Automation: Testing the given application using tool or a program.

How to automate:

We automate the testing using program or a tool where we configure the tests to be executed

And add assertions.

Get the reports.

UIAutomation- web application automation – Selenium- (IDE(tool),RC,Grid,WebDriver), Casper.js,Protractor-Angular.JS apps, Nightwatch.js ,SilkTest,QTP

WebServicesAutomation-RestAPI’s Automation – HttpClient,RestAssured, Chakram, SoapUI tool

MobileAutomation- Appium,Perfecto,Calabash

Advantages of automation:

Saves time and resources

Regression testing –

DataDrivenTesting- testcases will be repeated with different sets of data- valid/invalid, boundary values

CrossBrowserTesting- Mozilla,chrome,IE,safari etc

CrossPlatform- mac, windows,linux….

ManualTesting-ErrorProne sometimes

When to Automate and When not to Automate?

Tight DeadLines – we cannot automate under tight dead lines

When UI changes /going to change in future then do not automate that particular module.

When application is not going to change at all and its already in maintenance- meaning no enhancements or no new user stories – then also no need to automate

Selenium: 2004- Jason Huggins from thoughtworks

IDE – It is a record and playback tool for UI automation, it is plugin in firefox

Command- action

Target- html element (textbox,link,plaintext,dropdown,checkbox,radio,)which you are using to automate

Value- text you are giving as input in text box

Selenium - IDE, RC, Grid, webdriver

Drawbacks of IDE tool:

We cannot do continous build and continous integration

Datadriven testing- same test case with differnent sets of data is not possible

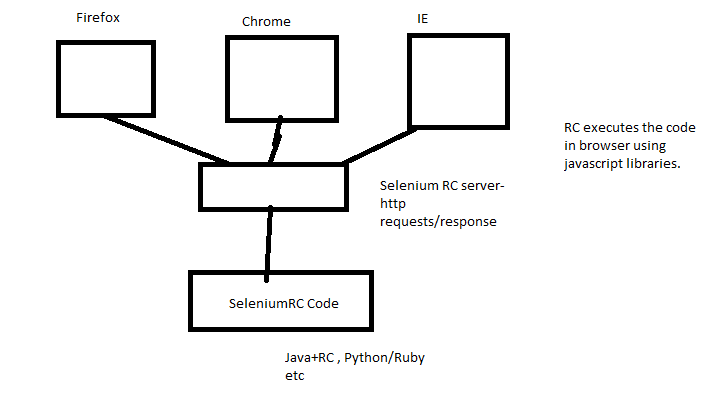
Testcases can be executed only in mozilla

Logical combinations which can be done in programming is not possible

Selenium 1 - RC

Javascript code

RC was developed as a server which executes the script on respective browser.

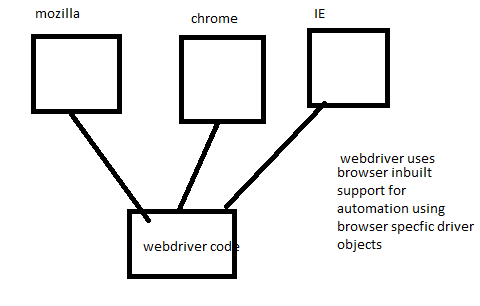


Selenium 2 - WebDriver - API(library based) that supports browser automation.

**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.

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.



Automating Steps:

OPen browser

Find the element

Perform action on element

Get the result- actual- assert it with expected

WebDriver:

WebDriver - interface with some common methods

Implementation Classes- Given by specific browsers

eg- FirefoxDriver, ChromeDriver,IEDriver,SafariDriver,OperaDriver

WebDriver- interface methods

get

findElement(By class object)

Every html elment has attributes:

div,a,input,select,header,strong- html elements

id, class, name, value etc....

Using these elements, attributes we create locators.

By - class which provides static methods to locate element

8 diff methods which take string input and return By object:

id - id is most efficient and fastest way as it internally uses JS doc.getElementByid method directly.

name

class

tag

linkText

partialLinkText

xpath- Try to locate the element by traversing across the html document

absoulte xpath - starts from the root and traverses for the given element.It starts with singleslash.

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

relative xpath - It starts with // and it directly points to specific element instead of pointing

the element from the root.

.//\*[@id='global-search-input']

Basic xpath - //elementNameOr\*[@attrName='valueOfAttribute']

Xpath using Multiple Attributes which does "and" condition- .//input[@name='email'][@id='email']

Id/ any other attribute is dynamically changing we use certain methods in xpath like contains,starts-with

.//\*[@id='c5H2m-location']

.//\*[@id='c0DpC-location']

.//input[contains(@id,'location')]

contains(@attrName,'value')

.//\*[starts-with(@class,'locationField')]

.//div[contains(text(),'My Account')]

.//\*[@id='u\_0\_3']/preceding::div[2]

.//\*[@id='u\_0\_3']/following::div[2]

css selectors:

tagname

#idValue

.classValue

Basic css - //elementName(optional)[attrName='valueOfAttribute']

multiple attributes- input[name='email'][class='inputtext']

input[id\*='location']- this gives all elements which have location in their id

div[class^='location'] - startswith

div[class$='location'] - endswith

cssselectors - they are faster than xpath and compatible with most of browsers unlike xpath

But in scenarios to deal with traversal with preceding nodes - xpath is used as we cannot

traverse back using css selectors.

We prefer to use relative xpath over absolute xpath bcz absolutexpath is lengthier and if

some element changes in between at later point of time then locator should be changed in code.

cssselector- It

WebElement-Interface

It has methods to perform actions on elements.

sendKeys- method that allows to input text in to the text box or text area