Application contains lots of dynamic object is called Ajax application.

**Xpath Syntax**

//htmlTag[@attribute=’value’]

//htmlTag[text()=’visibleText’]

**Advantages of Selenium IDE**

* Record and playback tool.
* UI based tool
* Easy to understand and developed the test run
* Selenium IDE code can be exported into Selenium RC and Web Driver Test

How to export:

File -> Export Test -> Java/TestNG/WebDriver

**Selenium RC (Remote Control)**

Selenium RC is the first automation tool in selenium community.

* Selenium RC is a first open source introduced by ThoughtWorks in 2004.
* Selenium RC was Jason Huggins
* Selenium is a collection of JavaScript Libraries. So, it doesn’t contain UI, Record and Playback Option.
* It supports multiple browsers. Like IE, Firefox, Chrome, Safari, Opera
* It also supports multiple operating systems. Example, Linux, Mac and Windows
* It supports multiple languages. (Java, Python, PHP, Ruby, Pers, C# etc)
* But the disadvantages are: No UI, No Record and Play Options.
* It doesn’t support stand alone applications.
* Selenium RC failed to support secured Applications which used Https. Because of SAME-ORIGIN-POLICY Restriction.
* As per the same origin policy, Javascript from one domain cannot execute in another domain.

**Selenium RC Client Library**

Selenium RC test cannot be executed without selenium server. This makes the execution become slower.

Components of Selenium RC:

* Selenium Client Lib: It is a collection of java script library, use to perform action on the browser.
* Selenium Core: It is a JavaScript engine which is use to execute selenium RC commands on the browsers.
* Selenium Server: It is a intermediate component between browser and application. It always receives the request from the selenium RC and Browser and bypass the request to application server.

**Web Driver**

* Web Driver is a collection of core java libraries
* It is not a UI based record and playback tool
* It is Introduced in 2008 by Jason and Stewart from Google
* It is a powerful tool in selenium community because it removes all the drawbacks of selenium RC and selenium IDE
* It supports multiple browsers like IE, Mozilla Firefox, Google Chrome, Opera, etc.
* It supports multiple Platforms, like Windows, Unix, Linux, Android and IOS.
* It supports multiple language, like Java, C#, Python, Perl, PHP, Ruby.
* Web Driver has native support with all the browser, so it can directly communicate with the browser without selenium server.
* Web Driver can support HTTP and HTTPS applications.
* Web Driver is a collection of core java interfaces and classes
* In other words, **WebDriver** is an Interface which extends **SearchContext** Interface and It is Implemented by Browser Classes and **RemoteWebDriver**.
* In order to launch browser we should create object of specific browser java class.
* Whenever we want to run the testScript without browser, we have to use **HtmlDriver** Class. It is also called Head less browser or ghost browser.
* Web Driver Installation required two stages.

**Stage 1:** Download Web Driver Jar from Selenium Community.

* Go to Google and search for download selenium.
* Click on link “ Navigate to Selenium Community”
* Go to Previous Releases
* Click on 2.53 folder
* Click on “[selenium-java-2.53.1.zip](http://selenium-release.storage.googleapis.com/2.53/selenium-java-2.53.1.zip)”
* Download and Unzip the zip file to a folder

**Stage 2:** Import Web Driver Jars into Eclipse

* Open eclipse with new Work Space
* Create new Java Project
* Create a new package
* Create a new Class with main method
* In order to import jar select the project -> right click -> Add External Jars
* Make sure all the jars inside the lib folder and outside the lib folder should be imported into the project.

**Note:**

The current statble version 2.53.1 will support only firefox 45 below version. Because all open source tool will have backward compatibility, but no forward compatability.

In order to run the test script firefox latest version(v59) we should download web driver latest libraries.

**Browser Controls**

WebDriver Methods:

* get()

*Navigate Controls*

* navigate().to()
* navigate().forward()
* navigate().back()
* navigate().refresh()

*Data Capture*

* getTitle()
* gettCurrentUrl()
* getPageSource()
* findElement()

*Element Identification Control*

* findElements()
* getWindowHandles()
* close()

*Window Controls*

* quit()
* manage().window().maximize()
* manage().window().setSize()
* manage().deleteAllCookies()

**findElement()** always take Locator as an argument and it always return web element reference. It is used to navigate to entire HTML document and identify the element by taking reference of locator. It returns web element reference if Element is available.

It throws NoSuchException and stop the execution if object is not available. Find element method can Identify the element based on 8 Locators:

In order to choose locator we should mandatorily look at the HTML source code of the elment.

**Locator**

Locator is a concept in selenium tool which is used to identify specific element in HTML source code based on element attributes and visible text.

In webDriver, Locators are static methods which is implemented inside the **By** Class.

* id()
* name()
* xPath()
* cssSelector
* llinkText()
* partialLinkText()
* className()
* tagName()

note:

* findElement will be always use to identify single element in UI
* findElements is used to identify multiple elements in UI

**id():** is used to identify the location of the Element based on “@id” attribute.

*<input id="identifierId" type="email">* Ex: WebElement textBox = webDriver.findElement(By.*id*("identifierId"));

**name():** is used to identify the location of the Element based on “@name” attribute. Eg: WebElement passwd = webDriver.findElement(By.*name*("password"));

**xpath():** is uded to identify the location of the Element based on “@any” attribute & visible Text. Eg: WebElement next = webDriver.findElement(By.*xpath*("//span[text()='Next']"));

**linkText():** identity the Elemen based on NAME of the LINK, it can be used only for LINK Element. Eg: WebElement link = webDriver.findElement(By.*linkText*("Sign UP")).click(); Note: This locator can not be used, when multiple link present with Same name.

**partialLinkText():** identify the Element based on part of the Link Name. Eg: WebElement link = webDriver.findElement(By.*partialLinkText*("Sign UP")).click();

**className():** is used to identify the location of the Element based on “@class” attribute. Eg: *<span class=”RveJvd snByac”>Next</span>*  WebElement link = webDriver.findElement(By.*className*("Sign UP")).click();

**Note**: In Real time, we will never use thid locator, because multiple elements might have same class

For cross verify: validate the xPath = “//\*[@class=”RveJvd snByac”]

**tagName():** It is used to identify the element using HTML tag, it can be used along findElements() to identify multiple elements.

Eg:

**int** count = webDriver.findElements(By.*tagName*("a")).size();

System.***out***.println(count);

**cssSelector:** Its used to Identify element using “@class” & “@id” attributes. for class= . symbol for id = # symbol

webDriver.findElement(By.*cssSelector*("#identifiedId")).sendKeys("selenium");

***Navigation Controls:***

Navigation Controls are use to perform browser navigation operations by taking help of browser history.

***Code Optimization:***

Getting same output with less number of Java Statement is called as code optimization.

***WebElement Controls***

WebElement is an Interface:

* sendKeys()

*Operational*

*Control*

* clean()
* click()
* submit()
* getText()
* getAttribute()

*Data Capture Control*

* getCssValue()
* getLocation()
* getSize()
* isDisplayed()

*Verification Controls*

* isEnabled()
* isSelected()

Test Cases:

|  |  |  |  |
| --- | --- | --- | --- |
| **Steps** | **Actions** | **Input** | **Expected Result** |
| 1 | Navigate to Actitime Application | URL | Login Page should be displayed |
| 2 | Enter invalid password and click on Login | Username=”admin”  Password=”invalid” | * “Username and password id invalid. Please try again” should be displayed * Msg should be in red color. |
| 3 | Verify Gmail Textbox   * Navigate to gmail login * Verify below attributes: * Capture default (placeholder text) * Capture height and width of the email box. * Capture location of the email box |  |  |

Data Capture Methods:

* getText() method is used to capture visible text from the identified web element, it return string value if element has text or else return empty string.
* getAttribute() method is used to capture any attribute value from the identified web element. It returns String
* getCssValue() method is used to capture style related attribute from the webElement, like color, background color, font-size, etc. It returns String value.
* getLocation() method is use to capture the X and Y coordinate of the element. It always return pointer class object.
* getSize() method is used to get the Dimension of the identified web element. It return Dimension object.

**Web Driver Wait Statement**

**Synchronization** means, whenever web driver try to perform action on the element, some of the element will not get loaded in the UI, In such cases web driver noSuchElementException and stop the entire execution, because of synchronization issue.

**Synchronization Wait**  the process of matching application speed with automation tool speed is called synchronization wait.

There are three synchronization wait statement available in web driver, **implicitly wait, explicitly wait, fluent wait.**

Why we should not use thread wait?

It is a hardcoded wait, it always wait specified amount of time even though page is getting loaded early.

webDriver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

* Implicit wait always wait for html document to be loaded in the UI
* Technically Implicit wait monitor HTML Document while loading the page
* Whenever the page is loaded, within expected duration it releases driver controller to the next line, instead of waiting entire expected duration.
* If page in not loaded even after waiting expected duration, it throws TimeoutException and stops the execution.
* If you insert implicit wait before get method, such wait will be applicable for all the action done by web driver.
* Implicitly wait is use to set maximum lifespan time for driver object.

**Disadvantage of implicit wait**

We cannot use to handle/wait dynamic elements in ajax application.

***Explicitly Wait:*** It always wait for element to be available in UI.Technically Explicitly wait check for the expected element for every 500 miliseconds, If expected element is available within expected duration. It release driver control to the next line instead of waiting entire expected duration. If expected element is not available within expected duration, it throws timeout exception.

It will always use to wait for dynamic object in ajax application, because dynamic object always take some time to code in UI.

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

wait.until(ExpectedConditions.*visibilityOf*(webDriver

.findElement(By.*linkText*(linkText))));

*visibilityOfElementLocated* is a static method available in expected condition class which always wait for element to be appear in UI.

*invisibilityOfElementLocated*  Wait for element to be disappear in UI.

***Fluent Wait:***  It is a extended wait of explicitly wait but we can customize polling wait of for condition check, it means it can verify the expected condition based on user defined time.

***Working with dropdown***

All the above dropdown will be implemented using select and option html tag.

Google search edit box looks like dropdown but it is not a dropdown, because it is not implemented using select html tag.

In order to work with dropdown we should take the help of select class.

Select:

* selectByVisisbleText()

Operational Contorl

* selectByIndex()
* selectByValue()
* deSelectAll()
* deSelectVisibleText()
* deSelectIndex()

Verification Controls

* isMultiple()
* getOptions()

Data Capture Controls

* getAllSelectedOptions()
* getFirstSelectedOption()

Program to work with dropdown by taking help of generic reusable methods

> Create a Class and add two overloaded methods for Select Operations.

Program to work with Multi-Select drop down

> Write a program to find expected value from the dynamic select drop down and select the value if option is available

***Actions***

In order to make use of mouse and keyboard, we should take help of action class.

Methods:

* moveToElement()
* sendKeys()
* contextClick()
* dragAndDrop()
* clickAndHold()
* release()
* build()
* perform()
* doubleClick()

Write a program to work with dropdown menu or Unordered List element.