**WebDriver Architecture**

* WebDriver implemented on Layered Design, the idea behind this implementation is more and more usage of WebDriver for automation and this could be possible by fitting best fit languages.
* Implementation of WebDriver is that each browser has a language that is most natural to use when attempting to drive it. Drivers are built as per the best fit language and we can just see the wrapper around them.
* We can say that for any of browser driver if any of the features works there in one binding language then it should be easy to add support to other binding languages also.
* Web Driver is a compact Object Oriented API which can directly interacts with the Application under tests.
* WebDriver utilizes the browser native compatibility to automation without using any peripheral entity.



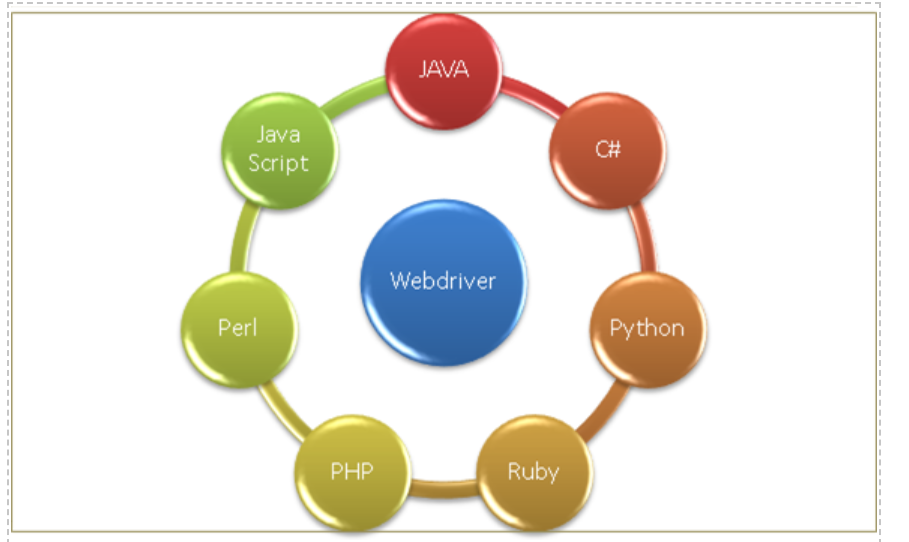
**Features of WebDriver**

1) “Interface WebDriver”, which represents an idealized web browser used for testing. Three categories of methods in this class.

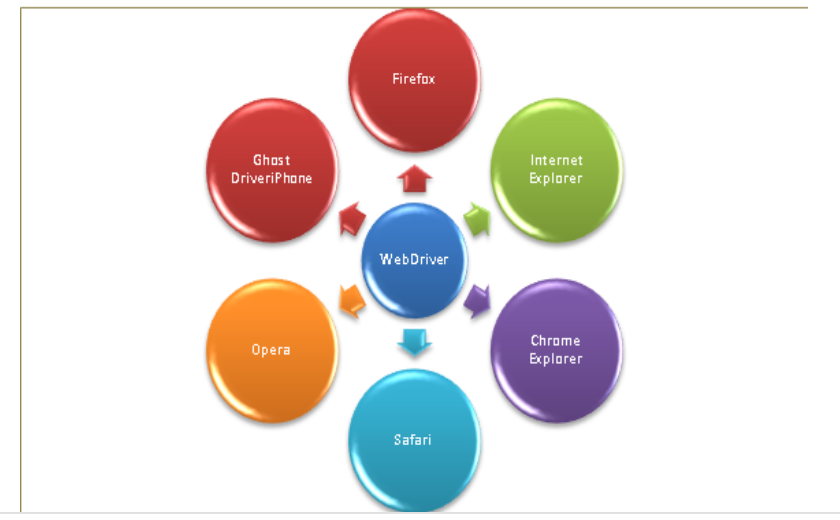
* *Control to browser*
* *WebElements selection*
* *Debugging*

2) We discussed above about language bindings with browsers and it is just a thin wrapper which is exposed for us to write code as per our needs. This leads *WebDriver* to support “***Multiple Languages***” as well as “***Multiple Support Browsers***” which means that if any API supports multiple languages then this automatically leads to “***Multiple Platforms***”.

***Multiple Languages Binding* SUPPORT**



***Multiple Browsers* SUPPORT**

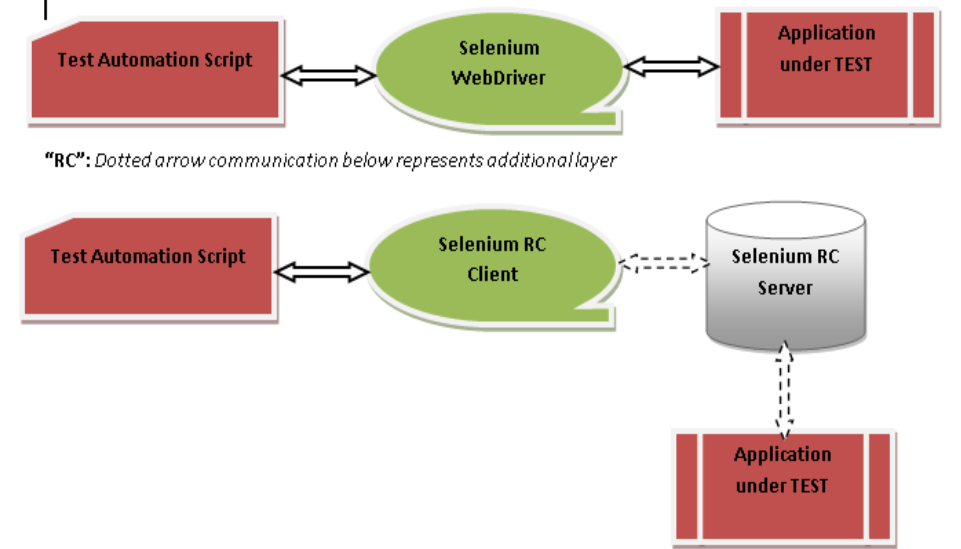


***Multiple Platforms* SUPPORT**



***Speed difference between RC and WebDriver***

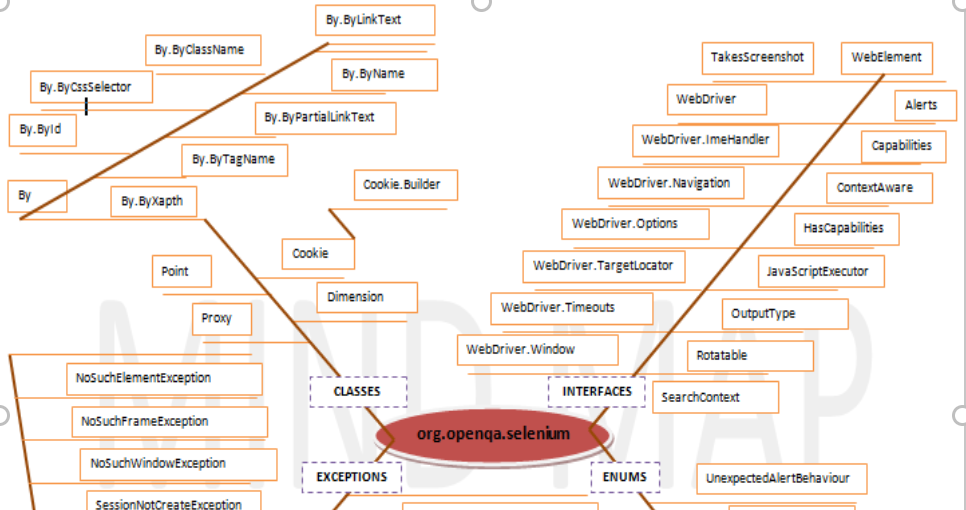
*WebDriver* one to one communication with “*Application under Test*” makes it faster than *Remote Control* as RC creates one more interface between communications which is ***RC Server*** makes it likes *Client Server Communication*. Snapshot below:-

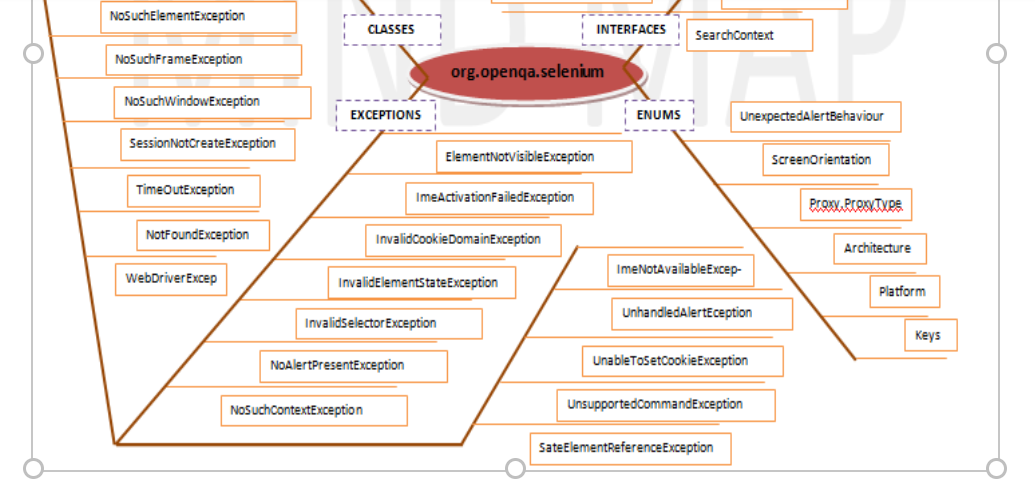
 

**WebDriver Existence in package *org.openqa.selenium***

***org.openqa.selenium***: The global package of Selenium which we import in our program every time when it is needed to use *WebDriver*. For reference, please see “*org.openqa.selenium*” mind map below :-

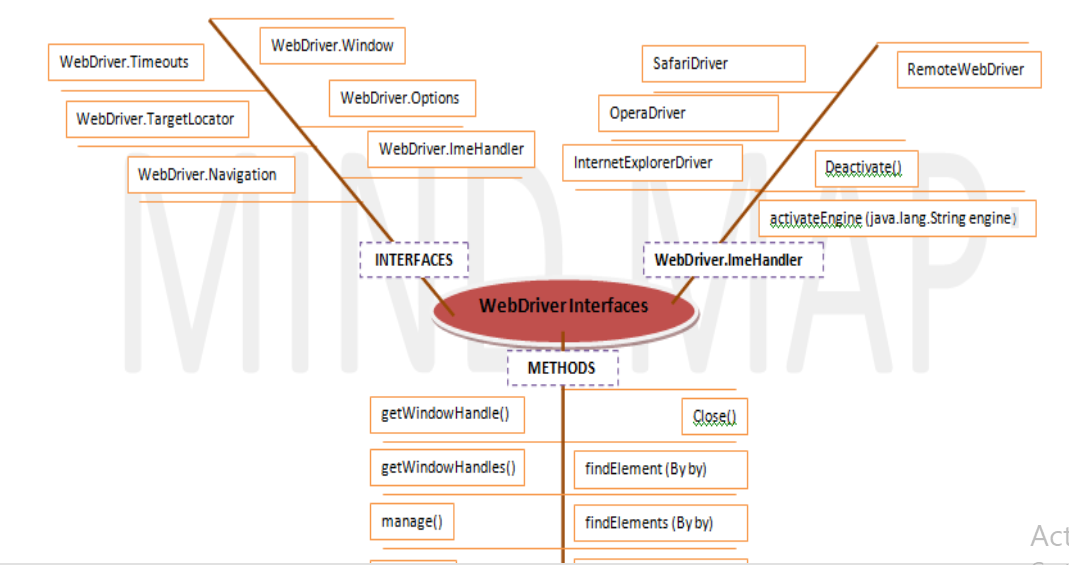
***WebDriver****: Is an interface comes under this package of Selenium and a sub interface of “****SearchContext****”. “SearchContext” consists of “WebDriver” & “WebElement” as a sub interfaces.*

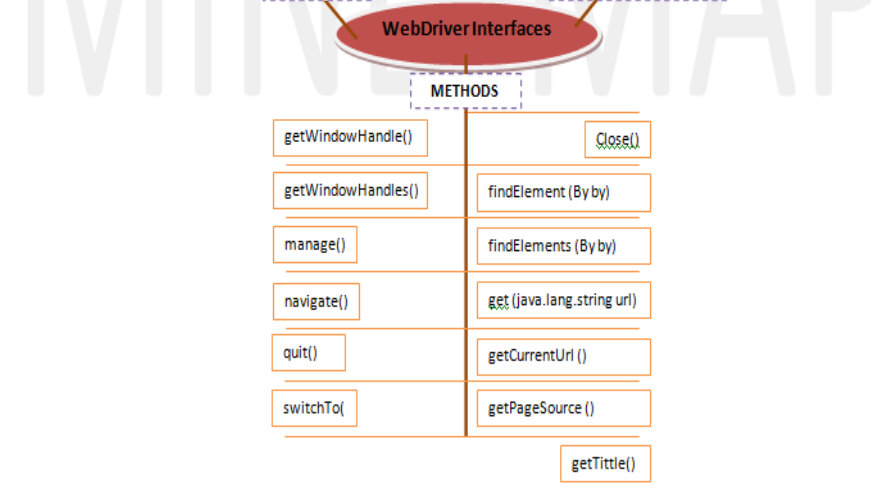




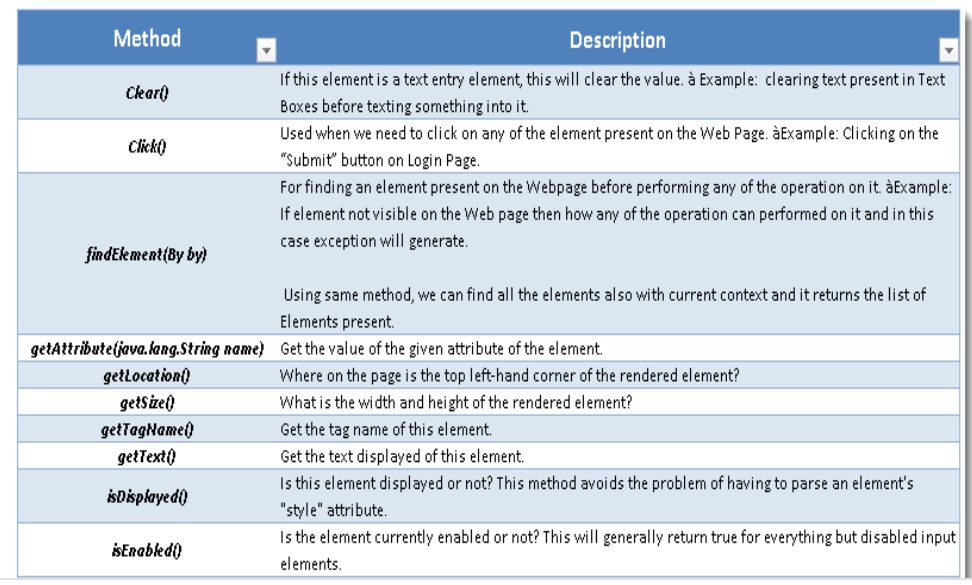
**WebDriver Interface**

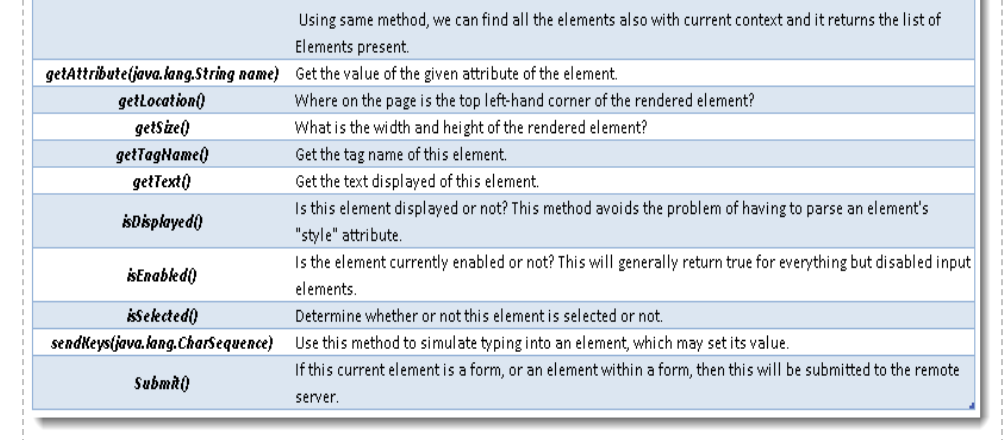
Let’s see what all *WebDriver* contains, for example: – *What is “FirefoxDriver”*: is a *WebDriver* which is used to interact with WebApps. So, this means that all the drivers should have all the implementations which are contracted in the *WebDriver interfaces* and all the *Drivers* can be called as *WebDriver*. See Figure below: –





* *At high level if we can say that Driver means as per the selection of the Explorer driver will be selected and Driver will be used to interact with Web Page.*
* *Web Page composed of Web Elements and these Drivers’s will communicate with the Web Page.*
* *For communicating with Web Page means communication with Elements present on the Web Page like for example: “Textboxes”, Buttons”, “Links” etc. “WebElement” a sub interface of “SearchContext” comes into picture. Let’s see this Interface also (Methods name and description in the table below).*





Above table is providing us all the methods which we can use to interact with ***WebElements***on Web Page but now for performing any of the operation on Web Element, we need to find element first on Web Page as per the above table and method ***findElement()***. For finding element on Web Page, We need address/locator of the element and we can achieve this.

In detail below that what all WebDriver comprises of.

**WebDriver Browser Commands**

Get Command

***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);***

Where ***appUrl***is the website address to load. It is best to use a fully qualified URL.

Get Title Command

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

***Command – driver.getTitle();***

Get Current URL Command

***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();***

Get Page Source Command

***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 Command

***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 the browser if it’s the last window currently open.

Quit Command

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

***Command – driver.quit();***

**Browser Navigation Commands**

Navigate To Command

***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 Command

***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 Command

***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();***

Refresh Command

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

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

**WebElement Commands**

***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 Command

***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 Command

***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 Command

***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();***

IsEnabled Command

***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.

IsSelected Command

***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.

GetText Command

***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 Command

***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 Command

***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 Command

***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 Command

***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.

**FindElement and FindElements Command**

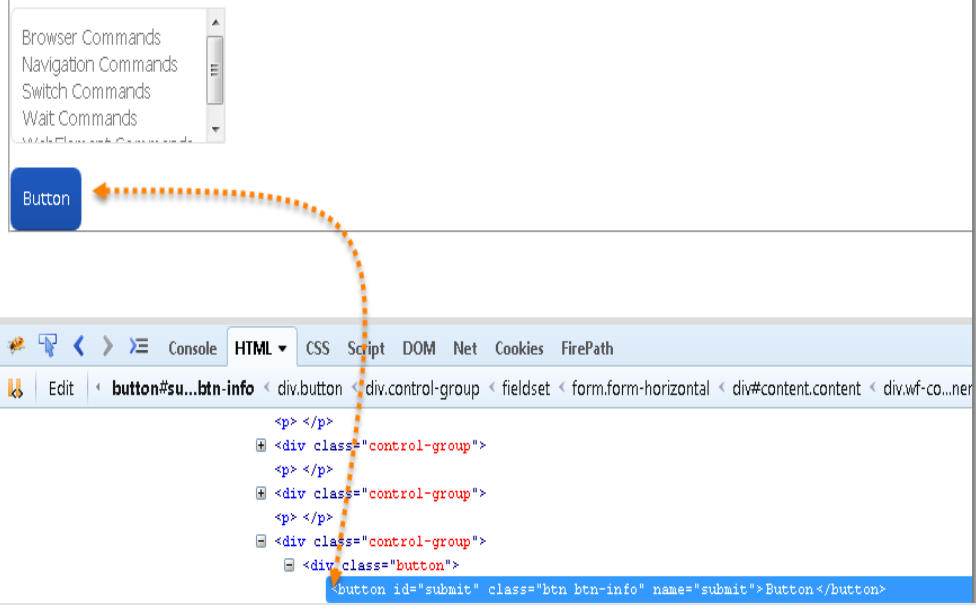
By ID

***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.

***Example***: If an element is given like this:



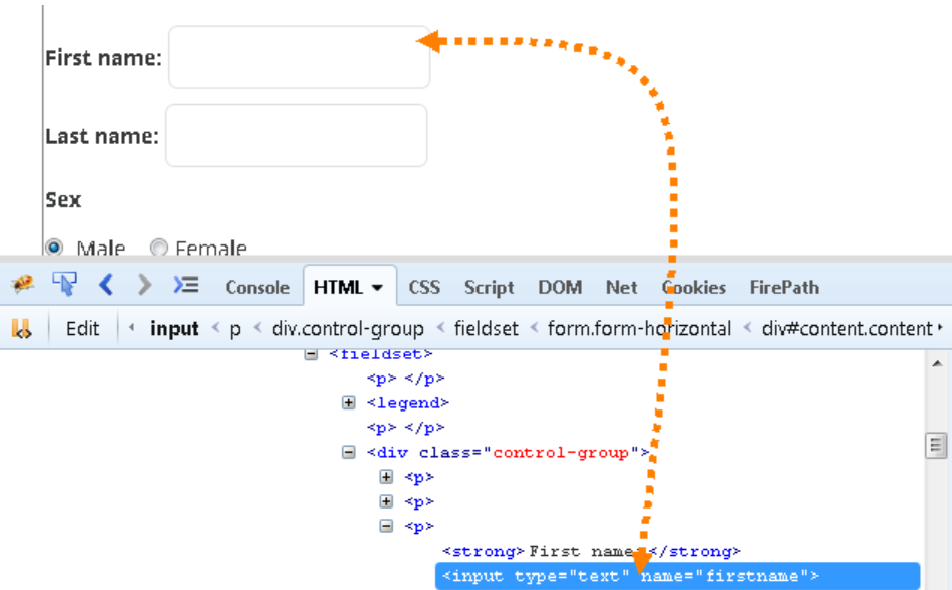
By Name

***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.

***Example***: If an element is given like this:



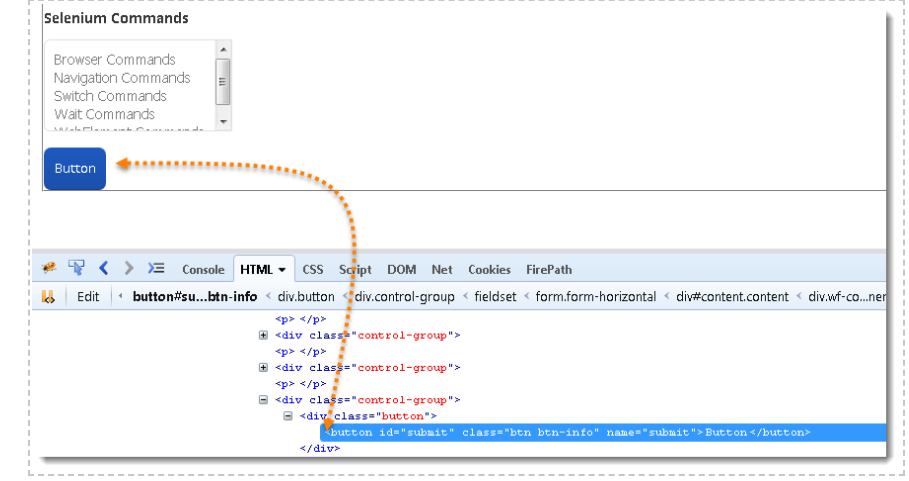
By ClassName

***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.

***Example***: If an element is given like this:



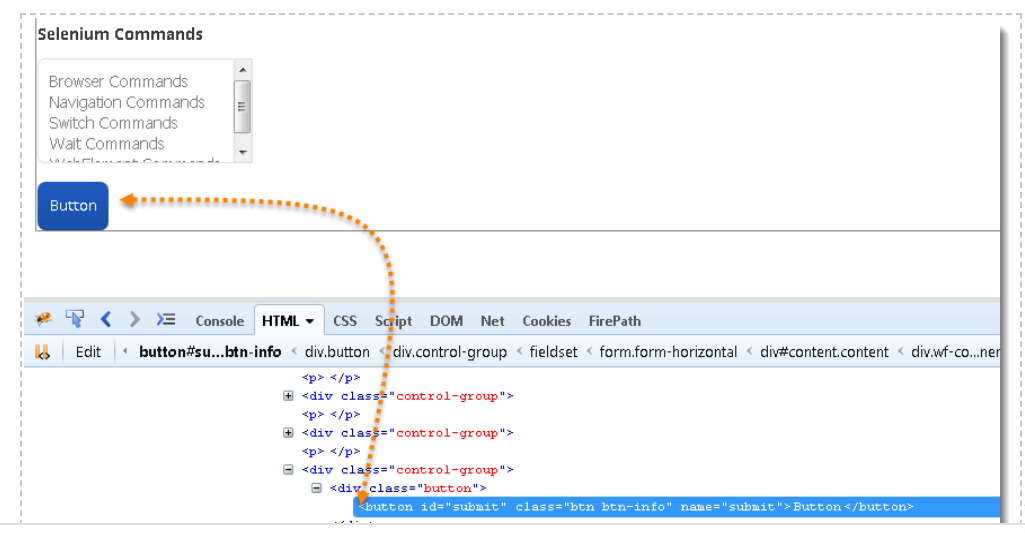
By TagName

***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*.

***Example***: If an element is given like this:

By LinkText & PartialLinkText

***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.

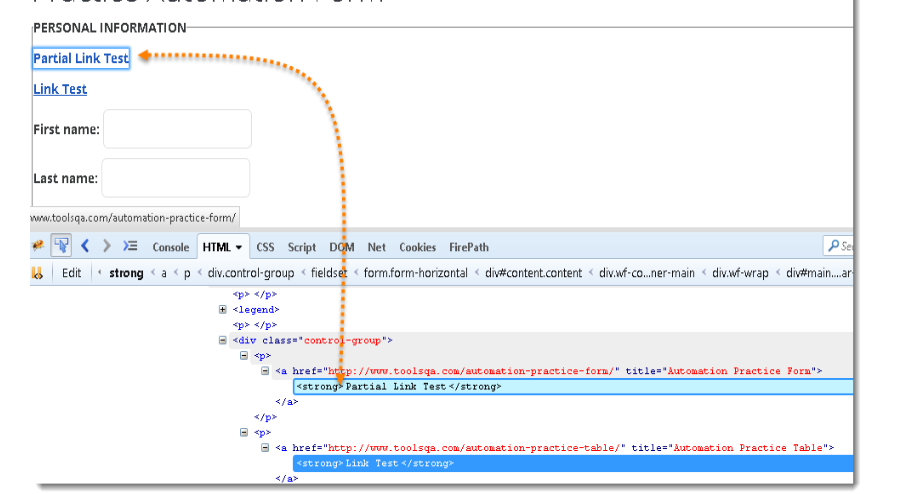
***partialLinkText(String linkText) : By*** – With this you can find elements of *“****a****” tags(****Link****)* with the partial link names.

***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.***

***Example***: If an element is given like this:



By XPath

***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”));***

The best thing in xpath is that it provides many different technique to locate elements. It gives you feature to locate single element in many ways.

We have a complete chapter on ***XPath techniques*** which we will come across during our learning journey on ToolsQA latter. 

Difference between FindElement & FindElements Commands

The difference between ***findElement()*** and ***findElements()*** method is the first returns a *WebElement* object otherwise it throws an exception and the latter returns a *List of WebElements*, it can return an empty list if no DOM elements match the query.

***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

