Selenium – set of tools

Selenium IDE, Selenium RC, Selenium WebDriver, **SeleniumGrid(Selenium server)**

Automation – UIAutomation:

Why automation:

Save time and resources

Run regression test cases easily

Cross browser testing

DataDriven Testing

Selenium IDE- record and playback tool

It is a plugin in firefox browser

To get IDE:

<https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/>

Command- action we do on the browser

eg: type, click, scroll etc…

Target- on which element we are doing action

html element can be identified by its attributes.

Value- input entered by user

assert and verify:

assert: if actual and expected do not match- test case is failed

verify- it verifies the actual and expected are equal or not- but even if it do not match

it will log the issue but it will not fail the test case.

Disadvantages of IDE:

We can run test cases only in Mozilla- cross browser testing is not possible.

We cannot do data driven testing

Not always the scripts given by tool are as efficient as written by developer/AutomatinEngineer.

We cannot go for continous build(maven) and continous integration(jenkins) in case of tools.

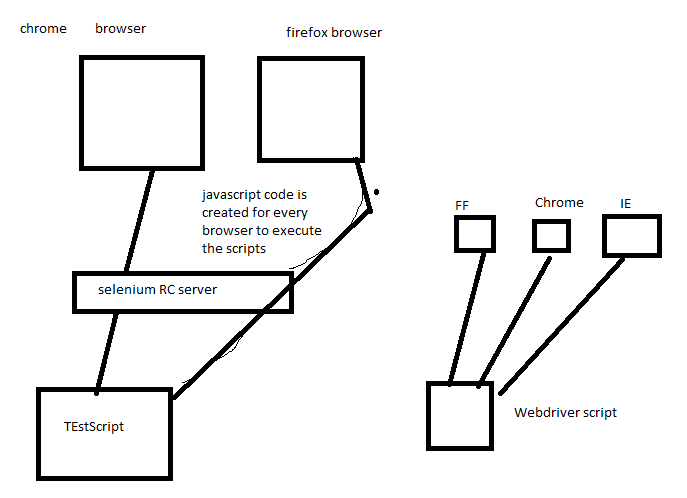
SeleniumRC:

SeleniumRC is based on programming – its not a tool like IDE.

Disadvantage in RC:

It uses server in between to execute the scripts on browser – which makes it slow

It executes script by using javascript code internally which again slows down the process.



**SeleniumWebDriver**:

The scripts are executed by **vendor (firefox, safari, chrome, opera etc) specific implementations** and selenium do not generate any javascript code for execution purpose.

WebDriver is main interface with implementation classes from each vendor like

**FireFoxDriver**

**ChromeDriver**

**SafariDriver**

There is no server in between script and browser.

**How to automate:**

we need language, test framework and automation tool.

Java+**TestNG+SeleniumWebDriver**

unit testing fw- JUnit, TestNG- it can be used by developers to write unit test cases,

it can be used by testers to run automation scripts.

How Does WebDriver ‘Drive’ the Browser Compared to Selenium-RC?

Selenium-WebDriver makes direct calls to the browser using each browser’s native support for automation. How these direct calls are made, and the features they support depends on the browser you are using. Information on each ‘browser driver’ is provided later in this chapter.

For those familiar with Selenium-RC, this is quite different from what you are used to. Selenium-RC worked the same way for each supported browser. It ‘injected’ javascript functions into the browser when the browser was loaded and then used its javascript to drive the AUT within the browser. WebDriver does not use this technique. Again, it drives the browser directly using the browser’s built in support for automation.

For setup:

### <http://www.seleniumhq.org/download/> - Selenium Client & WebDriver Language Bindings

Java- 3.2.0 – latest version

earlier versions- 3.0, 2.53

Create new java project in eclipse—create one empty lib folder--copy jars to lib folder--Add jars to build path

Also we need to download driver.exe(for windows) files which are specic to browser.

<http://www.seleniumhq.org/download/>

### under Third Party Drivers, Bindings, and Plugins

How to write an automation test script:

1. Open the browser and load the url

2. find the element on which we need to perform action

3. perform action- which gives actual result

4. Assert for actual vs expected

Mozilla:

FireBug

FirePath – helps to write our own xpath , css selectors and evaluate

them to make sure its correct before adding to program.

By class – has 8 different static methods to locate a html element from the webpage

By.id(“”)

By.name(“”)

By.className(“”)

By.linkText(“LogOut”)

By.partialLinkText(“Log”)

By.tagName(“”)

By.xpath(“”)

By.cssSelector(“”)

Which is most preferred locator?

id , name are preferred if they are unique and they are not dynamically changing.

If not then we use css selector and xpath.

Which is preferred? css selector or xpath?

css selector is preferred bcz its faster and simple but there are some places

where we need to use xpath like identifying element through its inner text-

for this we use text() in xpath,

Also using css selector we cannot traverse reverse in Dom means when

you have a child node you cannot get parent – but we can do this in xpath using preceeding

xpath –

absolute xpath – It traverse from the root of document to reach the element we are looking for

html/body/div[1]/div[1]/div[5]/div[1]/div/div/div[1]/div/div[2]/div/div/div/div[2]/div/form/div[2]/div[1]/input[1]

absolute xpath is lengthy and in case any element is added in hierarchy we need to change automation script for that locator.So relative xpath is preferred in automation.

relative xpath – identifies the element through its attributes without traversing from the root.

//elementName[@attrName=’valueOfAttribute’]

//input[@id=’uname’]

css selector:

basic syntax:

elementName[attributeName=’value’]

WebDriver methods:

get

getTitle

getCurrentUrl

findElement(By by) – this returns single WebElement object.

in case the element is not present this returns NoSuchElementException

findElements - this returns list of WebElement objects.

Also in case the element is not present this returns empty list means

list with size 0.

so findElements can be used to check element presence requirement-

where its returns 0 size list when it do not find element.

windowHandles:

getWindowHandle()- current window name

getWindowHandles()- list of windows

switchTo.window(“windowName”)

close – close will close only the currently active window

quit- closes all the windows opened by current driver instance.Useful to close

parent and child windows.

navigate()-return Navigation class object

back

forward

refresh

to

get and navigate difference:

get only loads the browser but navigate can load browser using to method and traverse backward/forward and also refresh page if required.

WebElement:

click

sendKeys

getAttribute

getText

clear- clear the text box