Selenium

**Automation**: Testing the given application(web or mobile) using the external tools or the automation framework code.

Selenium is used to automate the web applications.

**Advantages of Automation:**

Repeatability of the tests using tools or code rather than manual testing leads to

increase the speed of execution of test cases.

We can vary the tests for different user inputs easily by implementing data driven testing.

It easily supports frequent regression testing do be done in minimal tme and generates good test reports to analyse.

It also helps to find the defects which you may miss while doing manual testing.

**When to automate and when not to automation?**

Automation depends on timeframe available, if we have a very tight dead line then we cannot go for automation.

Automation should be avoided if there is any UI change expected in near future as we need to rewrite the automations scripts again.

**Also application scenarios wise – pages which need manual interactions cannot be automated like**

1. If there is some captcha code being generated instantly and u need to enter that code or payment page where OTP is bing sent to mobile.

2.Image / Video Previews - Between transitioning between thumbnail and playback of videos I have observed not well automated

3.BarCodeReader cannot be automated

**Different Automation tools & frameworks**: Selenium, QTP-vbscript,SilkTest etc

Selenium is set of different tools to support different test approaches…

In 2004- Jason Huggins started Selenium at Thoughtworks.

Initially he developed a javascript library

**Selenium IDE** – is a tool which geneartes automation scripts

It’s a plugin/add only in Mozilla browser.

Create automation scripts quickly , to learn automation scripts with the help of a tool aided mechanism.

Also as a begineer who move from manual testing to automation- he can have a glance of automation scripts using the selenium IDE.

He can also get an idea about locators.

**Selenium 1** – programmatic way of automation which had been now developed as webdriver..

Selenium RC – Selenium Remote Control - client server architecture.

Its currently deprecated.

**Limitations of Selenium RC:**

Security became an issue due to javascripr injection

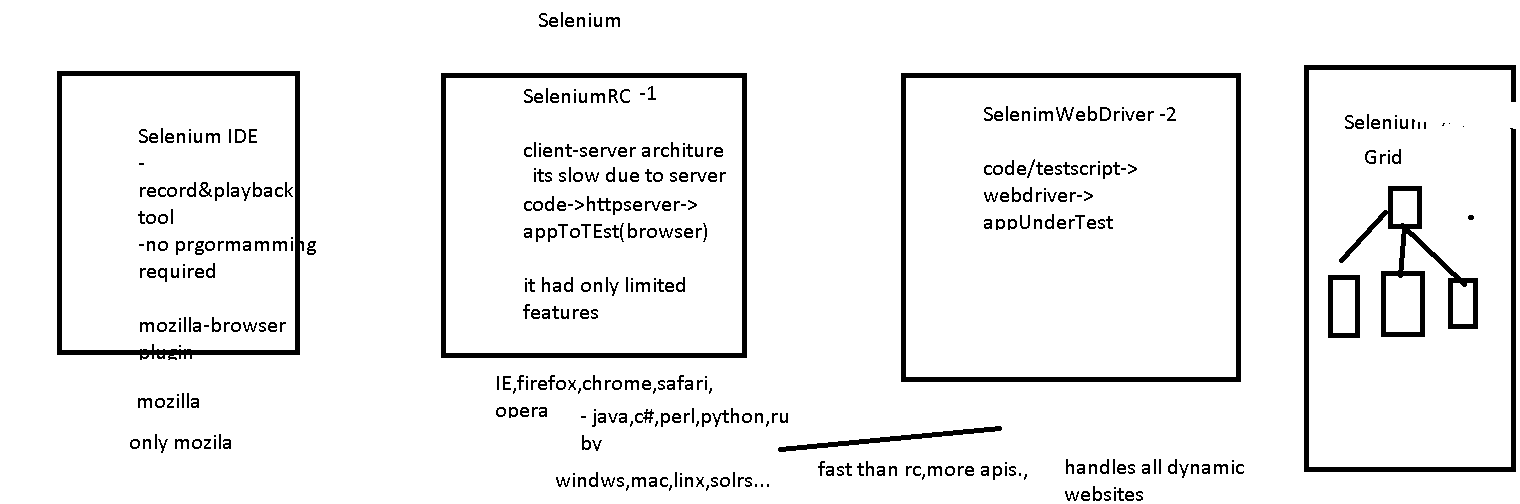
Its slower because of client –server architecture i.e., there was RC server in between TestAutomationCode and application to Test unlike webdriver

**Selenium 2 - Selenium Webdriver-** programmatic way of automation

It has number of API’s(set of libraries) to achieve automation programtically.

It helps to create very robust, browser-base regression testcases and suites.

It helps to manage things more efficiently than a tool for mutiple environments.

****

**SeleniumWebDriver:**

**What we do in automation:**

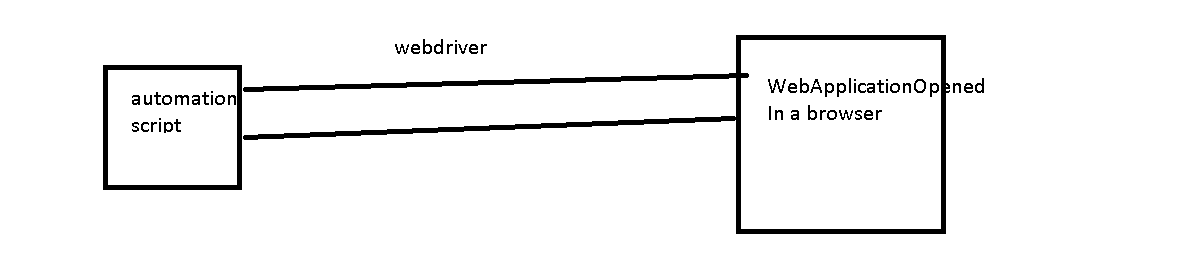
**Write scripts – selenium webdriver + java**

**Assert the testcases**

**Log the errors**

**Generate report to analyse the tests**

**Unittesting f/w – testng /junit**

****

**Automating steps:**

Open the browser and loading our website/any page we want to automate

Find the elements in a given page to automate and do the actions

Add assertions

Close the browser

**Webpage will be composed of html elements in window, html elements can be**

**Buttons,text boxes, links,dropdowns,statictext, titles, info, divs, image……**

**WebDriver-(I) deals mainly with browser/window related actions**

**Automation means we do some actions on this elemnts using our programs**

**WebElement(I) interface deals everything related to specfic html element you find—**

**How u find element – we use By class**

**WebDriver methods:**

Void **get**(String url) – pass what url we want to test as a string

WebElement **findElement**(By by)-we use this method to find a web element using By class(helps to pass the locator).It returns WebElement.

This will simply throw NoSuchElementException when it is not able to find the element.

**List<WebElement> findElements**(By by) – will return List<WebElement> eg: breadcumbs,nav links etc

to locate presence of element in a web page: if the element we are trying to get is not found then those findElements will return an emptylist- so we can also use this method to check for list size is not empty and perform logic accordingly in situations where we are not sure that element may be present or may not be present.

String **getTitle**() – title of the current page

Options **manage**() – 1. To maximize the browser window

Navigation **navigate**():

Naviagation: to – go to the paricular page url – its same like the get method

back()

forward()

refresh()

Void **close**()- close the curent window opened by the driver.

Void **quit**()- close all the windows opened by driver

String **getCurrentUrl**() – returns current url of the page which is loaded

String **getPageSource**()- gives the source code for the current page

**Takescreenshot** using TakeScreenShot Interface – used to record the error scenarios or log the error pages:

File screenshot = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(screenshot, new File("C:\\Users\\Training room\\MedhaJavaWS\\SeleniumWebDriver\\test.png"));

**WebElement methods:**

Void **Click**() – performs click action on given element- **links,buttons(both in and out of forms),checkboxes,radiobuttons**

Void **Clear**()- it will clear the text in the **input textbox or textarea**.

Void sendKeys(String text)- send some text to the textbox field

Void **submit**()- generally use for html form submissions on buttons i.e., **button inside a html form.**

WebElement **findElement**(By arg) – help to find element inside the element.

List< WebElement> **findElements**(By arg)- help to find elements inside the element

String **getText**() – returns inner text of the given html element

<a href=’testjk.com’..>CIiCKME</a>

String **getTagName**()- return the particular element tagName

<a href=’testjk.com’..>CIiCKME</a>

String **getAttribute** – returns html elemnt’s attribute value

String **getCSSValue** – returns css value for element

Point **getLocation** – gives where html element is located in webpage using x and y axis coordinates – using Point class a sreturn type

Dimension **getSize** – it will size of html element – width and height using Dimension class as return type

getScreenShotAs – to take the screenshot of the page ..

boolean isSelected – used for checkbox,radiobuttons,select dropdowns to see if option is selected or not

boolean isEnabled – to see if element is enabled or disabled

boolean isDisplayed – to see if element is present or hidden in page.

By class in webdriver has all static methods which we can simply call using the syntax By.methodName…these methods in By class help to locate a particular html element in our page.

By class methods:

1. id(String value) – this is most efficient and prferred way to locate any element.. it will take id value of html element as input and locate the html element.

Eg: <input id=”uname”…..>

<div id=”currentDiv”…>

2. name(String value)- this wil locate a html element with attribute name, this is also the most prfered way but we need to make sure that the same name is not used for more than one html element.

If we have more than matching name elements then the first macth will be returned .

<button id=”test1” name=”submit”….>

<input id=”test1” name=”username”….>

3. linkText(String value)- this wil locate the element using the link text for any of the given links.

<a href=”link.com”>Logout</a>

4. partialLinkText(String value) – this will locate the element with partial link name.

<a href=”link.com”>Logout</a>

5. className(String value) – this will locate the element using the class(css class) attribute of the html element

<button class=”submit-btn-class”…..>

6. tagName(String name) – help to locate the html element by its own tag name.

<input id=”test” name=”tt”…>

<button id=”test” name=”tt”…>

<font id=”test” name=”tt”…>

7. cssSelector(String value) – helps to locate elements based on css selector path

8. xpath(String value) – its most widely used , helps to locate the lelement based on its position in the Dom(html document object model).

Chrome browser ->inspect->copy xpath-//\*[@id="footerbot"]/div[1]/span/span/a/span[1]

Inspect using Mozilla-firebug - /html/body/div[1]/div/div/div/div/article/div[3]/div[1]/font – absolute xpath

Mozilla-Inspect element using firepath – relative xpath

.//\*[@id='result']/font

#footerbot > div.pull-left > span > span > a > span.region

Select dropdowns/Select API : First create webelement using driver.fineElement(By….)

Create Select class object by passing above created WebElement objcect as input.

On select object we get different methods to select the dropdown values using either selectByIndex or selectByVisbleText or selectByValue..

We can use getOptions method to see all the available dropdown options.

We can also use Select class to select multiple select dropdowns and here we can use methods like isMultiple,deselectAll, deselectByIndex or deselectByVisbleText or deselectByValue..

We can also use getAllSelectedOptions to see what all we selected.

Select select = new Select(element);

RadioButtons: You can simply findelement using By class and click on element.

Or we can findElements which return list of 2 or more elements() and we can select from that list whatever we want by using index.

Eg: radioList.get(1).click();

radioList.get(3).click();

Waits:

**Implicit wait:**

Whenever we are not sure which specific element we want to wait upon in a given html page and this wait will work for entire time the browser is open.

Initially it checks for element to be present and if its not prsent then it starts waiting for a given time and then finally at the end of wait time it again checks for element. If it could not find elemet – then it throws Exception..

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

It is generally prefered for overall small waiting time

**ExplicitWait:**

Gives option to specify which element we are trying to wait for, it accepts some condtions as input and it helps to handle waiting with specific elements/conditions.

Whenever we know that specific element takes more time to load – we prefer explicit time as it saves time not waiting for all the elements for same time.

WebDriverWait wait = new WebDriverWait(driver,20);

WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("close-button")));

wait.until(ExpectedConditions….anyConditionasperrequirement);

wait.until(function)

wait.until(predicate)

**FluentWait :** Same as explicit wait where in it wwaits for a given condition but then it takes frequency also as input.Also takes input to ignore a specific exception.

Whenever we know that a particular element is loading for 5 seconds sometimes and sometimes it takes 10seconds…wherein we are not sure about wait time…

Wait fluentwait = new FluentWait(driver)

.withTimeout(10, TimeUnit.*SECONDS*)

.pollingEvery(5, TimeUnit.*SECONDS*)//frequency

.ignoring(StaleElementReferenceException.class);

Function<applyMethodInputDataType, applyMethodReturnDataType>function = new Function<WebDriver,List<WebElement>>(){

public List<WebElement> apply(WebDriver driver){

return driver.findElements(By.*xpath*(".//\*[@class='suggestions-list']/li"));

}

};

totalWaitTime-20

Frequency - 5

5th, 10th,15th, 20th

Actions: helps to deal with mousehover interactions, drag and drop,keyboard handling…

build will compile list of actions to a single step and perform will do all actions

together

eg:actions.clickAndHold(menuElement).clickAndHold().click().build().perform();

but in single action scenaios ..even if u perform without build it will work—u need build only when u want multiple actions together

eg:

menuElement = driver.findElement(By.id("navigation-top-cat-label-1308"));

actions.moveToElement(menuElement).perform();

menuElement = driver.findElement(By.id("navigation-top-cat-label-1764"));

actions.moveToElement(menuElement).click().perform();

Deal with drapAndDrop:

actions.dragAndDrop(from,to).perform()

or

actions.clickAndHold(fromElement).moveToElement(toElement).release(toElement).build().perfom();

To handle differnet wiindows:

We use driver.getWindowHandle()- gives current window and driver.getWindowHandles()- all the list of windows open.

And we use driver.switchTo.window(String arg) to switch the control from one window to other window.

For all popup boxes like alert,prompt,confirm boxes we use Alert class or AlertAPI to accept/dismiss/sendkeys..

Alert alert = driver.switchTo.alert()

alert.dismiss/accept..

Switch to iframe – driver.switchto.frame

Inspectors –

Firepath- enhanced version of firebug.we can also write our own xpath and evaluate/verify them with eval button using firepath.

Firepath - Relative xpath - .//\*[@id='headerfblogin']

We can still absolute xpath also from firepath by changing the settings

Firebug- Absolute xpath - /html/body/header/div[2]/div/ul/li[1]/a/button

Chrome-xpath helper

To be discussed:

Parallel testing – testing..

Selenium grid

AutoIt- interacting with desktop tasks….

pagefactory

**HtmlUnitDriver** : Headless browser is a browser which do not have any GUI.

After 2.53 HtmlUnitDriver is not part of selenium jar ..so we need to download external jar and add it to project or add as maven dependency.

https://github.com/SeleniumHQ/htmlunit-driver/releases

When do we use it:

To do any quick sanity testing we prefer this browser as it is very fast in exceution when compared to other browser.

Also we can use it to simulate multiple browser environment in same machine.

This uses RhinoJavaScript engine.

By default javascript exceution is disabled in this browser…so we need to enable it using driver.setJavascriptEnabled(true)..

**DesiredCapabilities**: Information about browser, platfoem,device,etc..

These are define as JSON objects(key-value pairs) to define the features that a session will support.

Like DesiredCapabilties

We can also set any browser specific settings using

**chromeOptions** for chrome browser

**FirefoxProfile** for firefox browser

And then pass it to WebDriver objct.

1. It is a class in org.openqa.selenium.remote.DesiredCapabilities package.
2. It gives facility to set the properties of browser. Such as to set BrowserName, Platform, Version of Browser.
3. Mostly DesiredCapabilities class used when do we used Selenium Grid.
4. We have to execute mutiple TestCases on multiple Systems with different browser with Different version and Different Operating System.
5. Mobile automation – to specify platform, device, what file to test…

To mange **cookies**: We can use Cookie class in the Webdriver..we can add,delete and get existing cookies using this class methods.

**JavaScriptExecutor**: Its an interface in WebDriver which provides a way to exceute javascript code through selnium driver.

((JavascriptExecutor)driver).executeScript(…..)

When to use it: To enhance the capabilties of existing scripts – we can use this class.

AutomationFramework : To create reusable,modular and maintainable application

1. To do automation – we need to choose a language –Java,Javascript,C#
2. Don’t want to execute using main method – (Testing F/W – testng)
3. Test definitions- test cases – verify&Validate the testcases – assertions & logs -(Testing F/W – testng)
4. Test reports – to analyse -(Testing F/W – testng)
5. Also we need logs /screenshots to debug-(log4j)
6. We want a mechanism to run all the tests together –(Testing F/W – testing.xml)
7. Code reusability – Java DesignPatterns(dependencyInjection,singleton,PageObject,UIMapping,DataDriven…)
8. Avoid hardcoded data – properties file (java.uti.Properties)
9. Data driven using excel,csv,text… - apache poi
10. Build and continuously integrate the changes(Maven&Jenkins)
11. Automation F/w- web/mobile/RestAPis – SeleniumWebdriver,WebDriver+Appium,selendroid, http-client

Automation Framework:

Selenium + Testing F/W +DesignPatterns/Structure+ Utilities - external utitilities like apache poi for reading excel data,Log4j for logging etc …..

DataDriven- drive the test data from external file into the code

PageObject pattern – Every page in website has corresponding Object(java) to deal with all the functionalities of that particular page.

KeyWordDriven – this pattern follows like every keyword for certain functionalty

Modular

ActionsDriven

UIMapping – mapping the uilocators with certain key value pairs at one place in the project rather than hardcoding across the project.

So that we can avoid changing java code everytime when the locator is being changed. It also helps foe easy maintenance and readable.

Hybrid Framework – combination of one or more above mentioned f/w’s

Eg for hybrid : PageObject+data drivern

-----------------------------------------------------------------------------------------------

August-batch

Selenium

Automation – Testing or developing automation framework with the help of tools or programs.

Cross browser Testing – chrome, mozilla, chrome,safari,IE

OS’s – mac, windws, linux

Functional testing

Integration testing

System testing

Uat Testing

Prod

Dev sit uat prod

Dev qa stage prod

Dev qa(qa1, qa2) prod

RegressionTesting – retest a defect or when you are testing a new enhancement

Regression is part of all phases like functional, sit, uat…

Datadriven testing- test a functionality with different sets of data

Advantages of Automation:

Saves time and resources

You can perform data driven testing easily

Regression testing is easy

Repetative manual tasks are avoided

More efficient and can figure out defects which might have in manual testing

It increases productivity and less eroor prone

It reduces investment cost

Tools- QTP, UFT, Selenium IDE, …

Selenium

2004- Jason huggins – Thoughtworks – Developed SeleniumCore(javascript based)

Using which people started automation with browsers

Selenium core is base for Selnium Ide and Selenium RC

Around 2006, to overcome the drawbacks of IDE and RC stewart from google

Came up with WebDriver concept

Limitations of Selenium Core:

Javascript based and it was less secure

Not all functionlaities could be automated

And could not run efficiently in all browsers.

Selenium IDE:

Record and playback tool for automation

It is a plugin that works only in mozilla

It can automate applications to some extent but not complete automation.

Because we don’t have any flow control and data driven testing not possible.

Selenium IDE can be used for prototyping the test cases.

Continous build and Continous Integration( – maven and jenkins) is not possible.

Selenium RC:

Using selnium rc jars we write the code – to execute the script you need to start and stop selenium server that interacts between your code and AUT(application under test)

Selnium 1 – RC and IDE

Selenium2- WebDriver

To automate an application with webdriver:

1. WebDriver API jars
2. Java/ruby/python
3. Testing framework- TestNG

Selenium grid – multiple platforms and browsers testing

SeleniumServer

Browsers: chrome,moziulla,IE,safari,htmlunit,phantom js, android,ios

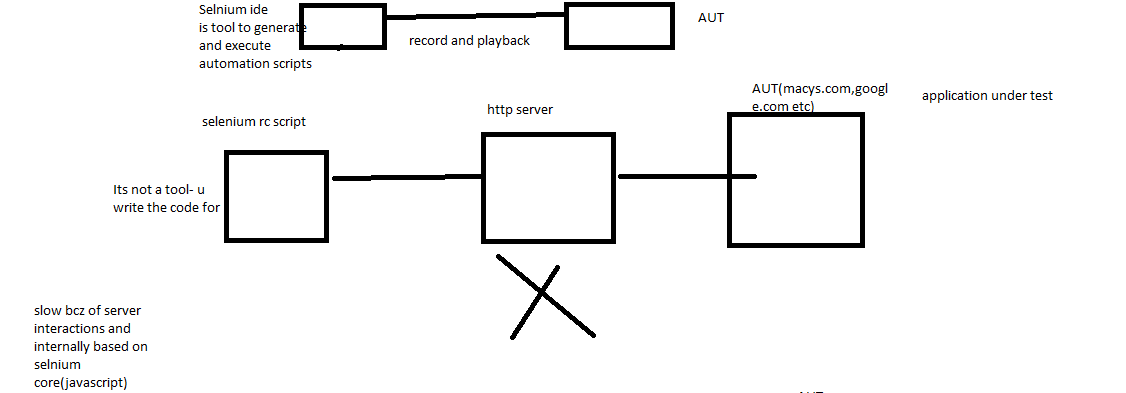
OS- mac,windows,linux

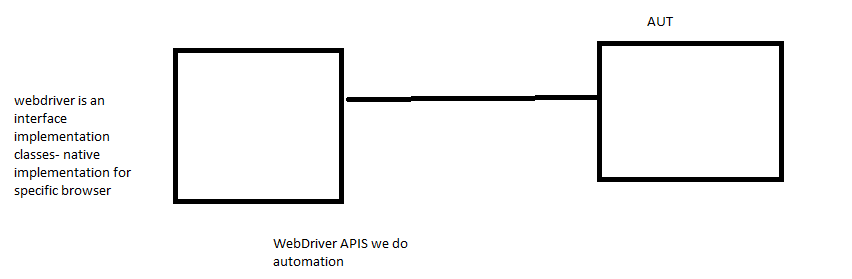
When to automate and when not to automate:

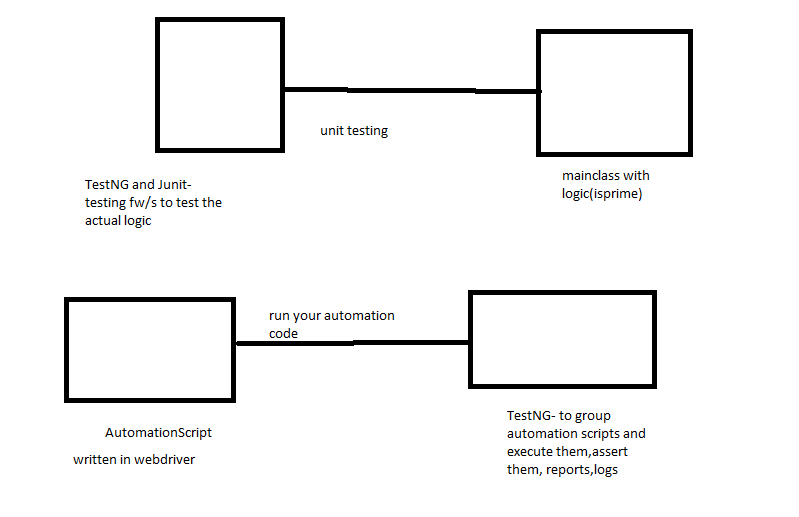
When you do not have time to automate or very tight deadlines to setu a automation framework.

When application look and feel is not stable or it is going to change in mere future.

Tasks which require human intervention cannot be automated.







WebDriver – interface- which has browser specific implementation classes

Eg: FirefoDriver,ChromeDriver,OperaDriver…..

WebDriver declares some important methods which are implemented by its subclasses.

Get

findElement – it return single webelement

this will return NoSuchElementException when element is not present in html page

findElements- list of webelements

this will return empty list when element is not present in html page

getTitle

getCurrentUrl

getPageSource

getWindowHandle

getWindowHandles

switchto()

close

quit

driver.manage.window.maximize

WebElement findElement(By by):

Click

Submit

sendKeys

clear – its used to clear textbox content

List<WebElements> findElements(By by)

By – is a class which is used to locate the elements. It has 8 different locator static methods.

By.id(“stringparam”)

By.name(“stringparam”)

By.tagName(“stringparam”)

By.class(“stringparam”)

By.linkText(“stringparam”)

By.partialLinkText(“stringparam”)

By.xpath(“stringparam”)

By.cssSelector(“stringparam”)

<input type=”text” id=”wblTextBox” class=”txt.box” >

<a id=”lid” href=”login.html” name=”linkname”>

<input type=’submit’

And element is inside form

Click will in all scenarios—inside or outside form

Select class – to select dropdowns and multiple select

Radio button click

WebElement methods…

Waits : Webdriver allows the programmer to wait for the element/elements to be located before throwing the exception like NoSuchElement/ElementNotFound exceptions…

1. Implicit wait – waits for entire time the browser is open by the driver…

It checks initially before the wait time for presence of element- if it do not find the element it will wait for given time and at the end of wait time it again checks for element (it do not check in between), if it finds the elemnt it return the element else it throws exception.

driver.manage().timeouts().implicitlyWait(20, TimeUnit.seconds);

1. Explicit Wait – waits for a specific condition or specific element for a given a amount of time before throwing the exception.

It checks in between the wait time for presence of element by default for every 500 ms and if it finds the element after this specific time it returns the element and do not wait further.

We can use until method with function/predicate or ExpectionConditions.

WebDriverWait wait = **new** WebDriverWait(driver,20);

WebElement element =wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//\*[text()='ASP.NET Web Forms']")));

a.FluentWait – it is similar to explicit wait but it allows to mention the frequency for which the element presence should be tested instead of going with WebDriver default frequency(500ms), along with wait time.

Also you can declare any specific exception to be ignored

20 secoonds – wait time

pollinEvery/frequency – 5 seconds

every 5 seconds it checks for the wait condition and do not for complete wait time if you find the element in between.

FluentWait<WebDriver> wait = **new** FluentWait<WebDriver>(driver)

.pollingEvery(10, TimeUnit.***SECONDS***)

.withTimeout(30, TimeUnit.***SECONDS***)

.ignoring(NoSuchElementException.**class**);

WebElement elem = wait.until(function);

Actions

Mousehover

drapAndDrop

Alerts

Actions – to automate mouse actions like hover, click, contextclick, doubleclick,

Keypress,keyrelease, dragAndDrop…..

JavaScriptExecutor – allows to execute javascript code in webdriver

Eg:

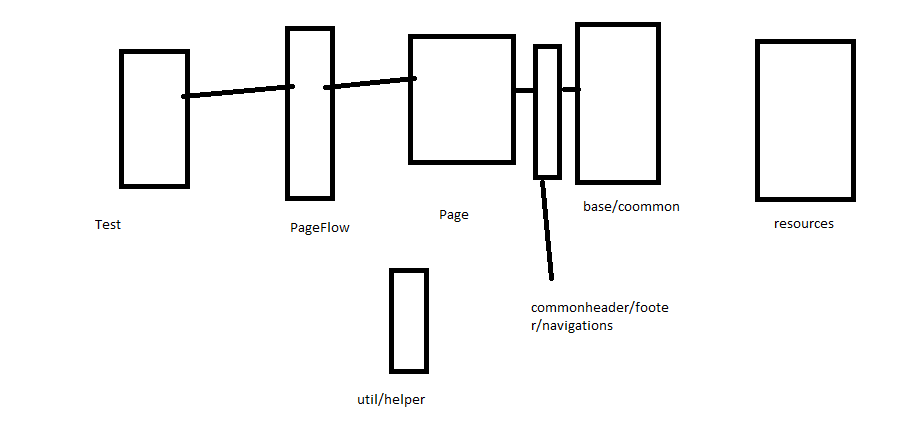
Scrolling down or up using webdriver

Click buttons, pop up some alerts while debugging, refresh the page

FileUpload in selenium:

1. We wil use AutoIt integration with selenium
2. Using Robot class(java.awt package) and selenium

Switchtoframe



Threshold

debug-> info ->warn-> error ->fatal