SELENIUM

A Brief Overview

Arun Motoori

This eBook is provided by

QAFox

www.QAFox.com

Contact Author

⇒ Name: **Arun Motoori**
⇒ Blog: https://www.qafox.com/
⇒ Email: arun.motoori@gmail.com
⇒ LinkedIn and Facebook

Connect to me on LinkedIn & Facebook >>>

QAFox

www.QAFox.com

Easiest way to find my blog is to search "QAFox" in Google.com

Happy Learning :)

Table of Contents

Introduction

Overview

The whole and sole purpose of this eBook is to give you a high level and big picture on Selenium Test Automation Tool. I believe this eBook will be very useful for the beginners, who wants to know the in and out of Selenium at a high level, so that they can begin their journey on Selenium well. Without any doubt, this book will also be very useful for the ones, who believe they have already learnt Selenium and also the people who are in the process of learning Selenium, as they can cross check their yet to date learning and see whether they are upto the mark in Selenium.

The main motive of this eBook is to make everyone aware of Selenium and plan their learnings in a better and organized way.

- Arun Motoori

https://www.qafox.com/

What is Selenium?

In simple words, Selenium is a Test Automation Tool developed by the company named “ThoughtWorks” from the year 2004.

Things that Selenium can automate

Selenium can automate Web Applications which generally run in Web Browsers. For example, Selenium can automate web applications like www.gmail.com, as it runs over web browser.

Things that Selenium cannot automate

Selenium cannot automate Desktop / Windows based applications which don’t run on Web Browsers . For example, Selenium cannot automate desktop / windows based applications like MS Word, MS Excel etc. Luckily most of the applications in the market are Web Applications.

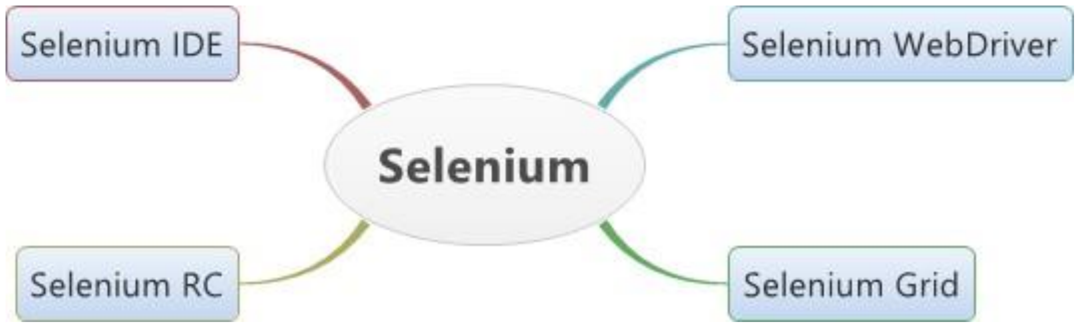
Selenium is an Open Source and Free tool

Unlike other proprietary tools which require us to purchase license for using them, Selenium is an open source and free tool. We don’t have to spend a single penny for using Selenium tool. Read more at the below link:

- Selenium is a free and open source functional test automation tool

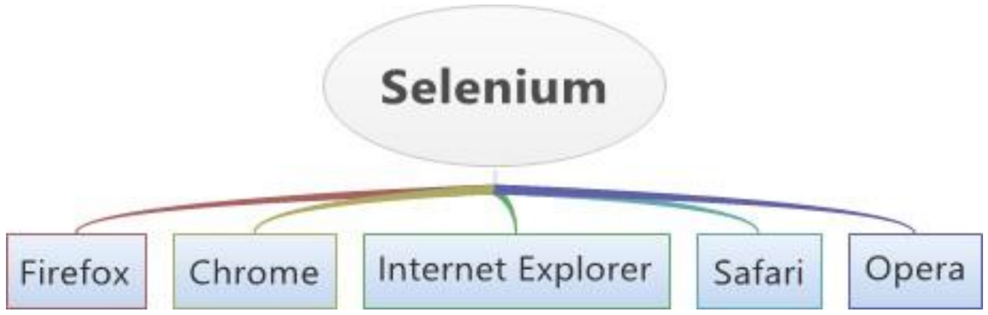
Different components in Selenium

Selenium is not a single tool, instead it is a set of tools. The below are the four different components which we together call as Selenium:



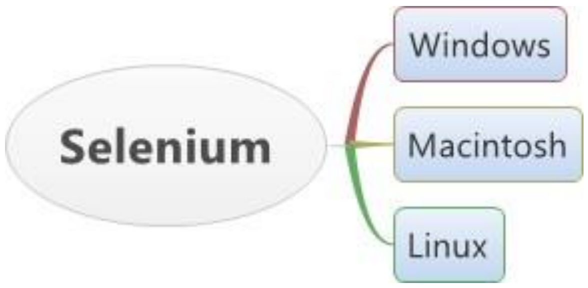
Browsers supported by Selenium

Most of the projects automate their applications for Firefox, Chrome and Internet Explorer browser types. The below are the 5 browser types that Selenium supports in total:



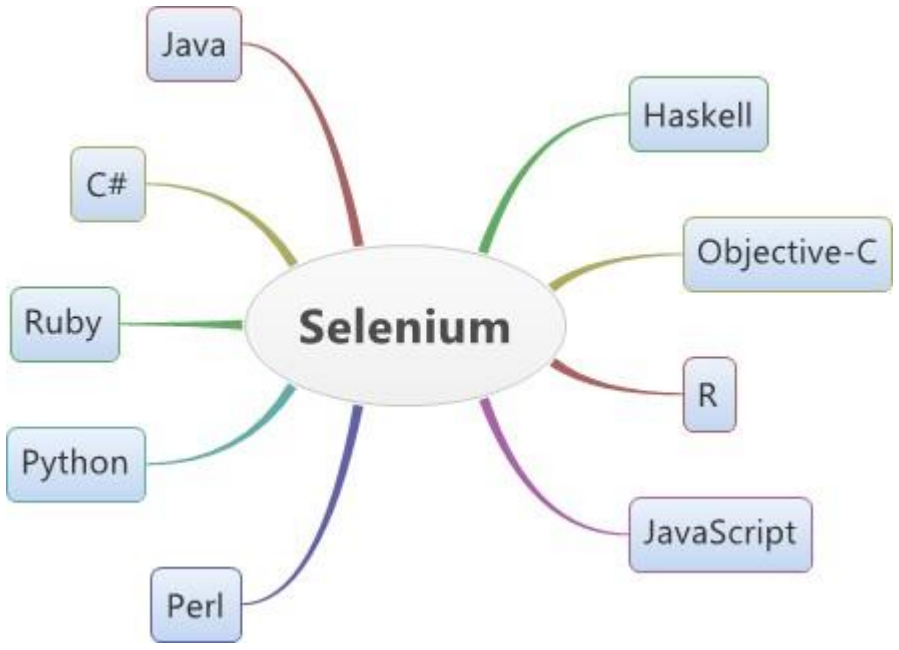
Operating Systems supported by Selenium

Selenium supports all the famous operating system in the market. The below are the 3 operating systems that Selenium supports:



Programming languages supported by Selenium

Selenium supports multiple programming languages for creating automation scripts. Hence we can use any of the below programming languages supported by Selenium:



Out of all the languages supported, Selenium has more support for Java language and most of the Projects use Java, C#, Ruby and Python for automation the scripts using Selenium. And also, we can automate the scripts in Selenium using Java language, even though the application under test is developed using C# languages.

Different versions of Selenium

So far, Selenium was released into the market in 3 different versions. The below are the three different version of Selenium:

- Selenium 1
- Selenium 2
- Selenium 3

The latest version of Selenium is Selenium 3 and it got released into the market on Oct 13, 2016.

History of Selenium

Selenium Core

Selenium was created by Jason Huggins in 2004. To avoid repeated execution of test cases day by day, he created a javascript program and named it as JavaScriptRunner which was later renamed as Selenium core. Over a period of time, Applications were only allowing the internal JavaScript programs by treating the external JavaScript programs as a security break. This became a drawback for Selenium core as it uses JavaScript programs to interact with the Application under test. To overcome this problem while testing using Selenium core, application testers used to install the Selenium Core's JavaScript programs into the Application code's local copy.

Selenium RC

To overcome the drawback of Selenium core, Paul Hammant has created a proxy server named 'Selenium RC' to trick the Application under test that the JavaScript programs are in the same local machine where the Application is residing, even though it is not. The drawback of Selenium RC approach is, we have to use a proxy server named 'Selenium RC' to communicate between the Application code and Automation code.

Selenium Grid

Selenium Grid was developed by Patrick Lightbody, to reduce the time of Automation scripts execution by running the scripts in parallel on different machines. i.e. Instead of executing all the scripts on a single machine, to reduce the time of execution, all the scripts will be divided across different machines and executed simultaneously.

Selenium IDE

Selenium IDE was developed by Shinya Kasatani, to record the tests like recording a video and execute the recorded tests like playing a video. Hence Selenium IDE is a record and playback tool. Selenium IDE is released into the market as a Firefox add-on/plugin/extension and can be installed on the top of the default Firefox browser. Once installed, we can simply record the tests on firefox browser and playback when required.

Selenium 1

Selenium 1 is nothing but the combination of Selenium IDE, Selenium RC and Selenium Grid.

Selenium 1 = Selenium IDE + Selenium RC + Selenium Grid

Selenium WebDriver

Selenium WebDriver was created by Simon Stewart in 2006. As the browsers and web applications were becoming more powerful and not allowing external JavaScript programs. While Selenium RC was tricking the Browser that Selenium core's Java Script program is part of the Application code using the Proxy server as a communicator, but Selenium WebDriver used the concept of native drivers to interact with the Application under test and hence eliminated the need for a proxy server. Selenium WebDriver got the support from the largest vendors of the Browsers like Firefox, Chrome, Internet Explorer, Opera and Safari to make Selenium a native part of their browsers. Hence the browsers wont treat Selenium's JavaScript programs as an external. The native drivers available for supported browsers are FirefoxDriver, ChromeDriver, InternetExplorerDriver, SafariDriver and OperaDriver. The development of this drivers is done by Selenium guys, even though the drivers are native to Browsers.

Selenium 2

Later Selenium 1 (i.e. Selenium IDE + Selenium RC + Selenium Grid) merged with Selenium WebDriver to make Selenium more powerful and got released into the market as Selenium 2 on July 8th, 2011. Earlier to this merger, there used to be separate teams for developing Selenium RC and Selenium WebDriver. As these two teams identified that Selenium can become more powerful on merging Selenium RC and Selenium WebDriver, and finally released the merged version of Selenium as Selenium 2 into the market.

Selenium 2 = Selenium IDE + (Selenium RC + Selenium WebDriver) + Selenium Grid

i.e. Selenium 2 is the combination of Selenium 1 and Selenium WebDriver.

Selenium 2 = Selenium 1 + WebDriver

Selenium 3

Selenium 3 is the latest version on Selenium and is released into the market on Oct 13, 2016.

Change#1 - The big change in this version is that the Selenium RC which is part of Selenium 2 is dropped out in Selenium 3. Selenium RC which was internally implementing Selenium Core’s JavaScript program libraries is now replaced with the backed WebDriver API implementation. i.e. What ever the tasks that can be performed only by Selenium RC’s JavaScript implementation, can now be performed using the backed WebDriver API implementation which is more flexible. There is no change in the WebDriver implementation from Selenium 2 to Selenium 3. Only the Selenium RC JavaScript libraries got replaced with backed WebDriver API implementation from Selenium 2 to Selenium 3. There is less impact due to this change, as most of the users are writing tests in WebDriver, but there is significant impact on the users who are writing tests in Selenium RC, as Selenium RC wont be supported by Selenium team in future releases and hence should migrate to WebDriver for future support.

Selenium 3 = Selenium 2 - Selenium RC + Backed WebDriver API for RC

Change#2 - Another big change in Selenium 3 is that, we have to use Mozilla’s gecko driver for running Selenium Automation scripts on Firefox browser. Till Selenium 2, development and support of selenium driver for Firefox is provided by Selenium guys using FirefoxDriver class in Selenium, but from Selenium 3 the development and support for selenium driver for Firefox will be provided by Mozilla Firefox Browser vendor using the gecko driver. Hence from Selenium 3, we have to use gecko driver in order to execute the Selenium Automation scripts on Firefox browser.

Change#3 - Apple company will be providing the support for Safari Driver for executing the Selenium Automation scripts on Safari browser of macOS Sierra operating system.

Change#4 - Microsoft company will be providing the support for Edge Driver for executing the Selenium Automation scripts on Edge browser.

Hence major changes in Selenium 3 from Selenium 2 is the dropping out of Selenium RC JavaScript libraries implementation by replacing them with Selenium WebDriver’s API implementation. And the browser vendors have taken the ownership to support the drivers which are required to run the automation scripts on the browser instead of selenium guys supporting their development. i.e. gecko driver is supported by Mozilla, safari driver is supported by Apple and edge driver is supported by Microsoft.

This eBook is provided by

QAFox
www.QAFox.com

Contact Author

⇒ Name: **Arun Motoori**
⇒ Blog: https://www.qafox.com/
⇒ Email: arun.motoori@gmail.com
⇒ LinkedIn and Facebook

Connect to me on LinkedIn & Facebook >>>

QAFox
www.QAFox.com

Easiest way to find my blog is to search "QAFox" in Google.com

Happy Learning :)

Prerequisites

Prerequisites required for learning Selenium

The below are the different prerequisites that are required to know before learning Selenium:

⇒ Software Testing Basics
⇒ Automation Basics
⇒ Core Java
⇒ Web Technologies like HTML, CSS, XML, XPATH, DOM and JavaScript

Software Testing Basics required for learning Selenium

The below are the few Software Testing Basics that are required for learning Selenium:

⇒ What is Software Testing ?
⇒ Why do we perform Testing ?
⇒ What is a Defect ?
⇒ Why do we find Defects?
⇒ What are Test Cases ?
⇒ What is Manual Testing ?
⇒ What is Regression Testing ?
⇒ What is Automation Testing ?

If you are new to testing, go through the below blog post to get started.

⇒ What is Software Testing ?

Automation Basics required for learning Selenium

The below are the few Automation Basics that are required for learning Selenium:

⇒ What is Automation Testing ?
⇒ How can an Automation tool perform testing without a tester ?
⇒ What is the purpose for automating the testing ?
⇒ Advantages of Automation Testing
⇒ Application suitable for Automation Testing
⇒ Which Tests can be automated ?
⇒ What is the goal of Automation Testing ?
⇒ What are the most popular Automation Tools ?

If you want to learn all of the above Automation basics, go through the below post:

⇒ Test Automation Basics

Java concepts required for learning Selenium

Java programming language plays a vital role in Selenium Automation. Its not required to learn the complete Java, instead learning of Core Java is enough for Selenium. The below are the different Core Java concepts that are required in Selenium at a high level are:

⇒ OOPS concepts
⇒ Primitive Data Types
⇒ Variables and Values
⇒ Operators
⇒ Decision making statements
⇒ Iterative Statements
⇒ Methods
⇒ Constructors
⇒ Arrays
⇒ String
⇒ Access modifiers
⇒ Wrapper Classes
⇒ Exception Handling
⇒ Files
⇒ Collections Framework.
⇒ And others miscellaneous Core Java concepts.

Having the knowledge on Core Java is enough for Selenium. Its not required to learn the Advanced Java concepts.

- Arun Motoori
https://www.qafox.com/

Web Technologies required for learning Selenium

Apart from learning Core Java, its required to learn different Web Technologies for Selenium The below are the different Web Technologies which are required for learning Selenium:

⇒ HTML
⇒ CSS
⇒ Xpath
⇒ DOM
⇒ JavaScript
⇒ XML

If you want to learn the HTML concepts, go through the below post:

⇒ HTML Basics

This eBook is provided by

QAFox

www.QAFox.com

Contact Author

⇒ Name: **Arun Motoori**
⇒ Blog: https://www.qafox.com/
⇒ Email: arun.motoori@gmail.com
⇒ LinkedIn and Facebook

Connect to me on LinkedIn & Facebook >>>

QAFox

www.QAFox.com

Easiest way to find my blog is to search "QAFox" in Google.com

Happy Learning :)

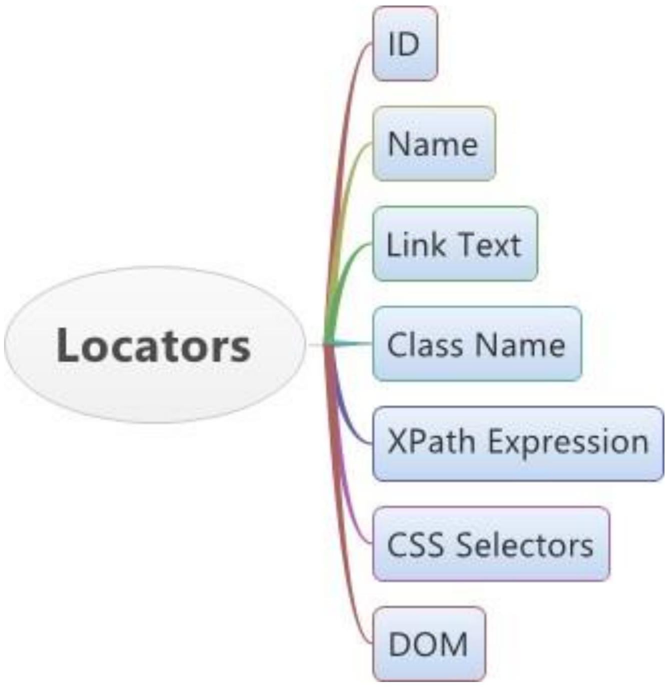
Locators and Add-ons

Purpose of Locators in Selenium

Locators are used in Selenium to identify the Graphical User Interface elements on the web pages. Lets say if you want to click on the ‘Login’ button on facebook page, we have to use the locator of ‘Login’ button in Selenium Automation code. Similarly to automate any Application using Selenium, we have to provide the locators of the Graphical User Interface elements of the Application to the Selenium code. Once the required locators for automating the application are provided to the Selenium code, Selenium uses the provided locators to identify the respective GUI elements on the application.

Different types of Locator Strategies

The below are the different types of locators that we can provide in Selenium code for identifying the web elements on the application:



Locators Priority

As all the different types of locators strategies can be used to locate the same GUI element on the Application, we have to choose a single locator strategy out of all the available locator strategies. As a first priority, we have to use ID strategy, followed by Name, followed by Class Name and followed by Link Text. But, if the web element is not getting identified using the ID/Name/ClassName/LinkText locator strategies, then we have to go for CSS Selectors locator strategy. Still, if the Web Element is not getting identified using the CSS Selector strategy, then we have to finally go for Xpath Selectors locator Strategy. And finally, if the Web Element is not getting identified using the Xpath Selector locator strategy, then we have to go for DOM locator Strategy.

Xpath Expressions versus CSS Selectors

Out of all the locator strategies Xpath Expressions locator strategy is the powerful one, but CSS Selectors locator strategy is the recommended one.

The below is the reason behind recommending CSS Selectors over Xpath Expressions:

⇒ CSS Selectors are speed in identifying GUI Elements and hence improves the performance of your Selenium Automation scripts

But practically speaking, the performance difference between CSS Selectors over Xpath Expressions in identifying GUI elements is negligible. Hence this is not the main reason for recommending CSS Selectors over the powerful Xpath Expressions.

The below is the reason behind why Xpath Expressions are powerful over CSS Selectors:

⇒ Xpath Expressions can traverse backward and forward in HTML code, for identifying the GUI elements, where as CSS Selectors can only traverse forward. i.e. Using Xpath Expressions, we can locate the child element using the parent element's unique ids, names etc. And also using Xpath Expressions, we can locate the parent element using the child element's unique ids, names etc. But CSS Selectors, cannot locate the parent element using the child element's unique ids, names etc. Hence Xpath Expressions are powerful in identifying GUI elements over CSS Selectors.

By looking the above explanation, Xpath Expressions needs to be recommend over CSS Selectors. But the truth is, in some cases Xpath Expressions are recommend over CSS Selectors and in some other cases, CSS Selectors are recommend over Xpath Expressions. Now, lets find out why:

Xpath Expressions are not suitable, if you want your Automation Scripts to run on Internet Explorer browser. By using Xpath Expressions for identifying GUI elements, Selenium may not be able to identify few GUI elements on the web pages which are rendered on Internet Explorer browser. Hence if your requirement is to run the Selenium Automation Scripts on Internet Explorer browser, CSS Selectors are recommend over Xpath Expressions. But, if the CSS Selectors are not able to identify the GUI elements, then the powerful Xpath Expressions needs to be used.

But if you requirement is to run your Automation Scripts on Firefox and Chrome browsers only, then Xpath Expressions are the best choice over CSS Selectors, as they are more powerful than CSS Selectors.

Hence, incase of Internet Explorer, CSS Selectors are recommended over Xpath Expressions. But if the browsers list for running automation scripts don't have Internet Explorer browser, then powerful Xpath Expressions can be used over CSS Selectors.

Add-ons required for generating Locators

Though we can manually create Xpath Expressions and CSS Selectors by understanding their syntax, Firefox browser is providing few add-ons which can be installed and used for any of the below:

⇒ Automatically generates the Xpath Expressions / CSS Selectors.
⇒ Helps us in manually creating the locators in a better way.
⇒ Automatically generates the locators and the required Selenium statement.

Firefox Add-ons: Firebug, Firepath and WebDriver Element Locator are the Add-ons which does this job.

This eBook is provided by

QAFox
www.QAFox.com

Contact Author

⇒ Name: **Arun Motoori**
⇒ Blog: https://www.qafox.com/
⇒ Email: arun.motoori@gmail.com
⇒ LinkedIn and Facebook

Connect to me on LinkedIn & Facebook >>>

QAFox
www.QAFox.com

Easiest way to find my blog is to search "QAFox" in Google.com

Happy Learning :)

Selenium Components

Selenium Tools Set

Selenium is not a single tool, instead it is a set of tools. The below are the four different tools which we together call as Selenium:

⇒ Selenium IDE
⇒ Selenium RC
⇒ Selenium WebDriver
⇒ Selenium Grid

We can also call these tools set as Selenium components. Its not mandatory to use all these components to automate applications, instead we select them for automation based on our Applications requirement.

Selenium IDE

Selenium IDE is a 'Record and Playback' tool and is available for us to use in the form of a Firefox Add-on. So, before understanding Selenium IDE, we have to first know about 'Record and Playback' tools.

In simple terms, Record and Playback tools are used to automate any application by recording the tests and playing back the recorded tests:

Recording the tests

⇒ Recording the tests is just like recording a video using cellphone
⇒ Recoding in 'Record & Playback' tool is performed to create automated tests.
⇒ Practically speaking 'Record' option of 'Record & Playback' tool will be turned on and then the activities performed on the Application which is displayed in a browser will be recorded until the 'Record' option is turned off.

Playing the recorded tests

⇒ Playback is just like playing the recorded video.
⇒ Playback in 'Record & Playback' tool is performed to execute the recorded automated tests.
⇒ Practically speaking 'Play' option of 'Record & Playback' tool will be selected to execute the recorded automated tests.

Advantages of using 'Record and Playback' tools:

⇒ Very Easy for Automating the Tests Can be used by the non programmers as programming knowledge is not required for using 'Record and Playback' tools.
⇒ Takes less time for Automating the Tests It wont take much time to automate the tests using 'Record and Playback' tool, as we simply record the application actions over browser and later execute the recorded actions using play options.
⇒ Suitable for less complex projects 'Record and Playback' tools are very useful for automating the projects whose complexity is less, as the creation of automation tests will be easy and the time taken to automate the tests will be less when compared to other ways.

Disadvantages of using ‘Record and Playback’ tools:

⇒ Automation scripts maintenance is more When any buttons or links or UI changes in the application, we need to re-record all the earlier recorded automation tests which are getting impacted due the changes. i.e. We need to use ‘Record and Playback’ tool to re-record all the impacted automation tests. But when we use programming for creating automation scripts, this maintenance will be reduced, as we will update the changes in a single file in the automation framework to which all the tests will refer, instead of editing the changes in all the automation tests which got impacted by these changes. Hence, even though the creation of automation scripts using ‘Record and Playback’ tool is easy, the maintenance of the recorded scrips will be high

⇒ Reusability will be less If we want to automate 100 tests, where all the tests needs to be performed after logging into the application, then using ‘Record and Playback’ tool, we need to record login functionality for all the 100 tests separately. But if we use programing for automating tests instead of ‘Record and Playback’ tool, then as part of framework we can have a single file where we can keep the reusable code for login functionality and use the same reusable with the 100 tests. Hence the reusability will be less in case of automating the tests using ‘Record and Playback’ tool.

⇒ Not suitable for complex Applications Due to the maintenance and reusability problems in ‘Record and Playback’ tools, automating the test cases for complex Application is not recommended. As complex applications contains huge list of tests and it will be difficult to re-record the huge list of tests when anything changes in the application. And also due the huge list of scenarios, the scope for reusability will increase. Hence ‘Record and Playback’ tools are not suitable for complex applications.

Selenium IDE

Selenium IDE as mentioned earlier is a ‘Record and Playback’ tool and is one of the 4 components of Selenium. Selenium IDE is provided by Firefox in the form of an Add-on / Plugin and is free to install on your Firefox browser.

As part of learning Selenium IDE, the basic understanding of using Selenium IDE is enough and is not required to learn in detail.

- Arun Motoori

https://www.qafox.com/

As you have now understood the usage of ‘Record and Playback’ tool, the following are the different posts using which you can explore the functioning of ‘Selenium IDE’:

⇒ Installing Selenium IDE
⇒ What is Selenium IDE ?
⇒ Record and Playback using Selenium IDE
⇒ Validating Elements using Selenium IDE
⇒ Validating Elements using Assert Mechanism
⇒ Validating Elements using Verify Mechanism
⇒ View the source code of the steps recorded using Selenium IDE
⇒ Finding the locators using Selenium IDE

Hence Selenium IDE is a ‘Recording and Playback’ tool and is useful for creating automation tests for simpler applications in an easiest and fastest way. And no programming language is required for using this tool.

Selenium RC

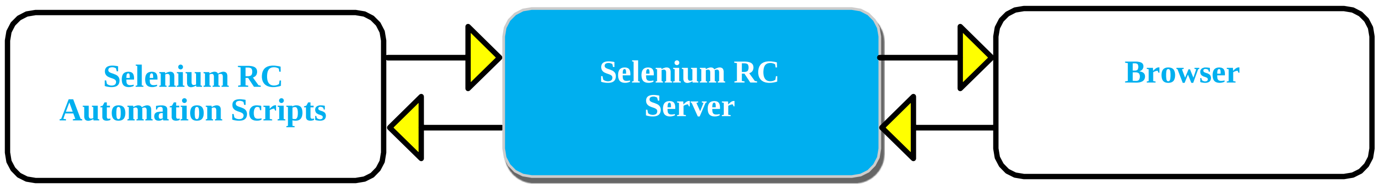
Selenium RC used to be the leader in the market for a long period of time until Selenium WebDriver is introduced into the market in 2006. Understanding Selenium RC will help us in understanding the power of the latest Selenium WebDriver. Selenium RC stands for Selenium Remote Control and is nothing but an API. Before understanding Selenium RC, we have to first understand the term API.

API stands for Application Programming Interface. API in simple terms is nothing but a group of functions and wont have any Graphical User Interface. Selenium RC is also an API and has its own set of functions which help us in automating the applications. For example, Selenium RC’s API has a function called click( ), which will help us clicking on the GUI elements on the Web Applications say Buttons, Links etc. Selenium guys did all the hard work and have already written the code for the functioning of methods like click( ) and provided them in the form of an API. We just need to use the click( ) method from the API on the GUI elements like Buttons and Links, to perform operations on them as part of Selenium Automation.

But Selenium RC is an outdated now, almost all projects now moved to a new API known as WebDriver. Since WebDriver API is the latest API of Selenium, we can simply ignore using Selenium RC. Even though Selenium RC is outdated, the below details of Selenium RC are explained for General Knowledge sake.

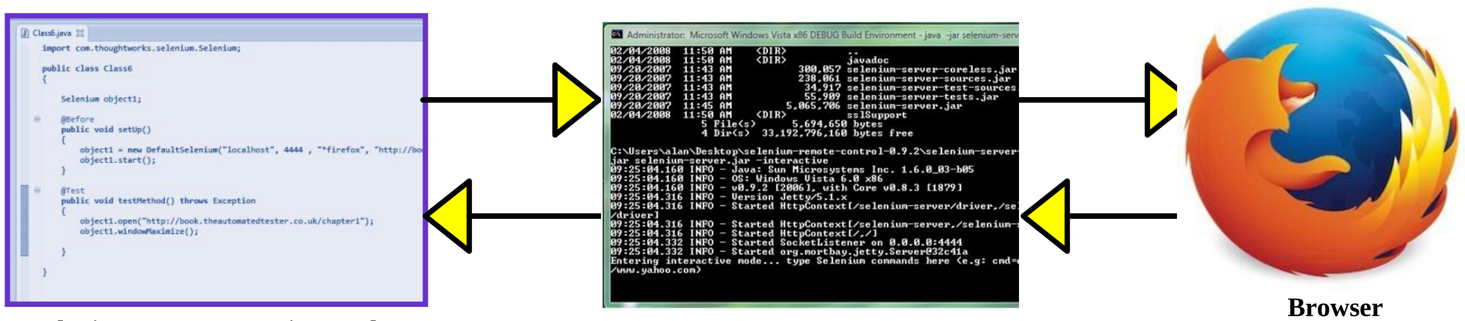
<u>Server is required for executing the automation scripts</u>

Selenium Remote Control Server is required for executing the automation scripts as shown below:



Selenium RC Server acts as the middle man between ‘Selenium RC Automation Scripts’ and the Browser as explained below:

⇒ Need to launch ‘Selenium RC Server’, as Selenium RC Server acts as the middle man between ‘Selenium RC Automation Scripts’ and the Browser. On executing the ‘Selenium RC Automation Scripts’, ’Selenium RC Server’ first injects a JavaScript program called Selenium Core into the browser. Once the Selenium Core is injected into the Browser by ‘Selenium RC Server’, Selenium Core injected into the browser will start receiving instructions by ‘Selenium RC Server’ from your test program. On receiving the instructions, Selenium Core will execute the instructions on the residing browser using the JavaScript commands. The browser will accept the instructions from Selenium core and the response of the executed instructions will be received by Selenium RC Server. Selenium RC Server will receive the responses of the browser and then displays the results. Selenium RC Server will then fetch the next instructions from the Automation scripts and repeats the same cycle. The below diagram depicts the same.


Class6.java
Administrator: Microsoft Windows Vista x86 DEBUG Build Environment - java -jar selenium-serv

```
import com.thoughtworks.selenium.Selenium;

public class Class6
{

    Selenium object1;

    @Before
    public void setUp()
    {
        object1 = new DefaultSelenium("localhost", 4444 , "*firefox", "http://bo
        object1.start();
    }

    @Test
    public void testMethod() throws Exception
    {
        object1.open("http://book.theautomatedtester.co.uk/chapter1");
        object1.windowMaximize();

    }

}
```

```
02/04/2008  11:50 AM    <DIR>          ..
02/04/2008  11:50 AM    <DIR>          javadoc
09/20/2007  11:43 AM           300,057 selenium-server-coreless.jar
09/20/2007  11:43 AM           238,061 selenium-server-sources.jar
09/20/2007  11:43 AM            34,917 selenium-server-test-sources
09/20/2007  11:43 AM            55,909 selenium-server-tests.jar
09/20/2007  11:45 AM         5,065,706 selenium-server.jar
02/04/2008  11:50 AM    <DIR>          sslSupport
               5 File(s)      5,694,650 bytes
               4 Dir(s)  33,192,796,160 bytes free

C:\Users\alan\Desktop\selenium-remote-control-0.9.2\selenium-server
jar selenium-server.jar -interactive
09:25:04.160 INFO - Java: Sun Microsystems Inc. 1.6.0_03-b05
09:25:04.160 INFO - OS: Windows Vista 6.0 x86
09:25:04.160 INFO - v0.9.2 [2006], with Core v0.8.3 [1879]
09:25:04.316 INFO - Version Jetty/5.1.x
09:25:04.316 INFO - Started HttpContext[/selenium-server/driver,/se
/driver]
09:25:04.316 INFO - Started HttpContext[/selenium-server,/selenium-
09:25:04.316 INFO - Started HttpContext[/,/]
09:25:04.332 INFO - Started SocketListener on 0.0.0.0:4444
09:25:04.332 INFO - Started org.mortbay.jetty.Server@32c41a
Entering interactive mode... type Selenium commands here (e.g: cmd=
/www.yahoo.com)
```

Selenium RC Automation Code **Selenium RC Server** **Browser**

Selenium WebDriver

Selenium WebDriver is introduced in 2006 and is the latest leader in the market. Selenium WebDriver is nothing but an API. Before understanding Selenium WebDriver, we have to first understand the term API.

API stands for Application Programming Interface. API in simple terms is nothing but a group of functions and wont have any Graphical User Interface. Selenium WebDriver is also an API and has its own set of functions which help us in automating the applications. For example, Selenium WebDriver's API has a function called click( ), which will help us clicking on the GUI elements on the Web Applications say Buttons, Links etc. Selenium guys did all the hard work and have already written the code for the functioning of methods like click( ) and provided them in the form of an API. We just need to use the click( ) method from the API on the GUI elements like Buttons and Links, to perform operations on them as part of Selenium Automation.

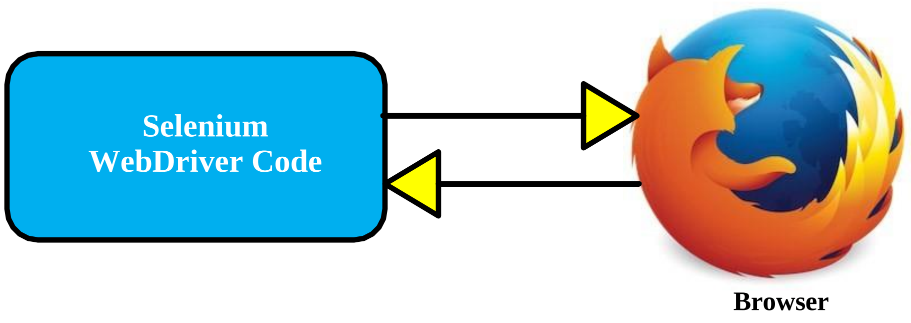
The older API of Selenium i.e. Selenium RC is an outdated now, almost all projects now moved to a new API known as WebDriver. Since WebDriver API is the latest API of Selenium, we can simply ignore Selenium RC.

<u>Advantages of Selenium WebDriver over Selenium RC</u>

- ⇒ Server is not required for executing the automation scripts
- ⇒ Supports Android and iPhone testing
- ⇒ Interacts natively with the applications running over supported browsers
- ⇒ Has complex and large API
- ⇒ API's are fully object oriented
- ⇒ Supports moving mouse cursors
- ⇒ Simpler Architecture
- ⇒ Faster, as it directly speaks to the browsers
- ⇒ Realistic interaction with GUI elements (For example, wont enter text into the disabled textbox)
- ⇒ Simpler, as it wont contain duplicate and confusing commands
- ⇒ Supports headless browsers like HTMLUnit, PhantomJS etc.

<u>Selenium WebDriver interacts natively with the Browsers and hence wont require proxy server as a mediator:</u>

Selenium WebDriver interacts with the browsers directly without requiring proxy server as shown below:



<u>Conclusion</u>

Selenium WebDriver is the latest API of Selenium and has successfully replaced the outdated Selenium RC API. The main advantage of Selenium WebDriver is that is natively interacts with the browsers and hence wont require any proxy server as a mediator for executing the automation scripts.

Selenium Grid

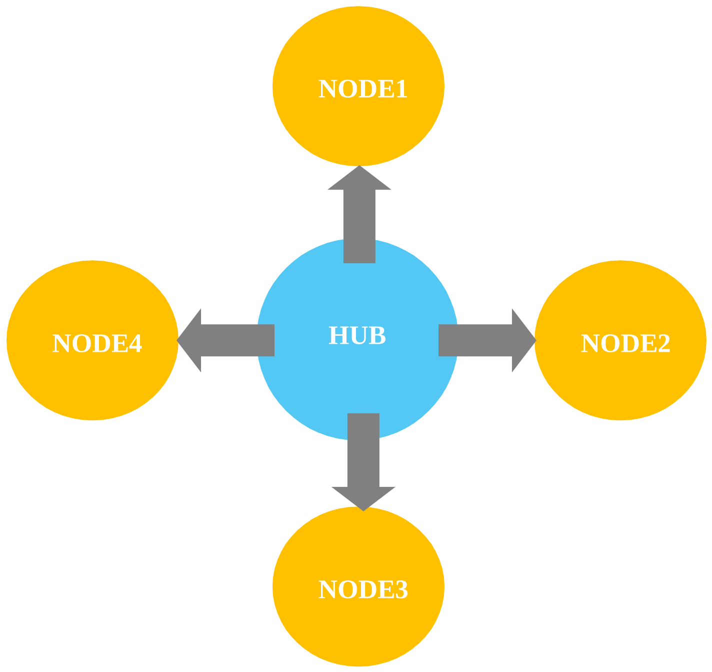
Selenium Grid is neither a tool having GUI, nor an API like Selenium RC / WebDriver. Instead Selenium Grid is just a concept, which will allow us to distribute the multiple tests across multiple machines, multiple Operating Systems and multiple browsers for execution.

Selenium Grid speed up the execution process when there are huge number of tests. i.e. Huge list of tests will be distributed across multiple machines and executed at the same time for reducing the execution time. For example, if one test is executing on machine1, at the same time another test will get executed on machine2.

Example: Lets say, we have 10 tests where each test takes 1 minute each. If you are not using Selenium Grid, then one test will be executed after another in the same machine and hence takes 10 minutes of time for executing 10 tests. But if we use Selenium Grid to distribute these tests across 5 machines, then 5 tests will be executed at the same time and hence it takes only 2 minutes for executing all the 10 tests.

Hub and Nodes

Using Hub and Nodes concept, Selenium Grid will distribute the tests across multiple machines, operating systems and browsers. Selenium Grid uses Hub as the central machine and uses it to distribute the tests for execution across multiple machines which are knows as Nodes. The below diagram depicts how Selenium Grid uses Hub and Nodes concept to distribute the tests across multiple machines, OS and browsers:



Versions of Selenium Grid

Selenium Grid has the two versions i.e. Selenium Grid 1 and Selenium Grid 2. Selenium Grid 1 can be used with the outdated Selenium RC and hence Selenium Grid 1 can be ignored. Selenium Grid 2 can be used with the latest Selenium WebDriver and hence it is the latest Selenium Grid version used in the market.

Conclusion

Selenium Grid is a concept using which we can define multiple machines as Hub & Nodes, and distribute the tests across multiple machines, operating systems and browsers. The purpose of Selenium Grid is to run multiple tests simultaneously across multiple machines to reduce execution time of tests.

This eBook is provided by

www.QAFox.com

Contact Author

⇒ Name: **Arun Motoori**
⇒ Blog: https://www.qafox.com/
⇒ Email: arun.motoori@gmail.com
⇒ LinkedIn and Facebook

Connect to me on LinkedIn & Facebook >>>

www.QAFox.com

Easiest way to find my blog is to search "QAFox" in Google.com

Happy Learning :)

Unit Testing Frameworks

Unit Testing

Unit Testing is a Software Testing Methodology in which individual tests are developed for each small unit of a Program. i.e. A Software Program is broken into the possible smallest units and individual tests are written to test all the smallest units. All the tests created for testing the smallest units of the program are known as Unit Test. The Methodology of Testing the Software Program using the created Unit Tests is known as Unit Testing. If you are still not able to visualize what is the term Unit in Unit Testing, lets understand it in a realistic way i.e. Unit is a smallest piece of software code. Still not convinced, lets me give an example of Unit. Example for Unit is a function or method in any programming language. Hope it is clear now :)

Unit Testing Frameworks

There are various Unit Testing Frameworks in the market. The below are few famous Unit Testing Frameworks which can be used with the famous Programming languages.

⇒ JUnit
⇒ TestNG
⇒ NUnit and many more.

The whole and sole purpose of these Unit Testing Frameworks is to make the process of developing and executing the Unit Tests in an easier way. Hence by using Unit Testing Frameworks, we can easily develop and execute the Unit Tests.

Purpose of using Unit Testing Frameworks in Selenium

Before understanding how Unit Testing Frameworks are used in Selenium, we have to understand that Unit testing is generally done by developers. i.e. Developer break the complete application code into smaller units and test all the units separately. As explained in the above sections of this chapter, Unit Testing is a process of breaking the code into smaller units and testing the broken units is known as Unit Testing. Unit Testing can be performed in both manual and by automation.

In order to automate this unit testing process, developer use different Unit Testing Frameworks based on the programming languages they have used. i.e. While the developer continue developing the application code by simultaneously creating unit test for the developed code, the existing unit tests will be automatically run by the Unit Testing Framework on the earlier developed code, to check the newly developed code is not breaking the earlier developed code. As this is done automatically by the Unit Testing frameworks, developers don't have to waste their time in manually testing the units tests on earlier written code, each time they develop new code. Though there are many other advantages of automating the unit testing process using the Unit Testing Framework, the one that I explained in this para is the main purpose of automating the Unit Testing process.

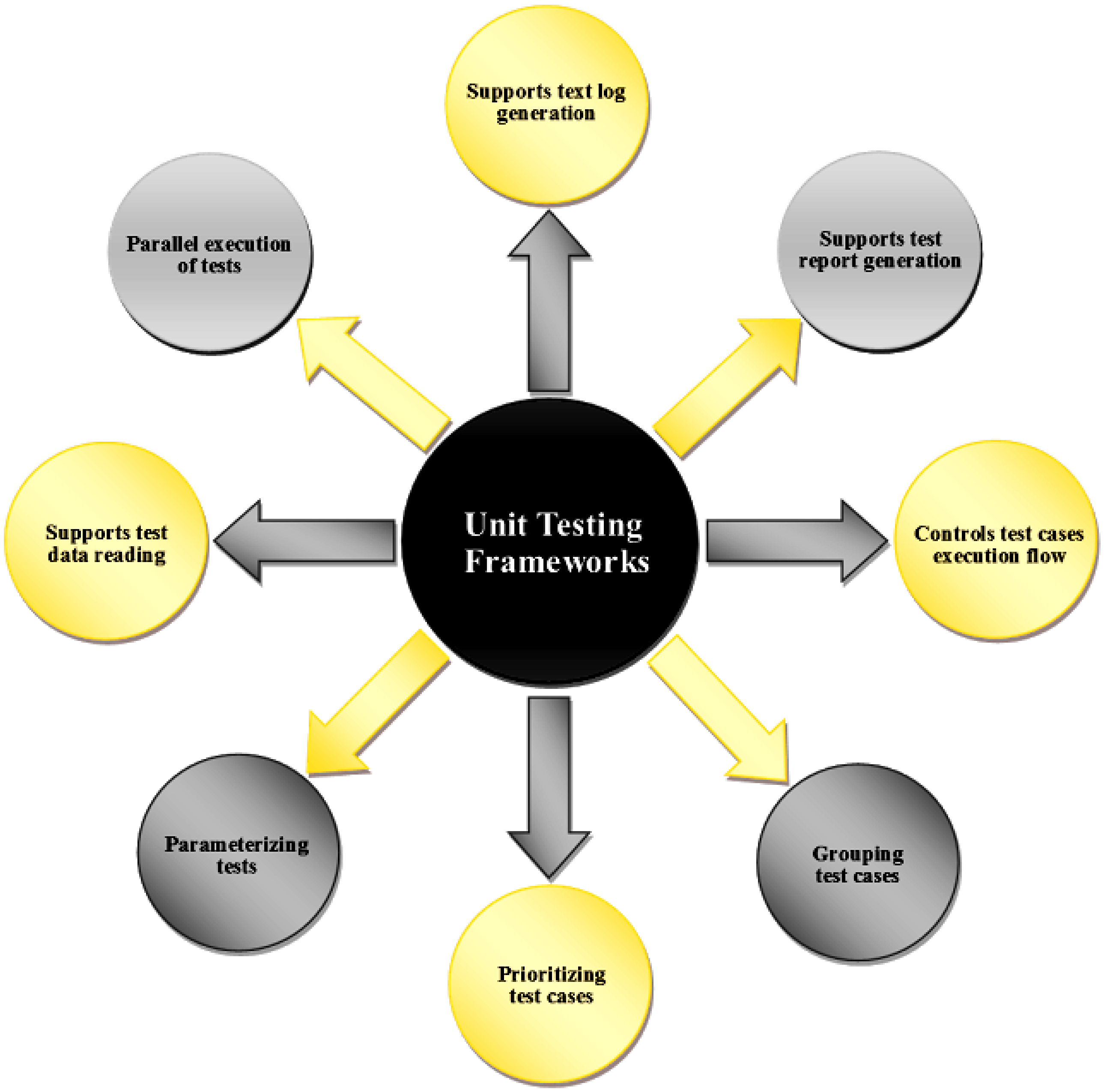
<u>Now, lets understand the purpose of Unit Testing Frameworks in Selenium:</u>

Unit Testing Frameworks play a major role in developing different types of Test Automation Frameworks in Selenium.

Unit Testing Frameworks can be used in Selenium to automate/support/perform any of the below:

⇒ Controls the flow of test case execution
⇒ Grouping the test cases in to separate groups is possible
⇒ Prioritizing the test cases is possible to prioritize which test needs to be executed first and which next.
⇒ Parameterizing the tests in such a way that the same tests can run multiple times using the different sets of data
⇒ Supports reading the data from the external sources like Excel files etc.
⇒ Parallel execution of tests to save time of test execution by executing multiple tests at the same time
⇒ Supports generating text logs to later find-out what are the different things that happened while the tests were executing
⇒ Supports generating reports to find out the test results after tests execution

The below diagram, depicts the purpose of Unit Testing Frameworks in Selenium:


Supports text log generation
Parallel execution of tests
Supports test report generation
Supports test data reading
Unit Testing Frameworks
Controls test cases execution flow
Parameterizing tests
Grouping test cases
Prioritizing test cases

Unit Testing Frameworks supported by Selenium

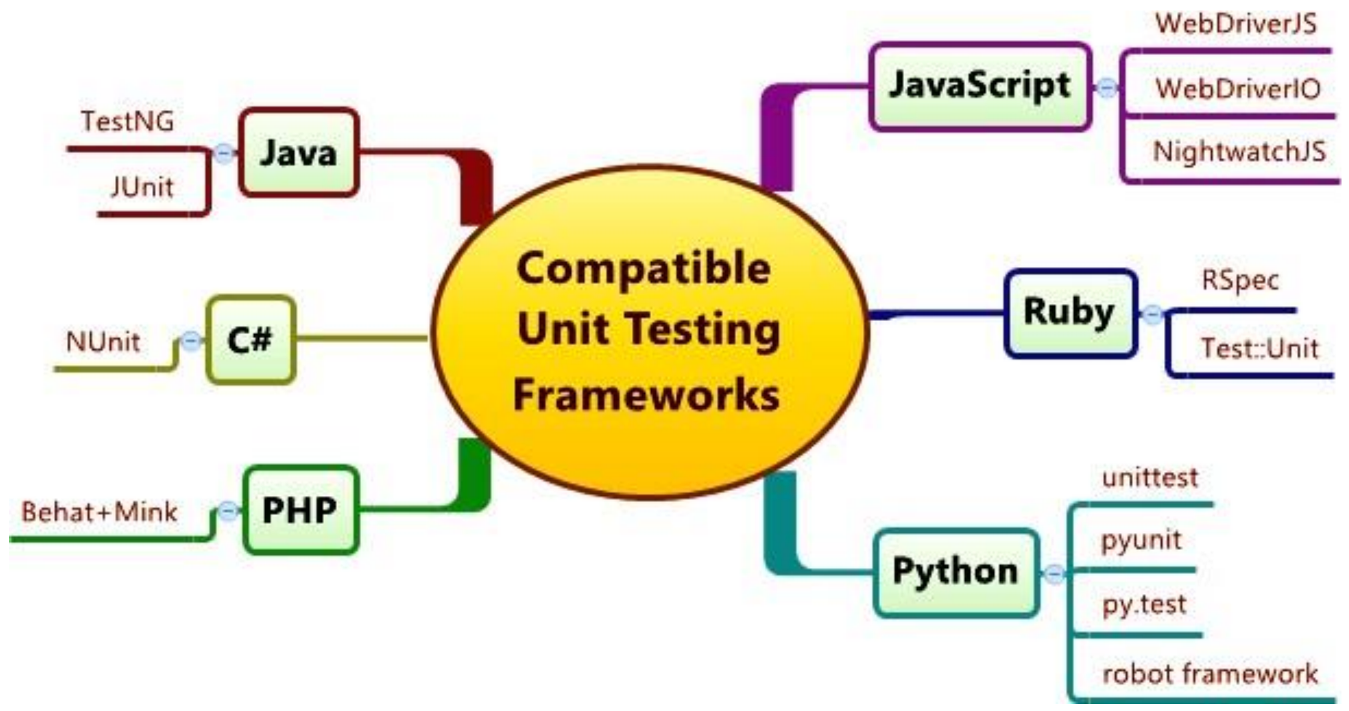
As we have understood the purpose of using Unit Testing Frameworks in Selenium Automation, now lets find out the different unit testing frameworks that are supported by Selenium based on different programming languages.

Programming languages and their compatible Unit Testing Frameworks:

The below are the different Unit Testing frameworks that are supported by Selenium and categorized according to their compatible Programming languages.

⇒ **Java :** TestNG and J Unit Unit Testing Frameworks
⇒ **C# :** NUnit Unit Testing Frameworks
⇒ **PHP:** (Behat+Mink) Unit Testing Frameworks
⇒ **Python :** unittest , pyunit , py.test and robot framework Unit Testing Frameworks
⇒ **Ruby :** RSpec and Test::Unit Unit Testing Frameworks
⇒ **JavaScript :** WebDr iver JS, WebDr iver IO and NightwatchJ S

The below diagram depicts the above categorization of Unit Testing frameworks according to different programming languages:



Hence these are different Unit Testing Frameworks supported by Selenium based on the programming languages. And these Unit Testing Frameworks play an important role in building Test Automation Frameworks in Selenium.

TestNG Unit Testing Framework

If we are using Java programming language in Selenium Automation tests, then we have to choose either JUnit or TestNG as our Unit Testing Frameworks. But in Selenium Project, TestNG is preferred over JUnit, as TestNG Unit Testing Framework is more powerful and is very much suitable for Selenium.

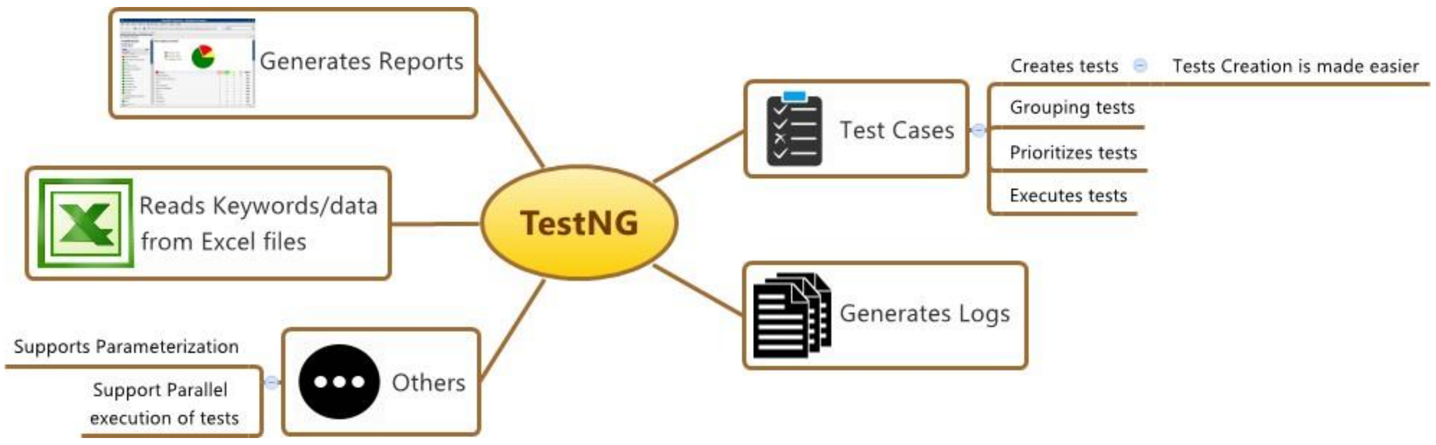
TestNG is a unit testing framework, which plays a major role in developing Test Automation frameworks using Java Programming language.

Advantages of TestNG Unit Testing Framework

The below are the different advantages of using TestNG Unit Testing Framework:

⇒ Test Case creation is made easier
⇒ Groups the Test Cases
⇒ Prioritizes the Test Cases
⇒ Executes the Test Cases
⇒ Generates Logs
⇒ Generates Reports
⇒ Reads Keywords / data from the Excel files
⇒ Supports Parameterization
⇒ Supports Parallel execution of Tests

The below diagram depicts the advantages of TestNG Unit Testing Frameworks:


Generates Reports
Reads Keywords/data from Excel files
TestNG
Supports Parameterization
Support Parallel execution of tests
Others
Test Cases
Creates tests
Tests Creation is made easier
Grouping tests
Prioritizes tests
Executes tests
Generates Logs

Installing TestNG

In order to use TestNG Unit Testing Framework in Selenium Java Project, we have to perform the below tasks:

⇒ Install TestNG Extension in Eclipse IDE (Click here for detailed explanation)
⇒ Configure TestNG Jar files in the Selenium Java Project (Click here for detailed explanation)

TestNG Annotations

TestNG provide a huge set of Annotations. The main reason behind TestNG's popularity is nothing but the TestNG annotations, which can be used for configuring the test cases and test cases in Selenium Automation. There annotations needs to be used with Java methods. i.e. a line before the beginning of the Java methods. When a particular TestNG Annotation is used before the Java Method, Java method will behave differently based on the specified Annotation. TestNG annotations begin with @ symbol. On using TestNG annotations with Java method, the syntax of the Java method will like below

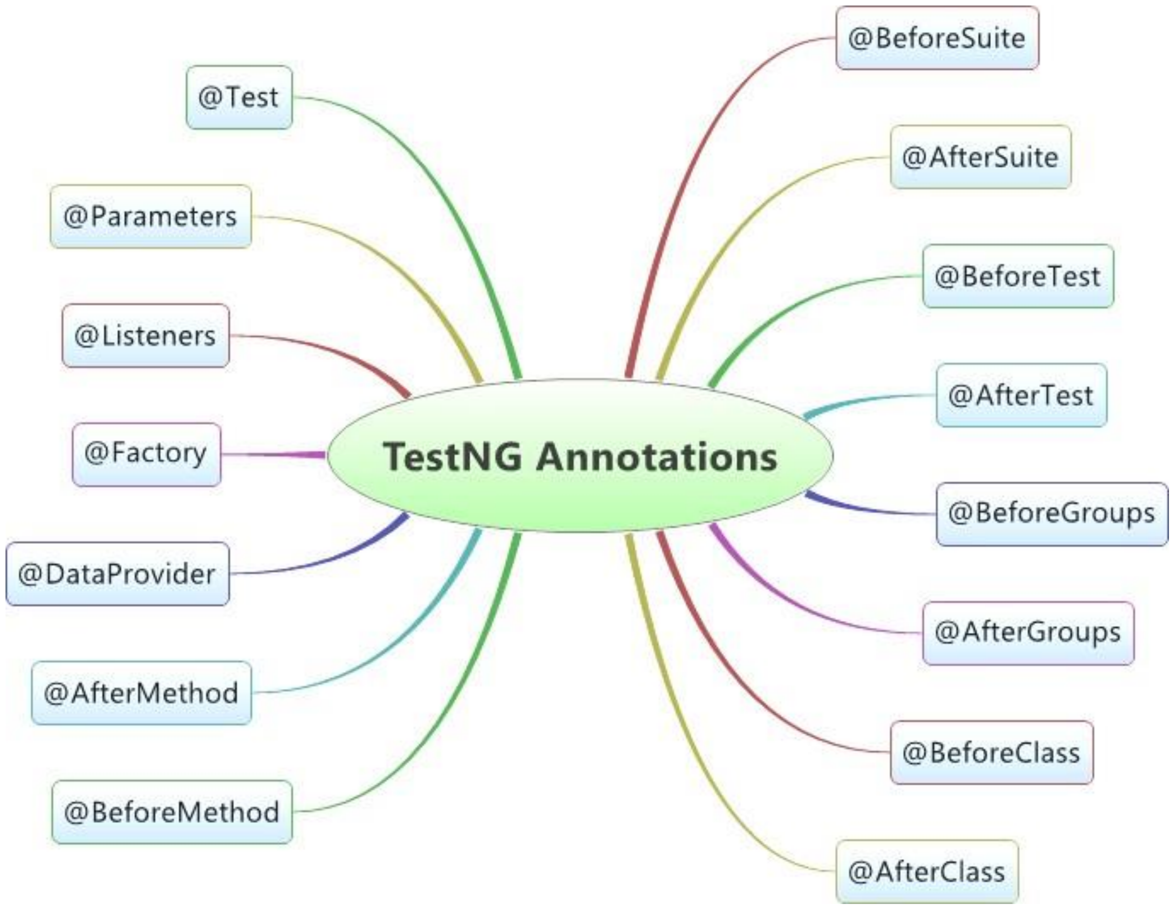
Syntax:

```
@AnnotationType
public void methodName(){

}
```

Different TestNG annotations

If you are planning to use TestNG for executing the Selenium Automation code, then the usage of main( ) method in Java is no more required. Using TestNG annotations, we can specify which methods needs to be executed first and which methods can be executed later and so on. The below are the different TestNG annotations that we use in Selenium Automation:



This eBook is provided by

QAFox

www.QAFox.com

Contact Author

⇒ Name: **Arun Motoori**
⇒ Blog: https://www.qafox.com/
⇒ Email: arun.motoori@gmail.com
⇒ LinkedIn and Facebook

Connect to me on LinkedIn & Facebook >>>

QAFox

www.QAFox.com

Easiest way to find my blog is to search "QAFox" in Google.com

Happy Learning :)