***SELENIUM WEBDRIVER:-***

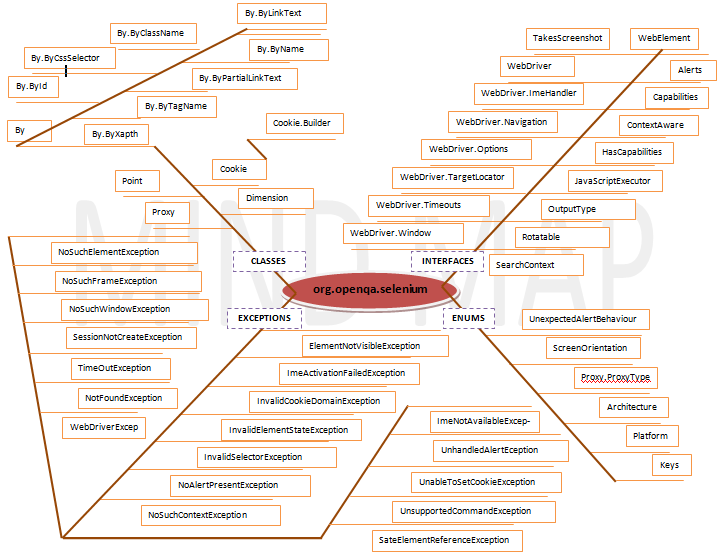
* Selenium web driver is a well-designed object oriented API that provides improved support for web-app testing problems.
* Supports dynamic web pages where the element of page may change even without reloading the page.
* All limitations of Selenium RC has overcome in web driver.

***LIMITIATIONS IN SELENIUM RC:-***

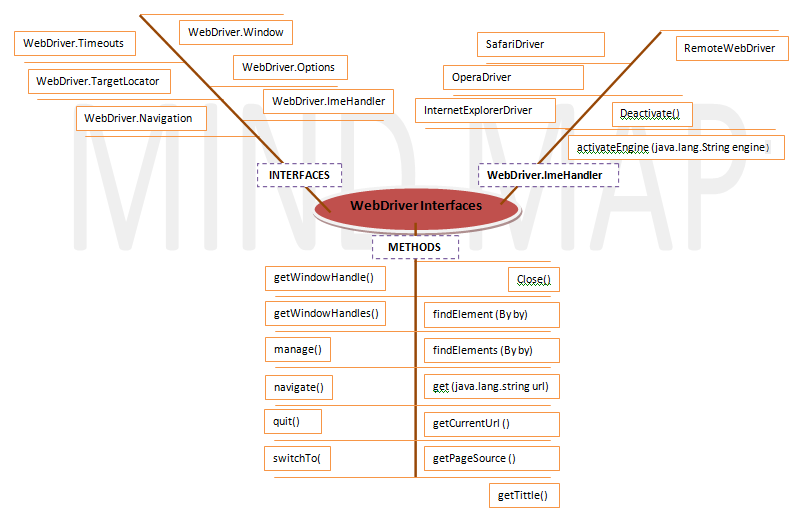
* In Selenium RC, using java script injection there occurs same origin policy issues. Web Driver has overcome this by having drivers for each different browsers.
* It is obvious to have server between application under test & test automation suite in Selenium RC. This makes Selenium RC slow compared to Web Driver.
* Redundancy of commands.
* RC cannot support headless browsers whereas Web driver can. Headless means- Running test suite with help of command line execution or by any network.

***FEATURES OF SELENIUM WEBDRIVER:-***

* It acts as interface between automation script and browsers. It has three categories of methods: control browsers, Web Elements selection, Debugging.
* Selenium Web driver supports all languages as there is provision to create thin wrapper on each language code and binding with them later. This lead to support of ‘Multiple Languages’ and when API supports all languages then it will support ‘Multiple Browsers’ in ‘Multiple Platforms’.
* Webdriver exists in global package called ‘org.openqa.selenium’. ‘Webdriver is an interface which comes under this package and a sub interface of ‘SearchContext’. SearchContext consists of ‘WebDriver’ & ‘WebElement’ as a sub interfaces.



* ‘**Firefox WebDriver**’ is a webdriver, which used to interact with webApps. This means that all drivers should have implementation contracted to WebDriver interface & hence all drivers called as WebDriver.



***BROWSER COMMANDS IN SELENIUM FROM WEBDRIVER OBJECT:-***

* ***get(String arg0) : void*** – This method ***Load***a new web page in the current browser window. Accepts String as a parameter and returns nothing.

**Command**– ***driver.get(appUrl);***

* ***getTitle() : String – This method fetches the Title of the current page. Accepts nothing as a parameter and returns a String value.***

***Command – driver.getTitle();***

* ***getCurrentUrl() : String*** – This method fetches the string representing the ***Current URL*** which is opened in the browser. Accepts nothing as a parameter and returns a String value.

***Command – driver.getCurrentTitle();***

* ***getPageSource() : String*** – This method returns the ***Source Code***of the page. Accepts nothing as a parameter and returns a String value.

***Command – driver.getPageSource();***

* ***close() : void*** – This method ***Close*** only the current window the *WebDriver* is currently controlling. Accepts nothing as a parameter and returns nothing.

***Command – driver.close();***

* ***quit() : void*** – This method ***Closes*** all windows opened by the *WebDriver.* Accepts nothing as a parameter and returns nothing. ***Command – driver.quit();***

***NAVIGATION COMMANDS IN SELENIUM:-***

* ***to(String arg0) : void*** – This method ***Loads***a new web page in the current browser window. It accepts a String parameter and returns nothing.

***Command*** – ***driver.navigate().to(appUrl);***

**It does exactly the same thing as the driver.get(appUrl) method. Where appUrl is the website address to load. It is best to use a fully qualified URL.**

* ***forward() : void*** – This method does the same operation as clicking on the ***Forward Button*** of any browser. It neither accepts nor returns anything.

***Command*** – ***driver.navigate().forward();***

Takes you forward by one page on the browser’s history.

* ***back() : void*** – This method does the same operation as clicking on the ***Back Button*** of any browser. It neither accepts nor returns anything.

***Command*** – ***driver.navigate().back();***

Takes you back by one page on the browser’s history.

* ***refresh() : void*** – This method ***Refresh*** the current page. It neither accepts nor returns anything.

***Command*** – ***driver.navigate().refresh();***

Perform the same function as pressing F5 in the browser.

* **What is Web Element?**

WebElement represents an***HTML element***. HTML documents are made up by HTML elements. HTML elements are written with a ***start*** tag, with an ***end***tag, with the **content** in between: ***<tagname> content </tagname>***

The HTML **element** is everything from the start tag to the end tag: ***<p> My first HTML paragraph. </p>***

HTML elements can be nested (elements can contain elements). All HTML documents consist of nested HTML elements

***WEBELEMENT COMMANDS IN SELENIUM:-***

* ***clear( ) : void*** – If this element is a text entry element, this will clear the value. This method accepts nothing as a parameter and returns nothing.

***Command***– ***element.clear();***

This method has no effect on other elements. Text entry elements are ***INPUT*** and ***TEXTAREA*** elements.

* ***sendKeys(CharSequence… keysToSend ) : void*** – This simulate typing into an element, which may set its value. This method accepts CharSequence as a parameter and returns nothing.

***Command***– ***element.sendKeys(“text”);***

This method works fine with text entry elements like ***INPUT*** and ***TEXTAREA***elements.

* ***click( ) : void*** – This simulates the clicking of any element. Accepts nothing as a parameter and returns nothing.

***Command***– ***element.click();***

Clicking is perhaps the most common way of interacting with web elements like text elements, links, radio boxes and many more**.**

* ***IsDisplayed( ) : boolean*** – This method determines if an element is currently being displayed or not. This accepts nothing as a parameter but returns boolean value(true/false).

***Command***– ***element.isDisplayed();***

NOTE: In some cases, it may that throw NoSuchElementFound exception when element is not present in page. If it is hidden, in this case it returns false. As element is present in DOM but not visible to us.

* ***isEnabled( ) : boolean*** – This determines if the element currently is ***Enabled or not***? This accepts nothing as a parameter but returns boolean value(true/false).

***Command***– ***element.isEnabled();***

This will generally return true for everything but I am sure you must have noticed many disabled input elements in the web pages.

WebElement element = driver.findElement(By.id("UserName"));

boolean status = element.isEnabled();

//Or can be written as

boolean staus = driver.findElement(By.id("UserName")).isEnabled();

//Or can be used as

WebElement element = driver.findElement(By.id("userName"));

boolean status = element.isEnabled();

// Check that if the Text field is enabled, if yes enter value

if(status){

element.sendKeys("ToolsQA");

}

* ***isSelected( ) : boolean*** – Determine whether or not this element is selected or not. This accepts nothing as a parameter but returns boolean value(true/false).

***Command***– ***element.isSelected();***

This operation only applies to input elements such as ***Checkboxes***, ***Select Options*** and ***Radio Buttons***. This returns ***True*** if the element is currently *selected or checked*, ***false*** otherwise.

* ***submit( ) : void***– This method works well/better than the *click()* if the current element is a form, or an element within a form. This accepts nothing as a parameter and returns nothing.

***Command***– ***element.submit();***

If this causes the current page to change, then this method will wait until the new page is loaded.

* ***getText( ) : String***– This method will fetch the visible (i.e. not hidden by CSS) innerText of the element. This accepts nothing as a parameter but returns a String value.

***Command***– ***element.getText();***

This returns an innerText of the element, including sub-elements, without any leading or trailing whitespace.

* ***getTagName( ) : String***– This method gets the tag name of this element. This accepts nothing as a parameter and returns a String value.

***Command***– ***element.getTagName();***

This does not return the value of the name attribute but return the tag for e.g. “***input***“*for the element****<input name="foo"/>***.

* ***getCssvalue( ) : String***– This method Fetch CSS property value of the give element. This accepts nothing as a parameter and returns a String value.

***Command***– ***element.getCssValue();*** Color values should be returned as rgba strings, so, for example if the “background-color” property is set as “green” in the HTML source, the returned value will be “rgba(0, 255, 0, 1)”.

* ***getAttribute(String Name) : String***– This method gets the value of the given attribute of the element. This accepts the String as a parameter and returns a String value.

***Command***– ***element.getAttribute();***

Attributes are Ids, Name, Class extra and using this method you can get the value of the attributes of any given element.

* ***getSize( ) : Dimension***– This method fetch the width and height of the rendered element. This accepts nothing as a parameter but returns the Dimension object.

***Command***– ***element.getSize();***

This returns the size of the element on the page.

WebElement element = driver.findElement(By.id("SubmitButton"));

Dimension dimensions = element.getSize();

System.out.println(“Height :” + dimensions.height + ”Width : "+ dimensions.width);

* ***getLocation( ) : Point***– This method locate the location of the element on the page. This accepts nothing as a parameter but returns the Point object.

***Command***– ***element.getLocation();***

This returns the ***Point object***, from which we can get X and Y coordinates of specific element.

WebElement element = driver.findElement(By.id("SubmitButton"));

Point point = element.getLocation();

System.out.println("X cordinate : " + point.x + "Y cordinate: " + point.y);

***LOCATE ELEMENTS IN SELENIUM:-***

Locating elements in WebDriver is done by using the ***findElement(By.locator())***method. The ***findElement*** methods take a locator or query object called ‘***By’***.

* ***id(String id) : By*** – This is the most efficient and preferred way to locate an element, as most of the times *IDs* are unique. It takes a parameter of String which is a *Value of ID attribute* and it returns a ***BY object*** to ***findElement()***method.

***Command***– ***driver.findElement(By.id(“Element ID”));***

With this strategy, If no element has a matching id attribute, a ***NoSuchElementException*** will be raised.

* ***name(String name) : By*** – This is also an efficient way to locate an element but again the problem is same as with ID that UI developer make it having non-unique names on a page or auto-generating the names. It takes a parameter of String which is a *Value of NAME attribute* and it returns a ***BY object*** to ***findElement()***method.

***Command***– ***driver.findElement(By.name(“Element NAME”));***

With this strategy, the first element with the name attribute value matching the location will be returned. If no element has a matching name attribute, a ***NoSuchElementException***will be raised.

* ***className(String className) : By*** – This finds elements based on the value of the *CLASS* attribute. It takes a parameter of String which is a *Value of CLASS attribute* and it returns a ***BY object*** to ***findElement()*** method.

***Command***– ***driver.findElement(By.className(“Element CLASSNAME”));***

If an element has many classes then this will match against each of them.

* ***tagName(String name) : By*** – With this you can find elements by their *TAGNAMES*. It takes a parameter of String which is a *Value of TAG attribute* and it returns a ***BY object*** to ***findElement()*** method.

***Command***– ***driver.findElement(By.tagName(“Element TAGNAME”));***

Locating Element By Tag Name is not too much popular because in most of cases, we will have other alternatives of element locators. But yes if there is not any alternative then you can use element’s DOM Tag Name to locate that element in *WebDriver*.

* ***linkText(String linkText) : By*** – With this you can find elements of *“****a****” tags(****Link****)* with the link names. Use this when you know link text used within an anchor tag. It takes a parameter of String which is a *Value of LINKTEXT attribute*and it returns a ***BY object*** to ***findElement()*** method.

***partialLinkText(String linkText) : By*** – With this you can find elements of *“****a****” tags(****Link****)* with the partial link names.

***Command***– ***driver.findElement(By.linkText(“Element LINKTEXT”));***

***Command***– ***driver.findElement(By.partialLinkText(“Element LINKTEXT”));***

If your targeted element is link text then you can use by link text element locator to locate that element. Partial Link Text is also same as Link text, but in this we can locate element by partial link text too. In that case we need to use ***By.partialLinkText*** at place of ***By.linkText.***

* ***xpath(String xpathexpression) : By*** – It is most popular and majorly used locating element technique or the easiest way to locate element in WebDriver. It takes a parameter of String which is a *XPATHEXPRESSION*and it returns a ***BY object*** to ***findElement()*** method.

***Command***– ***driver.findElement(By.xpath(“Element XPATHEXPRESSION”));***

***DIFFERENCE OF FIND ELEMENT() & FIND ELEMENTS():-***

***findElement()***

* *On Zero Match : throws NoSuchElementException*
* *On One Match : returns WebElement*
* *On One+ Match : returns the first appearance in DOM*

***findElements()***

* *On Zero Match : return an empty list*
* *On One Match : returns list of one WebElement only*
* *On One+ Match : returns list with all matching instance*

***SUMMARY OF FINDING ELEMENTS USING BY:-***

