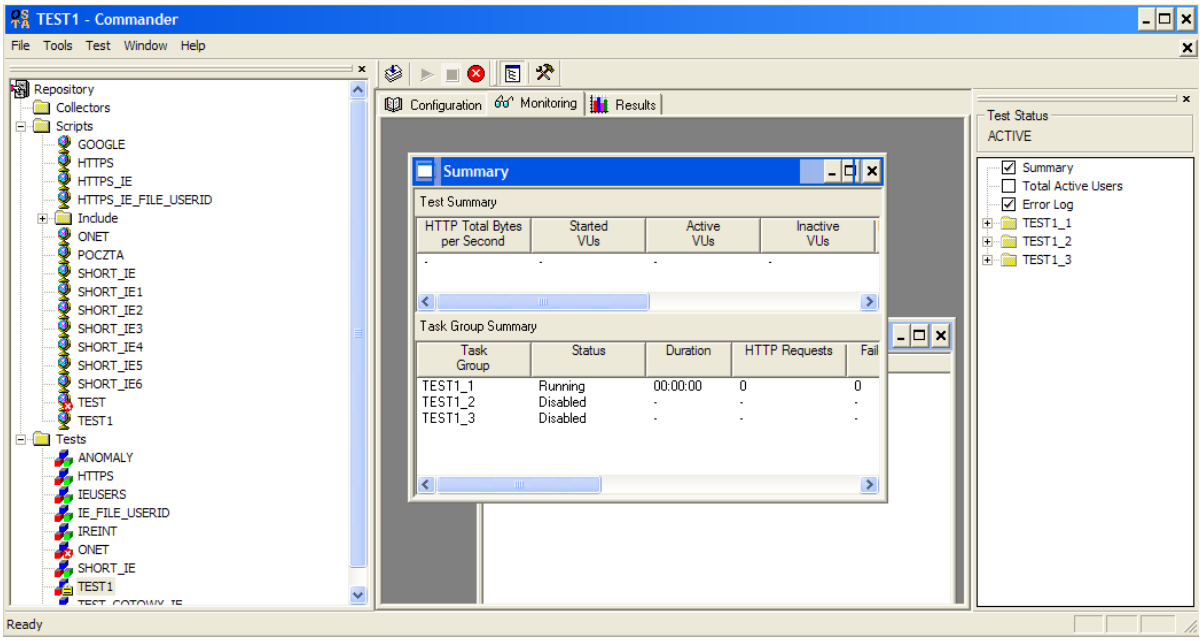
In Test configuration there are some options, which can be controlled. **Start** option is used for setting begin time of executed tests (Immediate, delayed and planned). Next setting is the number of virtual users set for every single task. Moreover we can split users in categories, i.e. Total amount of VU =1200, but 2 of them are assigned to “Timer Results” and to “HTTP results” respectively.

There is also possibility to define users directly in the source code (using loop) – i.e. for the purpose of creating account via bank website and checking how long does it take to accomplish the task. We can check this data in some reports which will be described later. Now click on the **Run** button and the test will start.



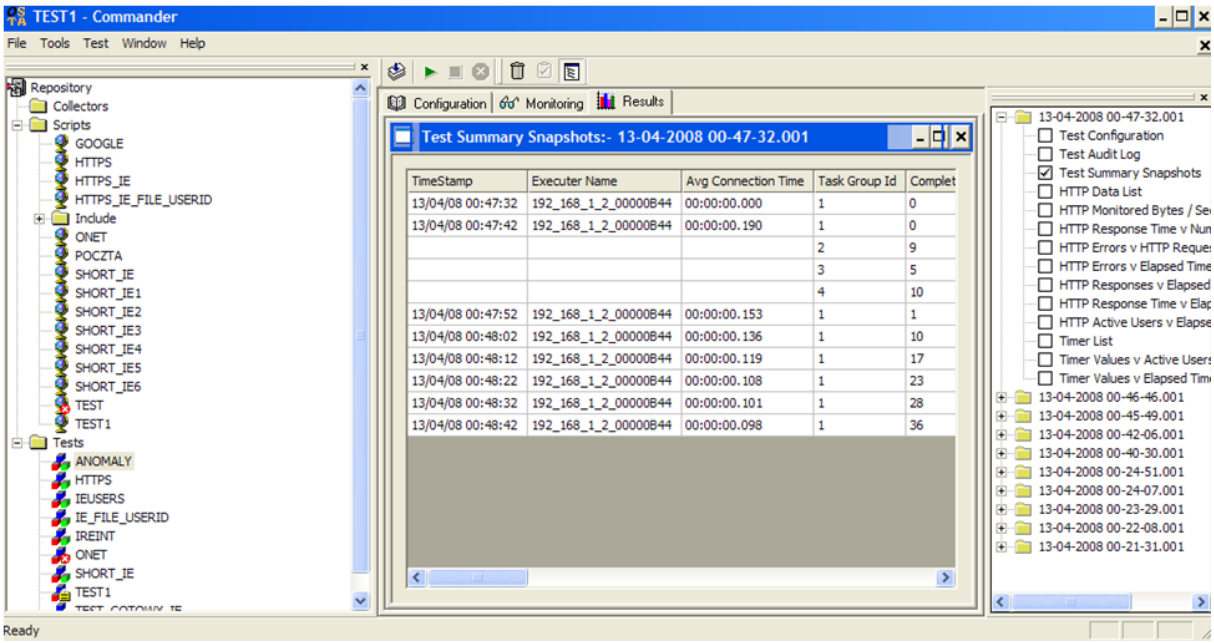
Picture 13. Task settings

During test execution we can monitor the progress and observe what is happening. This can be done via **Monitoring** tab**.** We can also check **Summary** tab and see how the test has been performed. (Unfortunately these reports are not user friendly since the information is not clear enough). In addition we can check whether any errors occurred during test execution.



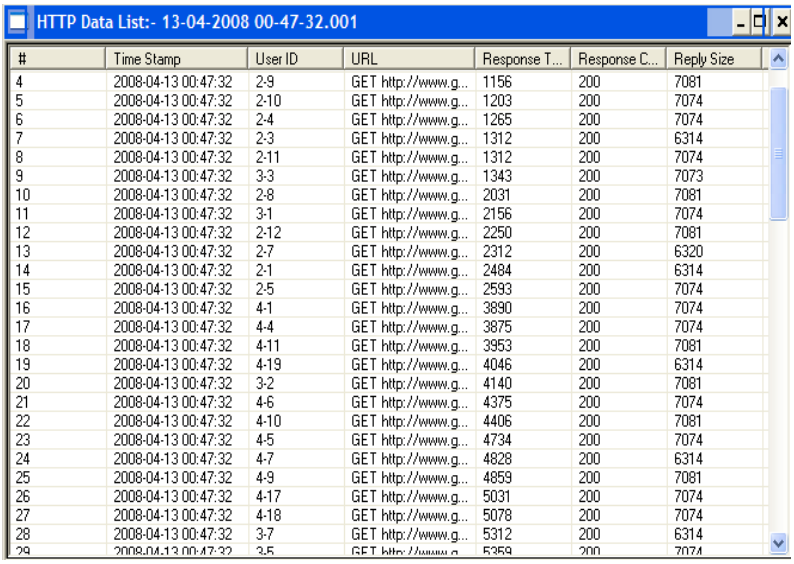
Picture 14. Report visible during test execution

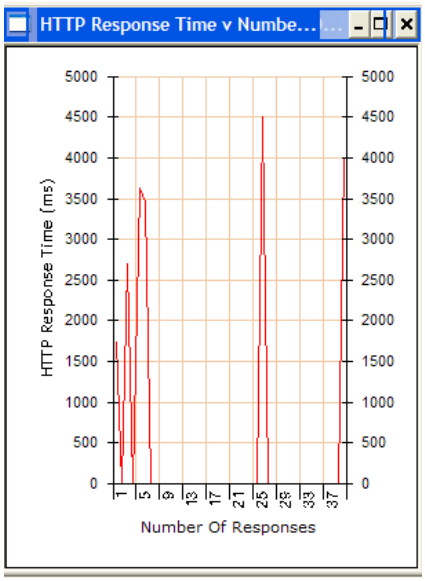
After test execution, we can go to the **Results** tab. There are plenty of reports from which we can find out some useful details like configuration, performance time, etc. We can also analyze graphs and see all other needed information.



Picture 15. Results after test execution

The appropriate report can be selected from the tree available on right side of the application window (see picture 15). I recommend to take a look at each option (each type of report) in order to find the information we need and to decide which is the most appropriate for your purposes. On the graphs we can see dependencies between tests and time responses for our scripts. If any errors occur when executing the script, the error log will be automatically created. Below we can see sample reports.





Picture 16. Sample reports

The functionality described above is just a description of creating simple test scenario. My intention was to show only simple example of using OpenSta. As you know there can be complex test scripts created with many functions inside. Moreover, we can set up many configuration options and – as the result - tests will be run on different environments. During test execution we do not see how the script is going through web pages. All operations are executed in the background.

Like all other applications, OpenSta has also some disadvantages. I would like to describe some of them. One of them is the fact, that recording via HTTPS sometimes doesn't work - in this case we need to record on HTTP and modify source code to adjust it to HTTPS. This problem is caused by some errors in the OpenSta application and should be resolved in new releases. Another problem with OpenSta is that when we launch the browser, starting page sometimes does not appear. To solve this problem, proxy server needs to be set up again – exactly like shown in the pictures above. Another error which I've noticed is the problem with the length of characters while recording a script. Due to this fact compilation was failed. Fortunately, if any error occurs, we are able to see in which line of the code it happened . In my case I reduced the length of string and it started to work properly. (problem with browser , cut to IE7 – String "*User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) ; .NET CLR 1.1.4322; InfoPath.2; MEGAUPLOAD 2.0; .NET CLR 2.0.50727)"*).

Summary

Summarizing, I think that OpenSta it is a great free tool for performance testing. Despite some disadvantages, it has many useful functions allowing us to easily check web load, stress and performance of our application. Moreover this tool is continuously developed and maintained. It's possible to join forum for OpenSta users and ask for help in case of any problems. I think nowadays it's worth to check and try new tools available on the market as they can support our work and optimize it. If you'd like to get more knowledge about OpenSta, please refer to the following resources:

http://www.opensta.org/

http://portal.opensta.org/