EXPERIMENT – 10

Software Testing and Quality Assurance

Abstract

Study of any open source testing tool (OSTA).

Syeda Reeha Quasar

14114802719

EXPERIMENT – 10

Aim:

Study of any open source testing tool (OSTA).

Theory:

Introduction

OpenSTA supplies versatile Test development software that enables you to create and run Tests tailor-made for the environment you are assessing. The contents and structure of a Test will depend on the type of test you are conducting, the nature of the system you are targeting and the aims of your performance test.

OpenSTA supports the creation of HTTP/S load Tests that include Scripts which supply the load element and may also include Collectors which are used to record additional performance data. You can also use OpenSTA to develop Collector-only used Tests for monitoring production environments. It is possible to use the same Test in both these scenarios.

Running a Test with both Script and Collector Task Groups enabled allows you to test and record the performance of a WAE during development. After the WAE goes live, Script-based Task Groups can be disabled and the Test re-run within a production environment. This enables you to generate useful comparative performance results and to continuously monitor the target WAE.

The example in this tutorial works through the creation of an HTTP/S load Test which includes Script-based and Collector-based Task Groups.

OpenSTA Architecture

OpenSTA supplies a distributed software testing architecture based on CORBA which enables you to create then run Tests across a network. The OpenSTA Name Server configuration utility is the component that allows you to control your Test environment.

After installing OpenSTA you will notice that the OpenSTA Name Server is running indicated by , in the Windows Task Bar. This component must be running before you can run a Test.

If no icon appears click **Start > Programs > OpenSTA > OpenSTA NameServer.**

If the OpenSTA Name Server stops the Name Server Configuration utility icon

appears , in the Task Bar. You can start it by right-clicking and selecting Start Name

Server from the menu.

Commander

Commander is the Graphical User Interface that runs within the OpenSTA Architecture and functions as the front end for all Test development activity. It is the program you need to run in order to use OpenSTA.

Launch Commander

Click **Start > Programs > OpenSTA > OpenSTA Commander.**

The Commander Interface

Commander combines an intuitive user interface with comprehensive functionality to give you control over the Test development process, enabling you to successfully create and run HTTP/S performance Tests.

Use the menu options or work from the Repository Window to initiate the creation of Collectors and Tests. Right-click on the predefined folders in the Repository Window and choose from the functions available.

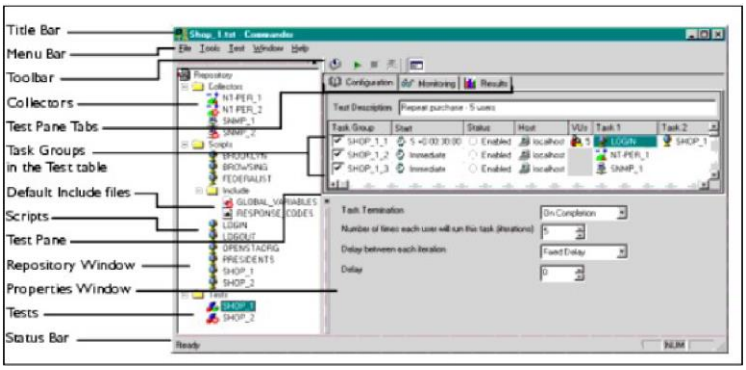
Work within the Main Window of Commander to create Collectors and Tests. The Main Window houses the Repository Window and supplies the workspace for Test creation using the Test Pane, and Collector creation using the Collector Pane. Use Script Modeler to create the Scripts you need.

After you have created or edited a Test or Collector in the Main Window it is automatically saved when you switch to another procedure or exit from Commander.

Commander Interface Features

The Commander interface is divided up into three primary areas:

- Commander Toolbars and Function Bars.
- The Repository Window.
- The Commander Main Window.



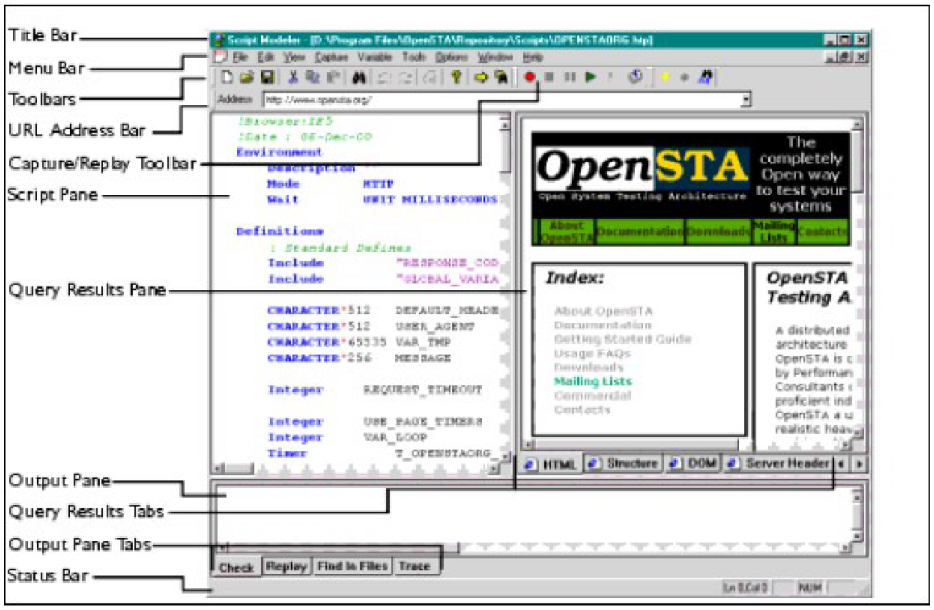
Script Modeler

Script Modeler supplies versatile Script creation and modeling functionality. Use the menu bar and right-click menu options to create and model Scripts.

After you create a Script or when you open one, it is displayed in the Script Pane on the left-hand side of the main window. It is represented using SCL code which enables you to model it using the menu options or directly by keying in the SCL commands you need.

The Query Results Pane is used to display Web site responses. HTML information is recorded during the same Web session as the corresponding Script and is directly related to it, which enables additional modeling options.

The Output Pane. Displays the results of Script compilation. Scripts need to be compiled after modeling to check the validity.



The Test Pane

Use the Test Pane to create and edit a Test, then apply the Task Group settings you require to control how they behave during a Test-run. Run and monitor the Test-run then display your results for analysis.

The Test Pane is displayed in the Main Window when you open a Test by double clicking a new Test , or an existing Test , in the Repository Window.

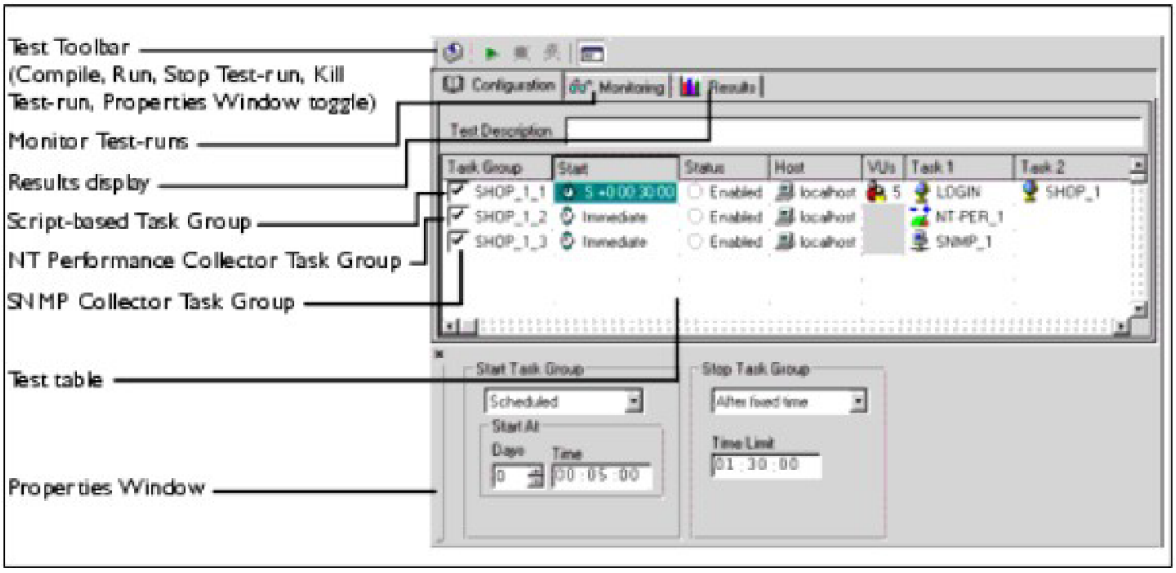
The Test Pane comprises three sections represented by the following tabs:

- **Configuration:** This is the default view when you open a Test and the workspace used to develop a Test. Use it in combination with the Repository Window to

select and add Scripts and Collectors. It displays the Test table which has categorized column headings that indicate where Script and Collector Tasks can be placed and the Task Group settings that apply to the contents of the Test. Select a Task Group cell to view and edit the associated settings using the Properties Window displayed below the Test table.

- **Monitoring:** Use this tab to monitor the progress of a Test-run. Select the display options you want from the Monitoring Window, including a Summary and data for individual Task Groups.
- **Results:** Use this tab to view the results collected during Test-runs in graph and table format. Use the Results Window to select the display options available which are dependent on the type of Test you are running.

Test pane Features



The Test Development Process

The Test development process typically includes the following procedures:

- Create a Test
- Add a Script to a Test
- Add Collectors to a Test
- Define Task Group settings, these include:
- Edit the Task Group Schedule Settings
- Select the Host used to Run a Task Group
- Specify the Number of Virtual Users to run a Script-based Task Group

- Edit the Number of Script Iterations and the Delay Between Iterations
- Save and Close a Test

Result:

The open source testing tool was successfully studied and explained.

Viva Questions

Q1. Why choose Open Source Performance Test tool?

Open source load testing tools are often maintained by some of the big developers and are free to use with their source code available to customize as required.

Advantages of using open source load testing tools

- No initial investment is needed, as it is free to use.
- The source code is available to extend the functionality based on internal use cases.
- Managed by top developers and open source communities.

Q2 Explain what is performance testing?

Performance testing can be used to analyze various success factors such as response times and potential errors. With these performance results in hand, you can confidently identify bottlenecks, bugs, and mistakes – and decide how to optimize your application to eliminate the problems.

Q3. Mention different types of performance testing?

1. Stress Testing.
2. Spike Testing.
3. Load Testing.
4. Endurance Testing.
5. Volume Testing.
6. Scalability Testing.

Q4. List out what are the common performance problem does user face?

Some common problems are:

1. Poorly written code.
2. Non-optimized databases.
3. DNS, firewall, and network connectivity.
4. Troublesome third-party services.

Q5: List out some of the performance testing tool?

Some performance testing tools are:

1. WebLOAD.
2. LoadNinja.
3. HeadSpin.
4. ReadyAPI Performance.
5. LoadView.
6. Keysight's Eggplant.
7. Apache JMeter.
8. LoadRunner.

Q6: List out some common performance bottlenecks?

Some common performance bottlenecks are:

1. CPU Utilization.
2. Memory Utilization.
3. Network Utilization.
4. Software Limitation.
5. Disk Usage.