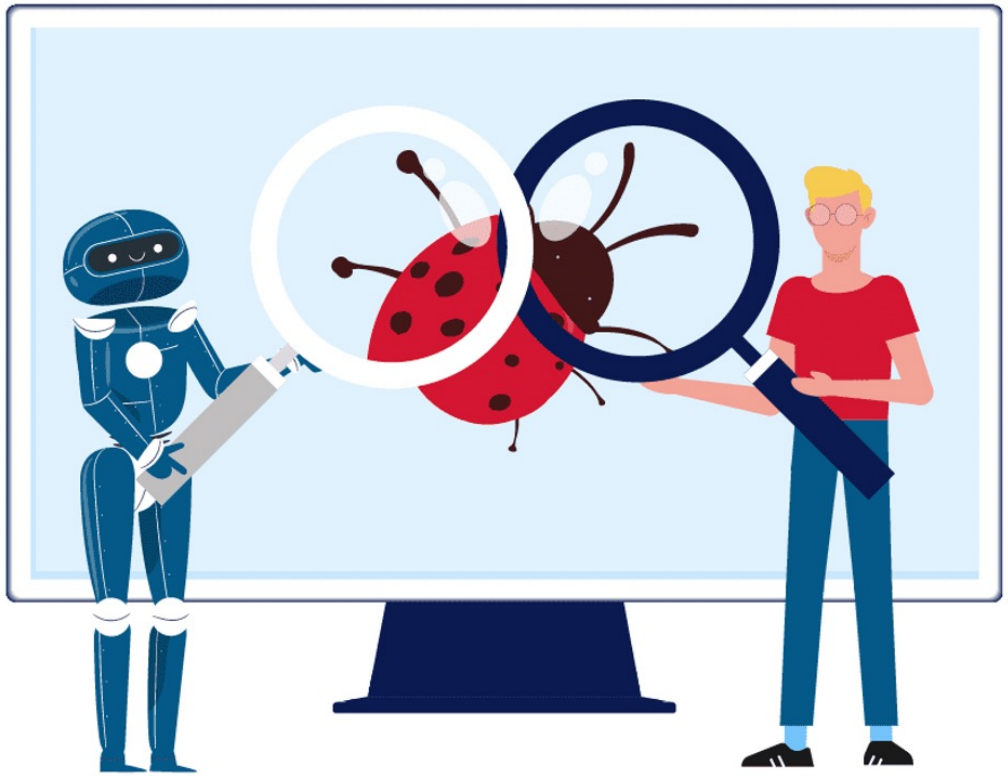
Automated Software Testing

Spring, 2023

Yi Xiang xiangyi@scut.edu.cn

Contents

- Why we need Automated Testing
- What is Automated Testing
- Automation Tools
  - Selenium
  - Jmeter



1. Why we need Automated Testing

1) Manual software testing is slow, error-prone, and hard to repeat accurately:
   - Testing All workflows/all fields/all negative scenarios is time consuming
   - Become boring and hence error prone as human testers lose concentration.
   - The testing finishes for the day when the testers go home

2) Yet software testing needs to be fast, accurate, and repeatable:
   - Performed frequently without a time penalty.
   - Test results can be relied on as a quality indicator.
   - Be repeatable to allow for regression testing.

2. What is Automated Testing?

It’s a software testing method that compares expected and actual results of test cases with the help of special automation testing tools

- Automated execution of tests, or collections of tests
- Automated collection of results
- Automated evaluation of results
- Automated reports
- Automated measurement of test coverage

The Key Stages of Automated Testing

Consider the following steps:

1. Define your goals and create your strategy in automation testing;
2. Choose automation testing tools to execute a test task like Selenium, Egg Plant, etc.;
3. Set up test environment;
4. Develop test scripts on the basis of testing requirements;
5. Test execution and result analysis.

The Key Differences Between Manual and Automated Testing

Benefits and drawbacks automated testing has：

Pros: （High ROI & Faster GoTo market）

- Supports execution of repeated Test Cases
- Aids in testing a large Test Matrix
- Enables parallel execution
- Encourages unattended execution
- Improves accuracy thereby reducing human-generated errors
- Saves time and money

Cons:

- Automation instruments are usually expensive;
- It is ineffective in testing user experience in applications;
- Coding knowledge and experience are a must.

The Key Differences Between Manual and Automated Testing

Manual testing is commonly used when:

- You have a short-term project with a low budget;
- You need to complete exploratory testing;
- You are going to run ad-hoc testing, which is usually unplanned, and gathering testing insights are important in this process;
- You should test app usability and measure how valuable the user experience is for your end users.

The Key Differences Between Manual and Automated Testing

Automation testing is suitable when:

- You know a specific number of regression tests for your project;
- You should test server models and web servers in load and stress testing;
- You have a big project where it's essential to test several software functionalities;
- You run performance testing to check software quality attributes like scalability, reliability, speed, etc.

3. Selenium

- Selenium is one of the most popular Automated Testing suites.
- Selenium is designed to support and encourage Automation Testing of functional aspects of web-based applications and a wide range of browsers and platforms.

  - It's an open-source
  - It has a large user base and helping communities
  - It has multi-browser and platform compatibility
  - It has active repository developments
  - It supports multiple language implementations

Selenium

Selenium automates browsers. https://www.selenium.dev

Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that.

Getting Started

Selenium WebDriver

If you want to create robust, browser-based regression automation suites and tests, scale and distribute scripts across many environments, then you want to use Selenium WebDriver, a collection of language specific bindings to drive a browser - the way it is meant to be driven.

Selenium IDE

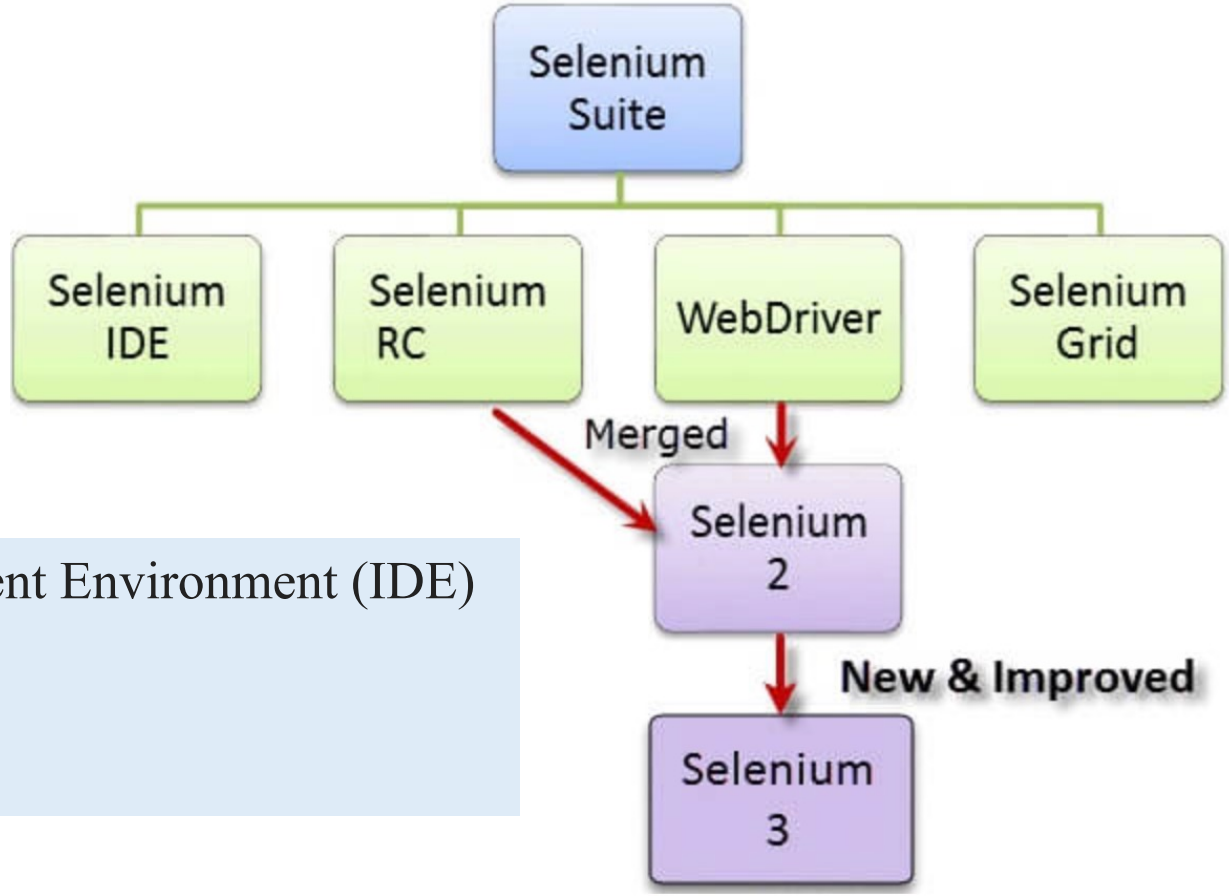
If you want to create quick bug reproduction scripts, create scripts to aid in automation-aided exploratory testing, then you want to use Selenium IDE; a Chrome, Firefox and Edge add-on that will do simple record-and-playback of interactions with the browser.

Selenium Grid

If you want to scale by distributing and running tests on several machines and manage multiple environments from a central point, making it easy to run the tests against a vast combination of browsers/OS, then you want to use Selenium Grid.

Selenium Components

- Selenium is a package of several testing tools, hence it is referred to as a Suite.
- Each of these tools is designed to cater to different testing and test environment requirements.



- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- WebDriver
- Selenium Grid

Selenium Components --- Selenium Core

- Selenium was created by Jason Huggins in 2004 at ThoughtWorks.
- As repetitious Manual Testing of their application was becoming more and more inefficient, a JavaScript program that would automatically control the browser's actions was created.
- First named as the "JavaScriptTestRunner."
- Open-source and later re-named as Selenium Core.

Selenium IDE (Selenium Integrated Development Environment)

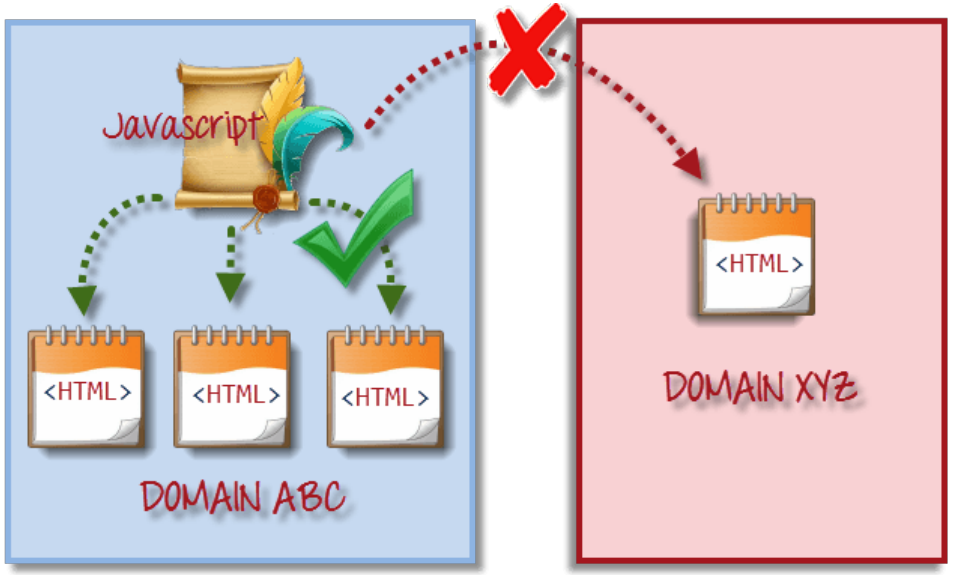
- Shinya Kasatani of Japan created Selenium IDE, a Firefox extension that can automate the browser through a record-and-playback feature.

- Further increase the speed in creating test cases.
  - The Simplest framework in the Selenium suite and the easiest one to learn.

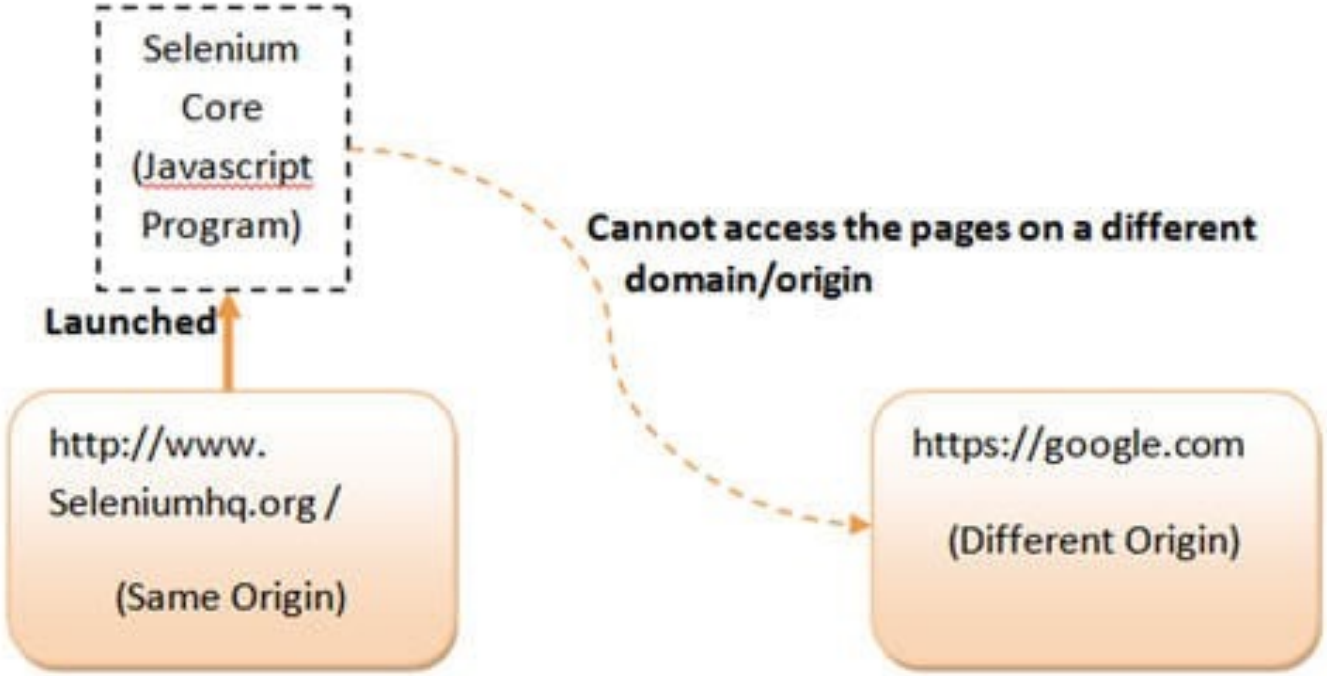
  - Doesn't support iterations and conditional statements
  - Doesn't support loops
  - Doesn't support error handling
  - Doesn't support test script dependency

Selenium Remote Control (Selenium RC)

- Testers using Selenium Core had to install the whole application under test and the web server on their own local computers because of the restrictions imposed by the **same origin policy**.

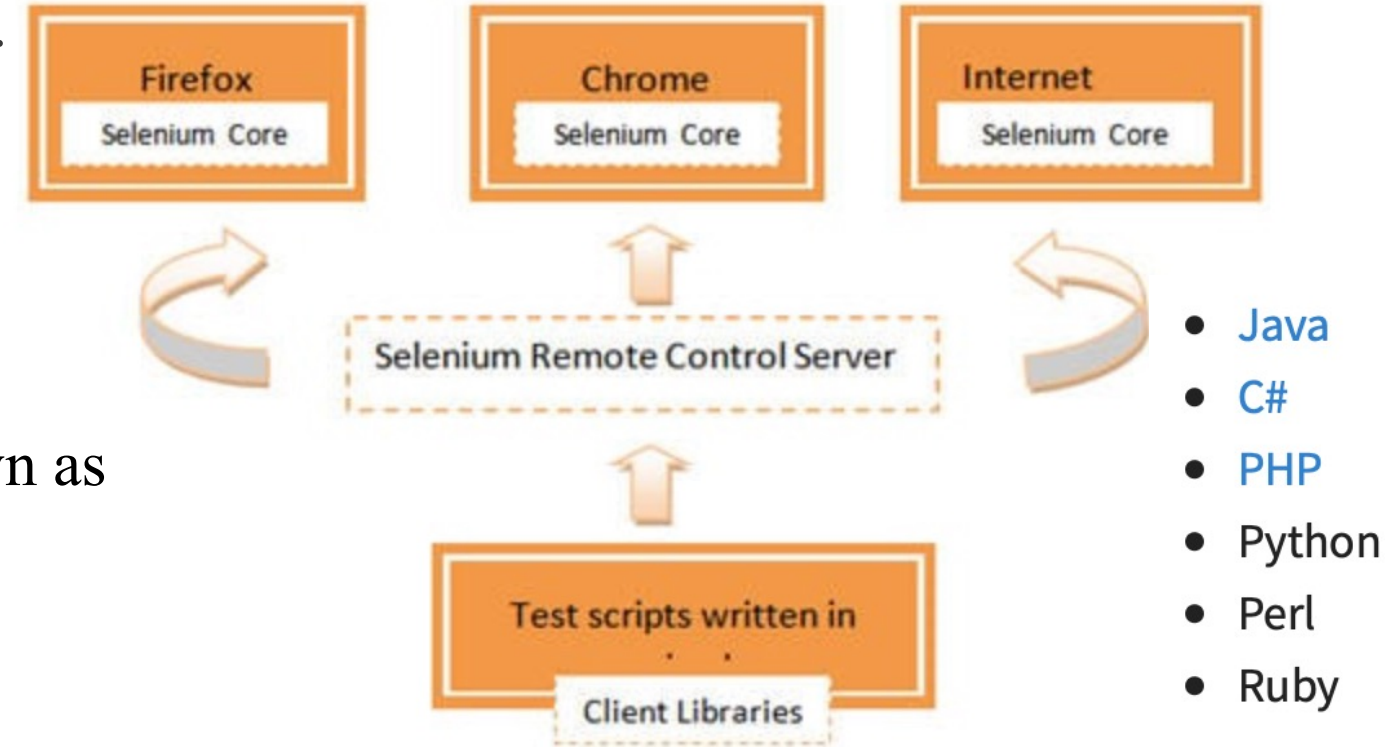


under same origin policy, a Javascript program can only access pages on the same domain where it belongs. it cannot access pages from different domains



Selenium Remote Control (Selenium RC)

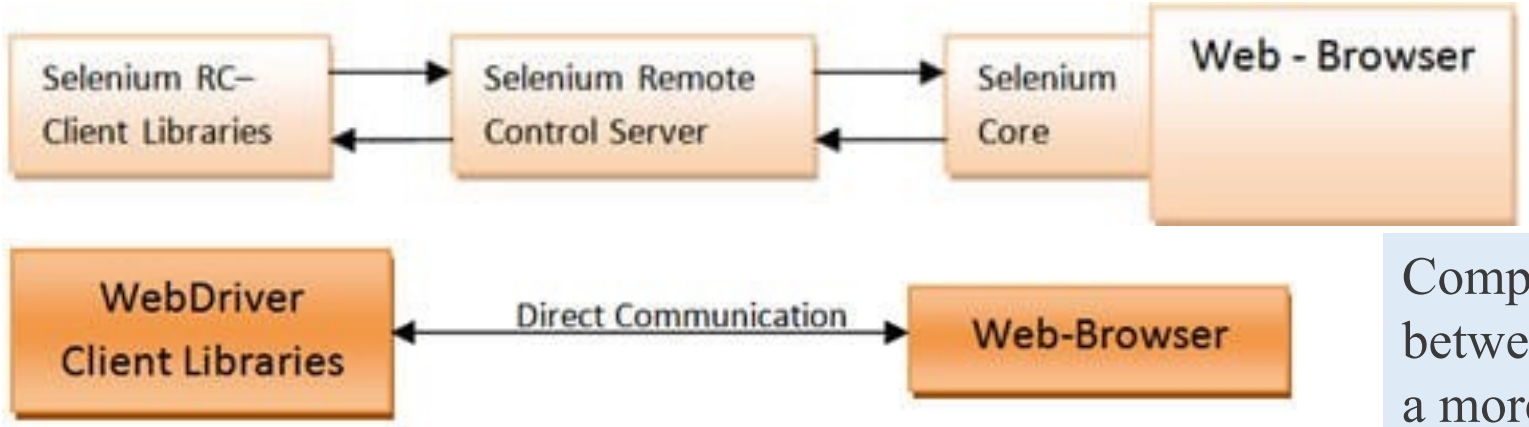
- ThoughtWork's engineer, Paul Hammant, created a tool written in Java to allow a user to construct test scripts for a web-based application in any programming language he/she chooses.
- Selenium RC came as a result to overcome the various disadvantages incurred by ***Selenium IDE or Core***.



This system became known as the Selenium Remote Control or **Selenium 1**.

Selenium WebDriver

- WebDriver allows a user to perform web-based automation testing. WebDriver is a different tool altogether that has various advantages over Selenium RC.
  - Directly communicates with the web browser from the OS level and uses its native compatibility to automate, which is faster than other tools of Selenium
  - Supports a wide range of web browsers, programming languages and test environments.
  - Supports efficient handling mechanisms for complex user actions like dealing with dropdowns, Ajax calls, switching between windows, navigation, handling alerts etc.



Compatibility analysis between Selenium RC and WebDriver, a more powerful Selenium ***2***

Selenium Grid

- Selenium Grid is a tool to run parallel tests across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

- Features:
  - Enables simultaneous running of tests in multiple browsers and environments.
  - Saves time enormously.
  - Utilizes the hub-and-nodes concept. The hub acts as a central source of Selenium commands to each node connected to it.

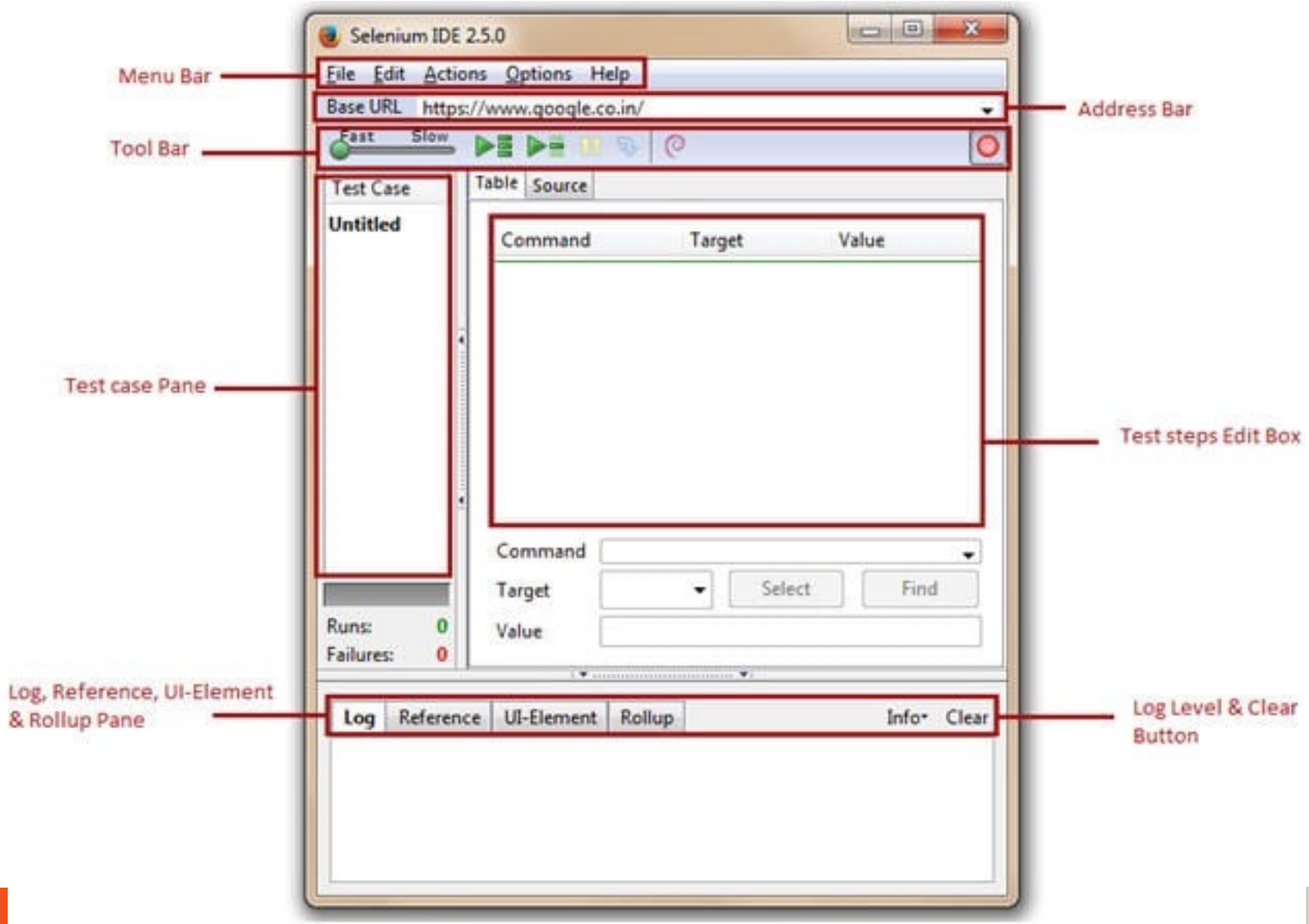
How to Choose the Right Selenium Tool

Selenium IDE	• To learn about concepts on automated testing and Selenium, including:
	  • Selenese commands such as type, open, clickAndWait, assert, verify, etc.
	  • Locators such as id, name, xpath, css selector, etc.
	  • Executing customized JavaScript code using runScript
	  • Exporting test cases in various formats.
	• To create tests with little or no prior knowledge in programming.
	• To create simple test cases and test suites that you can export later to RC or WebDriver.
	• To test a web application against Firefox and Chrome only.

How to Choose the Right Selenium Tool

WebDriver	• To use a certain programming language in designing your test case. • To test applications that are rich in AJAX-based functionalities. • To execute tests on the HtmlUnit browser. • To create customized test results.
Selenium Grid	• To run your Selenium RC scripts in multiple browsers and operating systems simultaneously. • To run a huge test suite, that needs to complete in the soonest time possible.

1) Testing with Selenium IDE



Creating Selenium IDE Script

#1: Recording – Selenium IDE aids the user to record user interactions with the browser and thus the recorded actions as a whole are termed as Selenium IDE script.

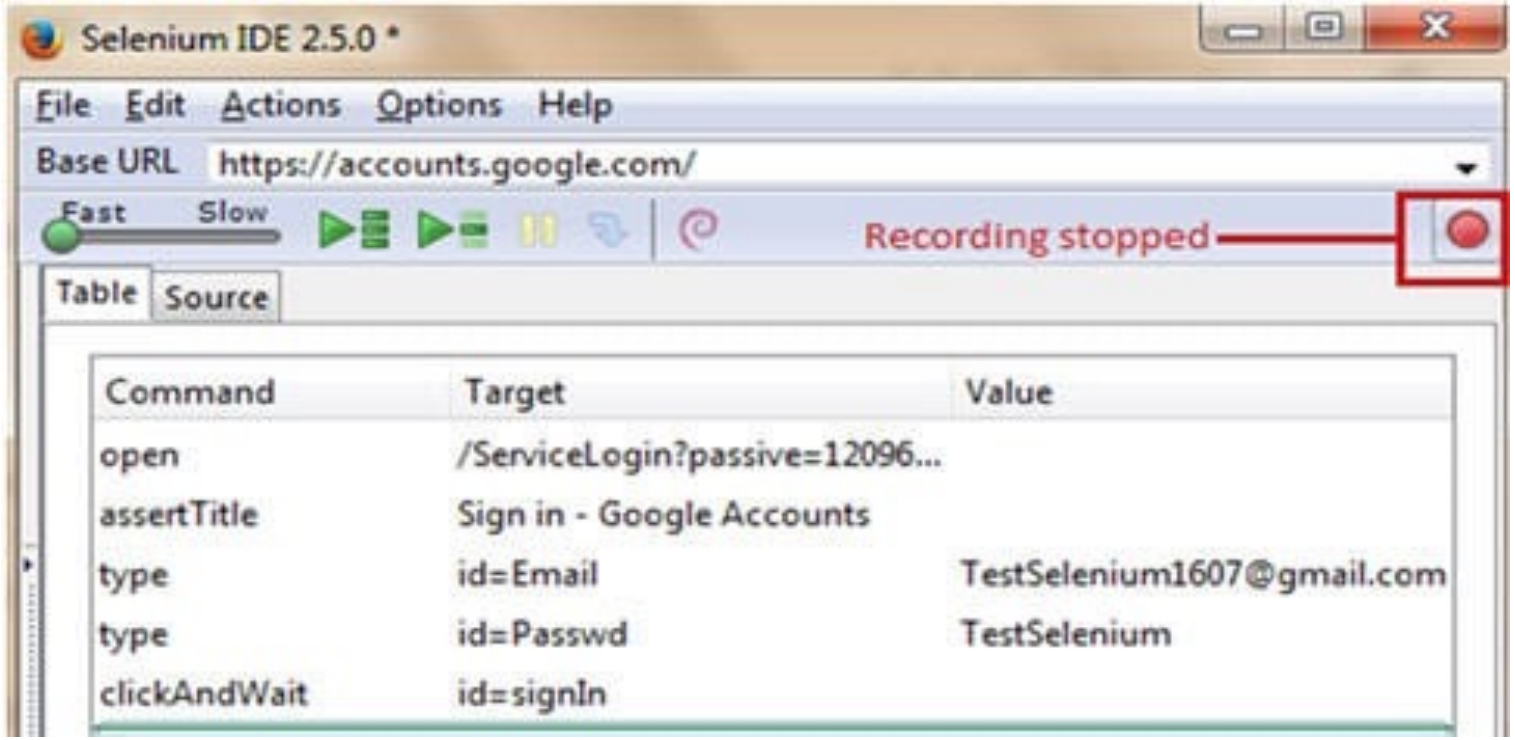
#2: Playing back – In this section, we execute the recorded script so as to verify and monitor its stability and success rate.

#3: Saving – Once we have recorded a stable script, we may want to save it for future runs and regressions

Creating Selenium IDE Script

Scenario

- Open “https://accounts.google.com”.
- Assert Title of the application
- Enter a valid username and password and submit the details to login.
- Verify that the user is redirected to the Home page.



The simulation of the same user actions can be seen in the Selenium IDE test editor.

Selenium IDE Commands

- **Actions**

  commands that interact directly with the application by either altering its state or by pouring some test data.
  - For Example, "type" like a text box, showed on the UI as well.
  - "click" command, user manipulate the state of the application.

- **Accessors**

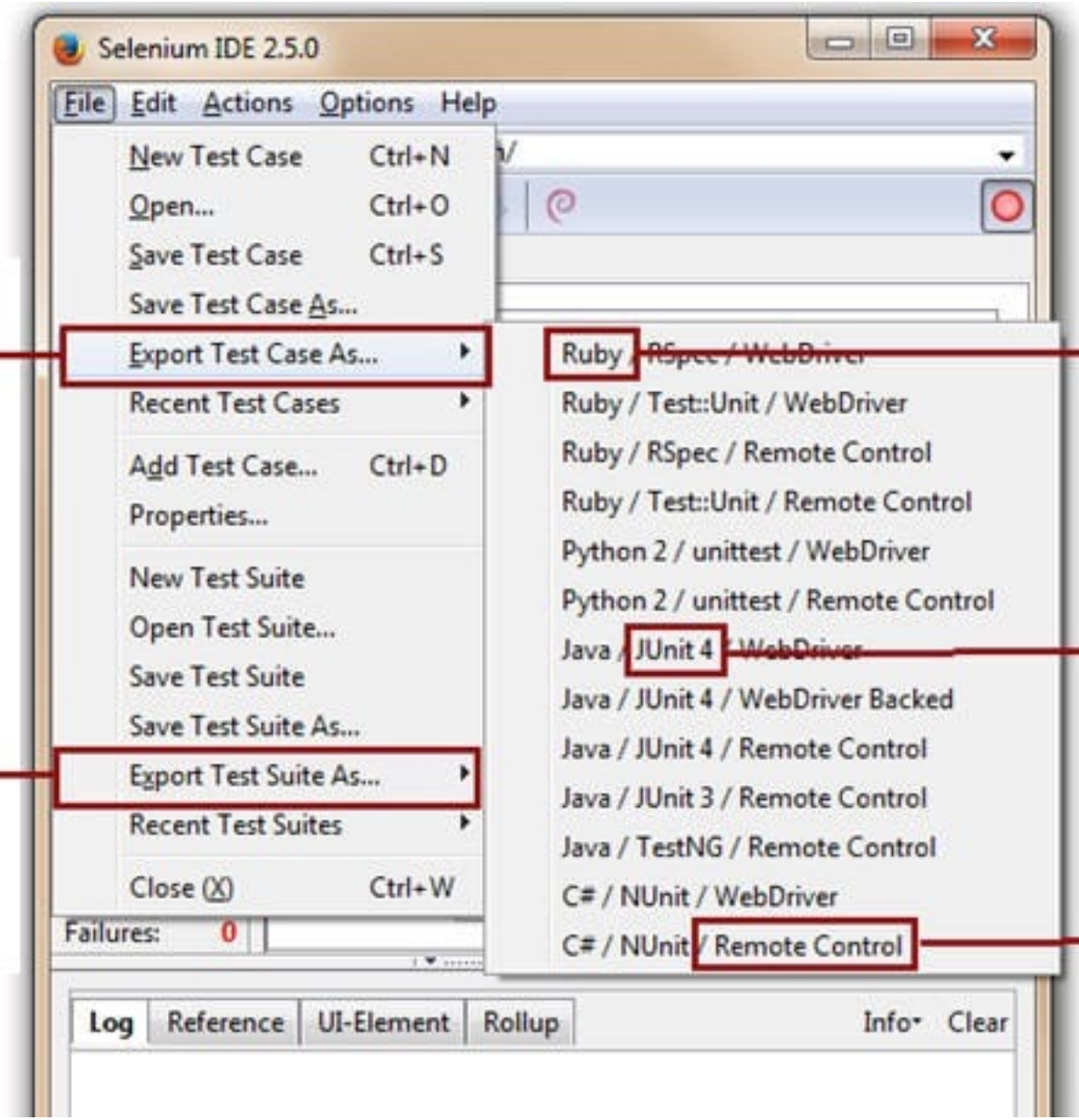
  commands that allow the user to store certain values to a user-defined variable, used to create assertions and verifications.
  - For example, "storeAllLinks" reads and stores all the hyperlinks available within a web page into a user-defined variable.

- **Assertions**

  Assertions are very similar to Accessors as they do not interact with the application directly. Assertions are used to verify the current state of the application with an expected state.

Command	Description	#Arguments
open	Opens a specified URL in the browser.	1
assertTitle, VerifyTitle	Returns the current page title and compares it with the specified title	1
assertElementPresent, verifyElementPresent	Verify / Asserts the presence of an element on a web page.	1
assertTextPresent, verifyTextPresent	Verify / Asserts the presence of a text within the web page.	1
type, typeKeys, sendKeys	Enters a value (String) in the specified web element.	2
Click, clickAt, clickAndWait	Clicks on a specified web element within a web page.	1
waitForPageToLoad	Sleeps the execution and waits until the page is loaded completely.	1
waitForElement Present	Sleeps the execution and waits until the specified element is present	1
chooseOkOnNext Confirmation, chooseCancelOn NextConfirmation	Click on ”OK” or “Cancel” button when next confirmation box appears.	0

IDE test case can be exported for a chosen union of programming language, unit testing framework and tool from the selenium package.



Exports only the current test case

Exports the entire testsuite (all the test cases associated with the current test suite)

The programming laguage format in which the test case would be exported

The testing framework in which the generated code would comply

The tool from the selenium tool suite

2) Testing with WebDriver (Java)

1. Java Installation

2. Eclipse IDE Installation

3. Configuring WebDriver for Java
   - Download the Selenium Java Client Libraries
   - Configuring Libraries with Eclipse IDE (*Configure Build Path and Add External Jars…)*
   - Or Maven etc.

4. Install Driver Server for different web browsers
   - Each browser has a different driver implementation in WebDriver.
   - In WebDriver, a few of the browsers can be automated directly.
   - Some of the web browsers require an external entity(Driver Server) to be able to automate and execute the test script.

Eg: ChromeDriver [ChromeDriver()] / Gecko Driver [FirefoxDriver()]

Test Script Creation

Scenario:

- Launch the browser and open “accounts.google.com”.
- Verify the title of the page and print the verification result.
- Enter the username and Password.
- Click on the Sign in button.
- Close the web browser.

Create a new java class named as “Gmail_Login” :

- Import Statements
- Object Instantiation
- Launching the Web browser
- Fetch the Page Web Element
- Comparison between Expected and Actual Values
- Close the Web Browser

Test Script Creation

Import Statements:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.By;
```

- ***import*** *org.openqa.selenium.WebDriver* – References the WebDriver interface which is required to instantiate a new web browser.
- ***import*** *org.openqa.selenium.firefox.FirefoxDriver* – References the FirefoxDriver class that is required instantiate a Firefox specific driver on the browser instance instantiated using WebDriver interface.
- ***import*** *org.openqa.selenium.WebElement* – References to the WebElement class which is required to instantiate a new web element.
- ***import*** *org.openqa.selenium.By* – References to the By class on which a locator type is called.

Test Script Creation

```
public class Gmail_Login {
/**
* @param args
*/
	public static void main(String[] args) {

// objects and variables instantiation
		WebDriver driver = new FirefoxDriver();
		String appUrl = "https://accounts.google.com";

// launch the firefox browser and open the application url
		driver.get(appUrl);

// maximize the browser window
		driver.manage().window().maximize();

// declare and initialize the variable to store the expected title of the webpage.
		String expectedTitle = " Sign in - Google Accounts ";

// fetch the title of the web page and save it into a string variable
		String actualTitle = driver.getTitle();

// compare the expected title of the page with the actual title of the page and print the result
		if (expectedTitle.equals(actualTitle))
		{
			System.out.println("Verification Successful - The correct title is displayed on the web page.");
		}
		else
		{
			System.out.println("Verification Failed - An incorrect title is displayed on the web page.");
		}
```

Test Script Creation

```
// enter a valid username in the email textbox
                WebElement username = driver.findElement(By.id("Email"));
                username.clear();
                username.sendKeys("TestSelenium");

// enter a valid password in the password textbox
                WebElement password = driver.findElement(By.id("Passwd"));
                password.clear();
                password.sendKeys("password123");

// click on the Sign in button
                WebElement SignInButton = driver.findElement(By.id("signIn"));
                SignInButton.click();

// close the web browser
                driver.close();
                System.out.println("Test script executed successfully.");

// terminate the program
                System.exit(0);
        }
}
```

Locating Web Elements

Web elements are located with the help of the dynamic finders (**findElement(By.locatorType**("locator value")))

Locator Types and their Syntax

Locator Type	Syntax	Description
id	driver.findElement (By.id("ID_of_Element"))	Locate by value of the "id" attribute
className	driver.findElement (By.className ("Class_of_Element"))	Locate by value of the "class" attribute
linkText	driver.findElement (By.linkText("Text"))	Locate by value of the text of the hyperlink
partialLinkText	driver.findElement (By.partialLinkText ("PartialText"))	Locate by value of the sub-text of the hyperlink

Locating Web Elements

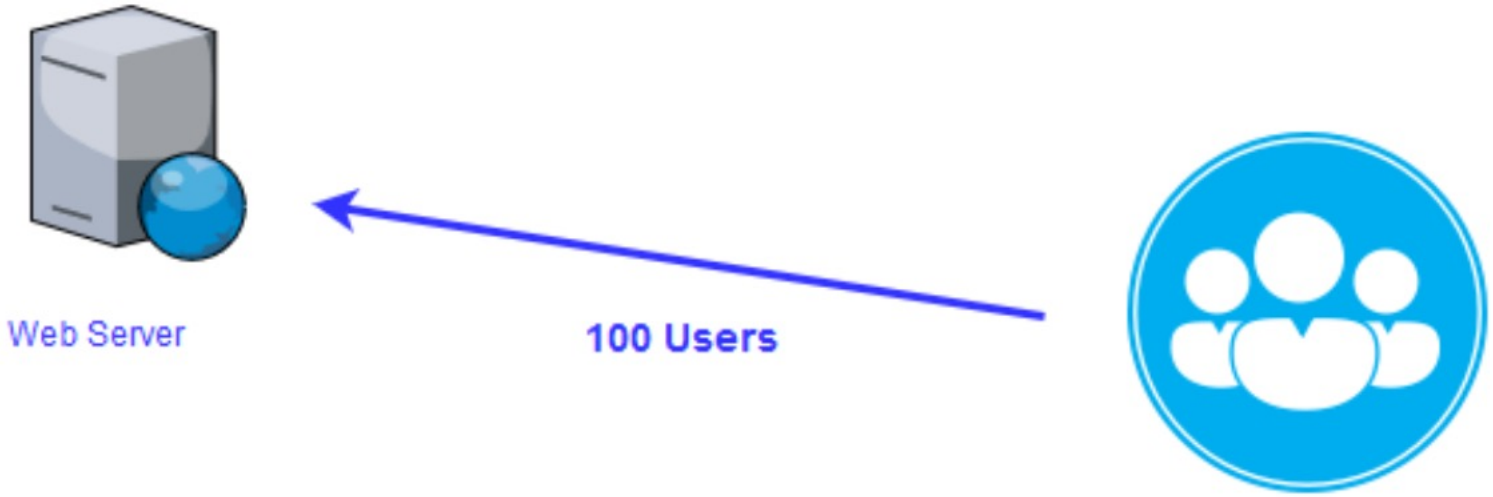
Locator Type	Syntax	Description
name	driver.findElement (By.name (“Name_of_Element”))	Locate by value of the “name” attribute
xpath	driver.findElement (By.xpath(“Xpath”))	Locate by value of the xpath
cssSelector	driver.findElement (By.cssSelector (“CSS Selector”))	Locate by value of the CSS selector
tagName	driver.findElement (By.tagName(“input”))	Locate by value of its tag name

4. Jmeter

- 'Apache JMeter' is an open source, 100% java-based application with a graphical interface, which can analyze and measure the performance of web application or a variety of services.

https://jmeter.apache.org

- Developed by Stefano Mazzocchi of the Apache Software Foundation.
  - Primarily written to test the performance of Apache JServ(currently known as Apache Tomcat project).



JMeter Features

- **Open source application**
- **User-friendly GUI :** simple and interactive GUI.
- **Support various testing approach:** like *Load Testing, Distributed Testing, and Functional Testing*, etc.
- **Platform independent:** run on any environment /workstation that accepts a *Java virtual machine*
- **Support various server types:** highly extensible for different server : *Web, Database, Mail...*
- **Support multi-protocol:** protocols such as *HTTP, JDBC, LDAP, SOAP, JMS, and FTP…*
- **Simulation:** using *virtual users* to generate heavy load against web application under test.
- **Framework:** *multi-threading* framework which allows concurrent and simultaneous sampling of different functions by many or separate thread groups.
- **Remote distributed testing:** *Master-Slave* concept for distributed testing where master will distribute tests among all slaves and slaves will execute scripts against your server.
- **Test result visualization:** viewed in different formats like *graph, table, tree, and report etc.*

How JMeter Works?

JMeter simulates a heavy load on the server or group of servers to test its strength and to analyze the performance of the server when different types of loads are applied to it.

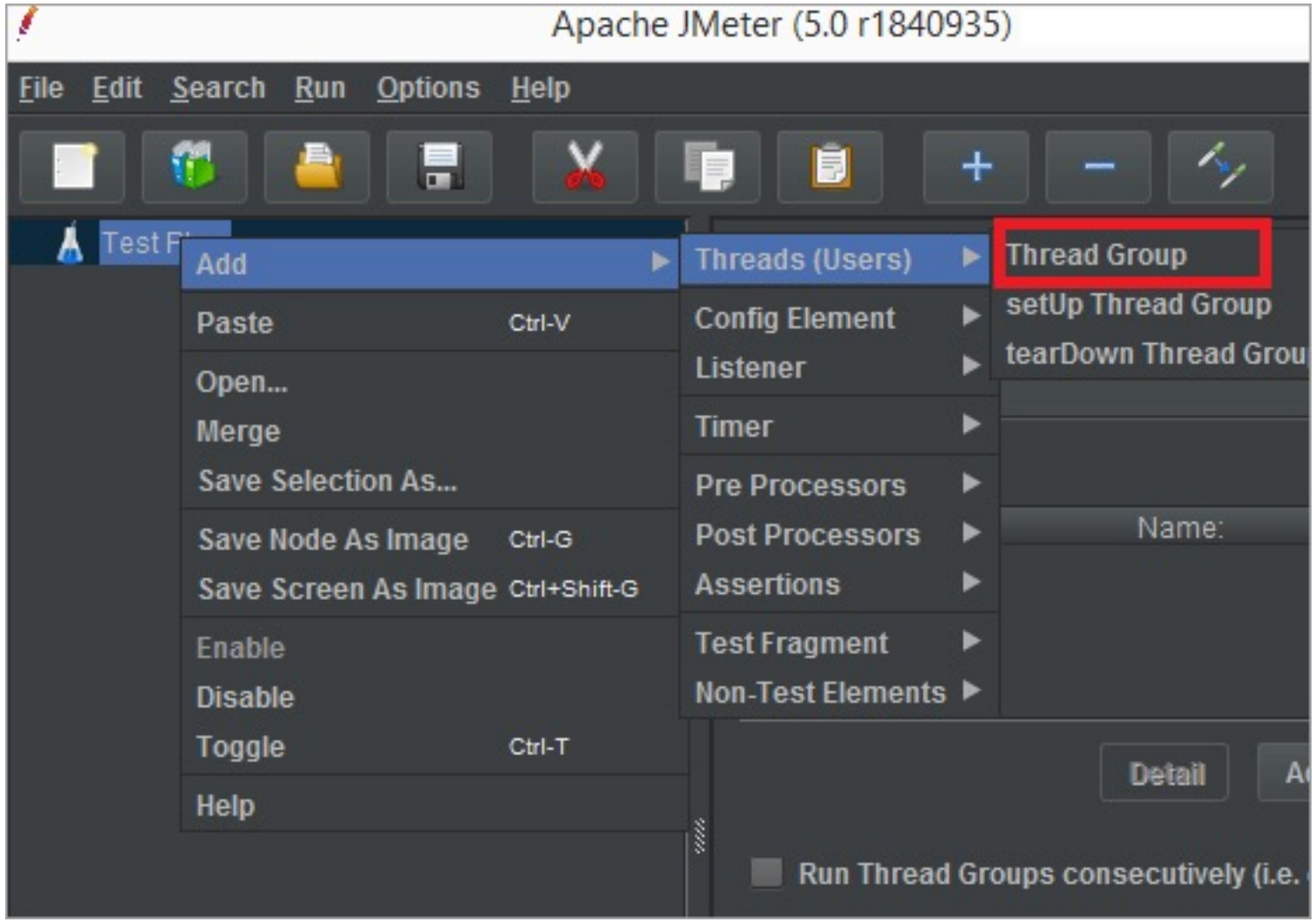
A minimal test will consist of the Test Plan, a Thread Group and one or more Samplers.

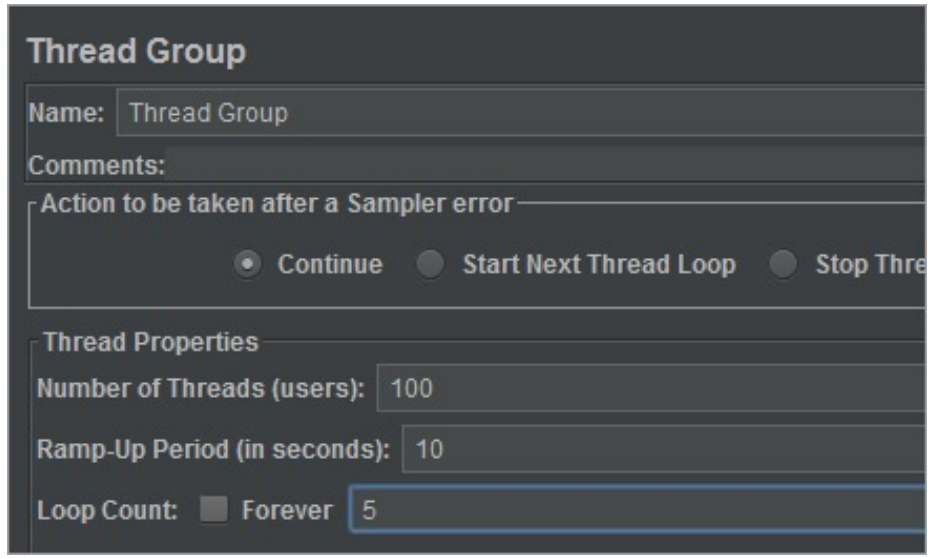
#1) Add Thread Group:

Based on the expected number of users, define :

- threads (each thread represents a user)
- loop count (means how many times the test must repeat)
- Ramp Up Period (means how long to take to "ramp-up" to the full number of threads chosen).

Add Thread Group

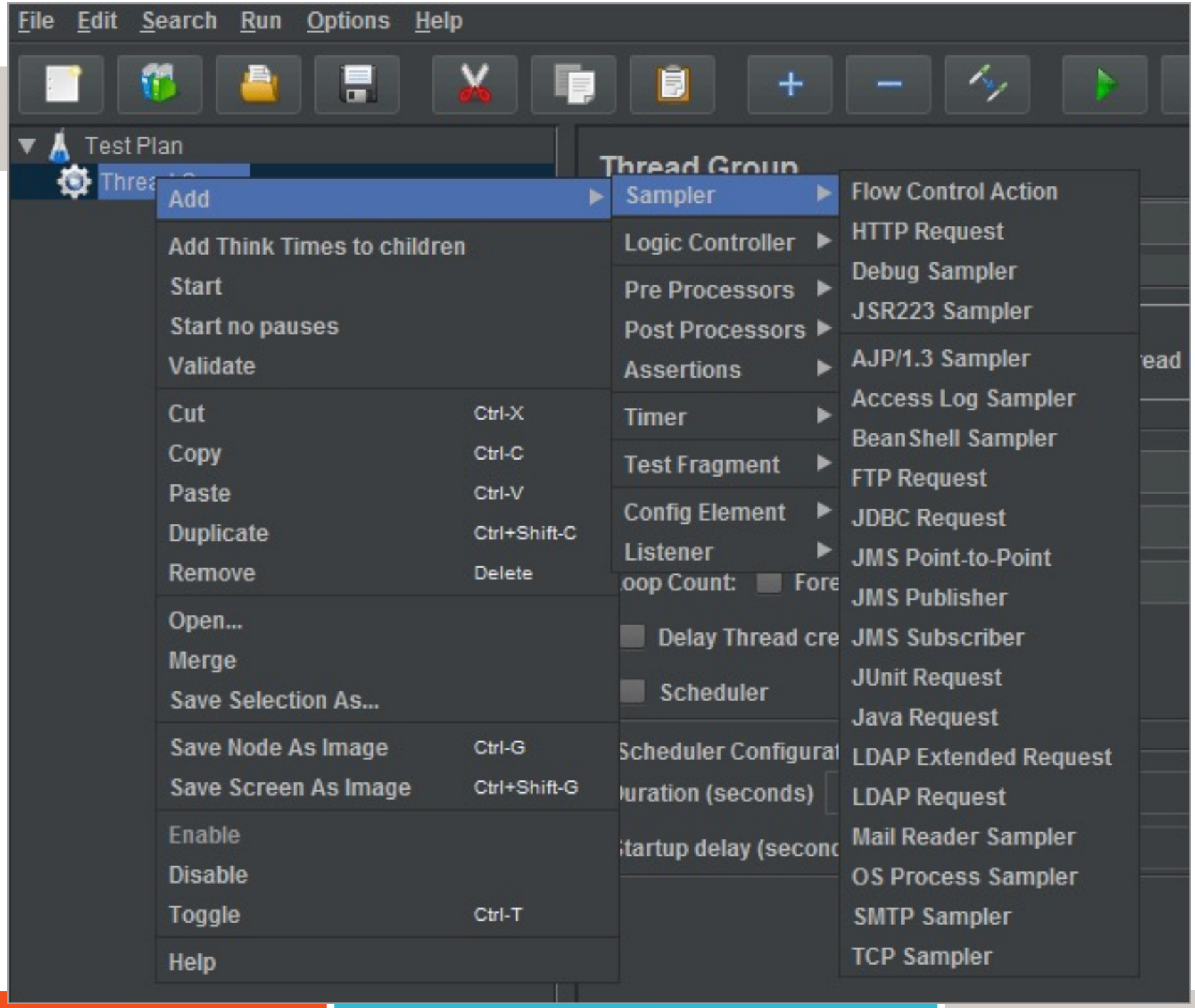




How JMeter Works?

#2) Add JMeter Element :

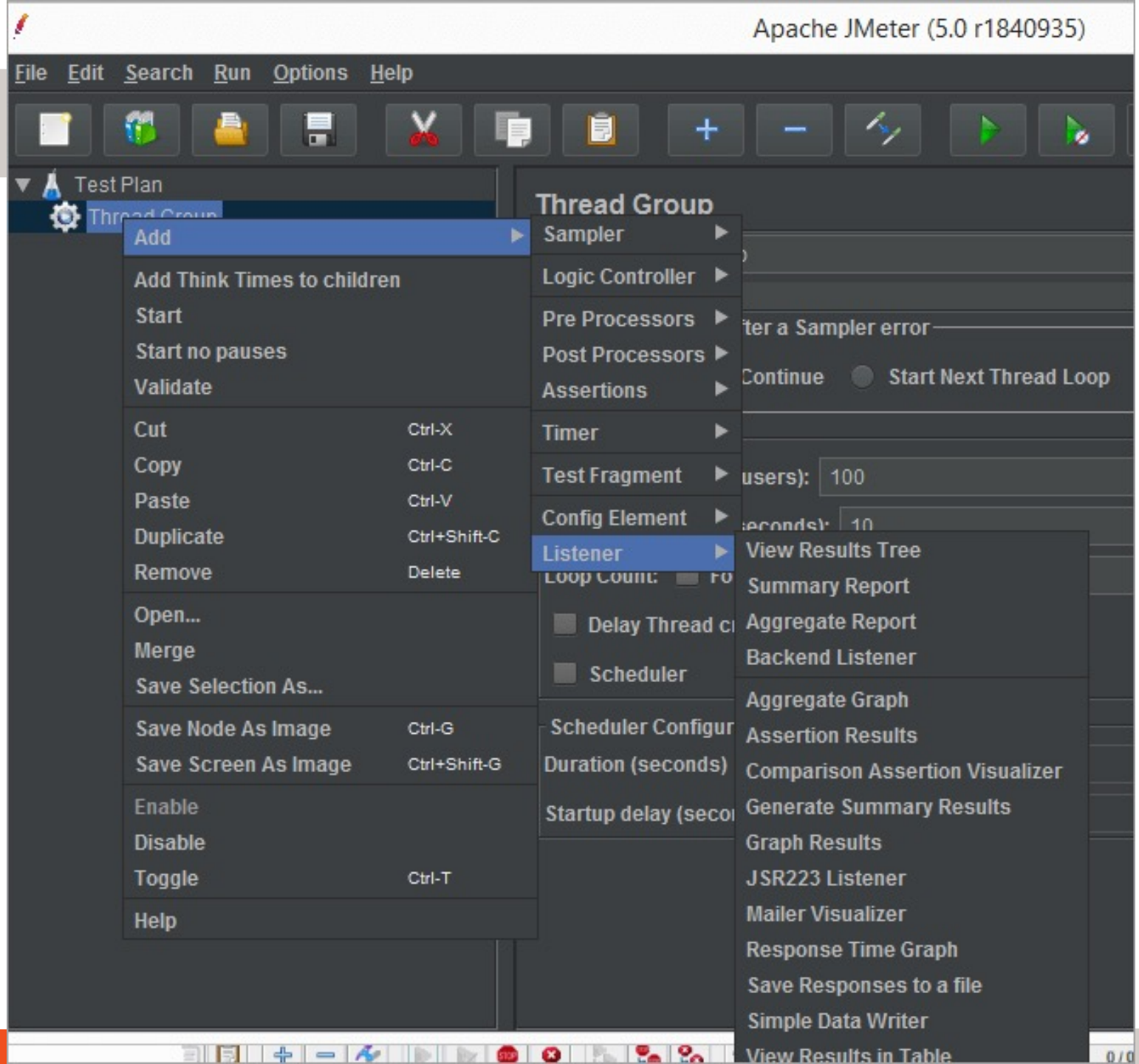
- Elements are added through Samplers
- Sampler tell JMeter about which type of request is sent to which server and with what parameters.
- HTTP Request, FTP Request, JDBC Requests are a few of the commonly used elements


File Edit Search Run Options Help
Test Plan
Thread Group
Add
Add Think Times to children
Start
Start no pauses
Validate
Cut Ctrl-X
Copy Ctrl-C
Paste Ctrl-V
Duplicate Ctrl+Shift-C
Remove Delete
Open...
Merge
Save Selection As...
Save Node As Image Ctrl-G
Save Screen As Image Ctrl+Shift-G
Enable
Disable
Toggle Ctrl-T
Help
Sampler
Logic Controller
Pre Processors
Post Processors
Assertions
Timer
Test Fragment
Config Element
Listener
Flow Control Action
HTTP Request
Debug Sampler
JSR223 Sampler
AJP/1.3 Sampler
Access Log Sampler
BeanShell Sampler
FTP Request
JDBC Request
JMS Point-to-Point
JMS Publisher
JMS Subscriber
JUnit Request
Java Request
LDAP Extended Request
LDAP Request
Mail Reader Sampler
OS Process Sampler
SMTP Sampler
TCP Sampler

How JMeter Works?

#3) Add Graph Result:

- The Listener lists various reports that provide the tester with a variety of graphical analyses of performance reports.
- The tester can set different reports to measure the performance.


Apache JMeter (5.0 r1840935)
File Edit Search Run Options Help
Test Plan
Thread Group
Add
Add Think Times to children
Start
Start no pauses
Validate
Cut Ctrl-X
Copy Ctrl-C
Paste Ctrl-V
Duplicate Ctrl+Shift-C
Remove Delete
Open...
Merge
Save Selection As...
Save Node As Image Ctrl-G
Save Screen As Image Ctrl+Shift-G
Enable
Disable
Toggle Ctrl-T
Help
Sampler
Logic Controller
Pre Processors
Post Processors
Assertions
Timer
Test Fragment
Config Element
Listener
Delay Thread c
Scheduler
Scheduler Configur
Duration (seconds)
Startup delay (seco
ter a Sampler error
Continue
Start Next Thread Loop
users): 100
View Results Tree
Summary Report
Aggregate Report
Backend Listener
Aggregate Graph
Assertion Results
Comparison Assertion Visualizer
Generate Summary Results
Graph Results
JSR223 Listener
Mailer Visualizer
Response Time Graph
Save Responses to a file
Simple Data Writer
View Results in Table

How JMeter Works?

#4) Execute Test:

- Once the required settings are done, click on the "Start" icon to begin the test.

- Once the test is complete, the tester can view the results using Listener in the above step.

can be increased in two ways:

- Run the tests in nonGUI mode (using console – find more)
- Distribute the load generation to multiple JMeter servers.