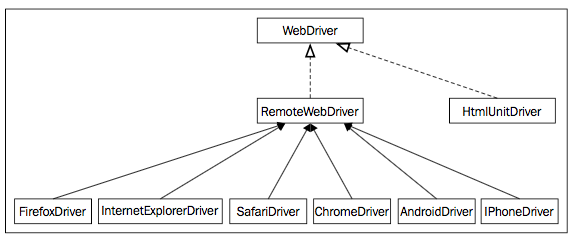
**Selenium Basics**

**Launching Browser:**

**System.setProperty(“webdriver.gecko.driver”, “path of the driver executable”);**

WebDriver driver = new FirefoxDriver();

In the above line of code, we instantiate the Firefox implementation of the WebDriver interface. *WebDriver* is an interface whose implementation is done in *RemoteWebDriver* and HtmlUnitDriver classes. *FirefoxDriver* is a subclass of *RemoteWebDriver* class, which extends the *RemoteWebDriver* class more specifically for the *FirefoxDriver*. Similarly, we have the *InternetExplorerDriver*, *ChromeDriver*, *SafariDriver*, *AndriodDriver* and *IPhoneDriver* classes, which are specific implementations for the respective browsers and devices.



**Browser Commands:**

**get(String arg0)... void**

This method *Loads* a web page in a current browser window. This method accepts URL of the website as String argument and return void.

*Syntax*:

*driver.get(URL of the website);*

**getTitle()... String**

This method fetches the title of the current browser window and returns the title value to a String variable.

*Syntax*:

*String titleVal = driver.getTitle();*

**getCurrentUrl()... String**

This method fetches the current url of the browser window and stores in a string variable.

*Syntax*:

String urlVal = driver.getCurrentUrl();

**getPageSource... String**

This method fetches the source code of the page and store in a string variable.

*Syntax*:

String pageSourceVal = driver.getPageSource();

**close()... void**

This method close the current browser window. This method doesn’t accept any arguments and doesn’t return anything.

*Syntax*:

driver.close();

**quit()... void**

This method closes all the browser windows opened by the webdriver instance. This method doesn’t accept any arguments and doesn’t return anything.

*Syntax*:

driver.quit();

**navigate().to(String url)... void**

This method Loads the url provided as String argument in a new browser window. This method accepts String argument and returns nothing.

*Syntax*:

driver.navigate().to(url of the website);

**navigate().back()... void**

This method performs the back button operation of the any browser. This method neither accepts an argument nor returns.

*Syntax*:

driver.navigate().back();

**navigate().forward()... void**

This method performs the forward button operation of the any browser. This method neither accepts nor returns anything.

*Syntax*:

driver.navigate().forward();

**Working with different browsers:**

**Work with Firefox browser(Selenium 3)**

* Download the gecko driver executable file from seleniumhq.org.
* Set the system property “webdriver.gecko.driver” to path of the gecko driver executable file.
* Create the instance of the FirefoxDriver class

*Syntax*:

System.setProperty(“webdriver.gecko.driver”, ”path of the gecko driver”);

WebDriver driver = new FirefoxDriver();

**Work with Chrome Browser**

* Download the chrome driver executable file from seleniumhq.org.
* Set the system property “webdriver.chrome.driver” to path of the chrome driver file
* Create the instance of the ChromeDriver class.

*Syntax*:

System.setProperty(“webdriver.chrome.driver”, ”path of the chrome driver”);

WebDriver driver = new ChromeDriver();

**Work with IE Browser**

* Download the internet explorer driver executable file from seleniumhq.org.
* Set the system property “webdriver.ie.driver” to path of the internet explorer driver executable file.
* Create the instance of the InternetExplorerDriver class

*Syntax*:

System.setProperty(“webdriver.ie.driver”, ”path of the gecko driver”);

WebDriver driver = new InternetExplorerDriver();

**work with Safari Browser**

Working with safari browser is little complicated compared with other browsers. To work with safari browser follow the below instructions in safari browsers

Ensure that the Develop menu is available. It can be turned on by opening Safari preferences (*Safari > Preferences* in the menu bar), going to the *Advanced* tab, and ensuring that the *Show Develop menu in menu bar* checkbox is checked.

Enable Remote Automation in the Develop menu. This is toggled via *Develop > Allow Remote Automation* in the menu bar.

Authorize safaridriver to launch the webdriverd service which hosts the local web server. To permit this, run /usr/bin/safaridriver –enable once manually and complete the authentication prompt.

*Syntax*:

WebDriver driver = new SafariDriver();

**WebElements**

A web page is a combination of many different HTML elements, like images, buttons, tables, links, labels, forms, edit boxes, paragraphs, dropdown boxes and so on, these elements are WebElements in the context of WebDriver.

**Locating web elements using web driver**

**findElement()**

WebDriver’s findElement() method is a way to locate an element in the web page.

*Syntax*:

WebElement findElement (By by)

The input parameter for the *findElement()* method is the *By* instance and returns WebElement instance that represents actual HTML element or component of the web page. If WebDriver doesn’t find the element, it throws a runtime exception named NoSuchElementException.

**FindElements()**

WebDriver’s findElements() method is way to locate zero or more number of web elements for given locating mechanism on a web page.

*Syntax*:

java.util.List<WebElement> findElements(By by)

The input parameter is same as the findElement() method, which is an instance of by class but returns an empty list if no element is found, and returns a list of elements if there is multiple elements. If it doesn’t find any element it doesn’t throw any Exception.

**Using By class to locate the elements**

There are 8 different ways to identify an HTML element on a web page. They are

|  |  |
| --- | --- |
| id | By.id() each element is uniquely identified by an ID, if provided. |
| name | By.name() element is identified by name attribute |
| linkText | By.linkText() used to identify HTML links by providing the complete link text. |
| partialLinkText | By.partialLinkText() same as above but will provide partial link text. |
| tagName | By.tagName() identifies list of elements with the tagName. |
| className | By.className() identifies a list of elements which a the same class name attribute. |
| xpath | By.xpath() identifies the element with the specified xpath. |
| cssSelector | By.cssSelector() identifies the element with the specified cssSelector. |

Note: Locating elements using tagName and className is little bit different compared with the other locating mechanisms. With the same tagName and className there will be number of elements in the page and those elements will be stored in a List.

**Actions on different web elements**

**getAttribute(String arg0)... String**

This method is applicable on any web element and is used to fetch the value of the attribute provided as an argument. This method accepts attribute name as String arguments and returns value of the attribute as a String.

*Syntax*:

String attributeValue = getAttribute(attribute name)**;**

**getCssValue(String arg0)... String**

This method is applicable on any web element and is used to fetch the CSS properties values like font-family, background-color, color, margin and so on of the given element. This is useful when we want to validate the CSS style that are applied on elements.

*Syntax:*

String cssValue = getCssValue(css property name);

**sendKeys(CharSequence)... void**

This method is applicable for text box or text area HTML elements and is used to type text into the text box.

*Syntax*:

void sendKeys(CharSequence keysToSend);

**clear()... void**

This method is applicable for text box or text area elements and is used to erase the text ehat is entered in a WebElement using sendKeys() method.

*Syntax*:

void clear();

**submit()... void**

This method is applicable on a form or on an element, which is inside a form and is used to submit a form of a web page to the server hosting the web applications.

*Syntax:*

void submit();

**click()... void**

This method is applicable on any element. In general is used to click on buttons, links and images.

*Syntax:*

void click();

**getLocation()... Point**

This method is applicable on any web element and is used to get the relative position of an element where it is rendered on the web page. This position is calculated relative to the top-left corner of the web page of which the (x, y) coordinated are assumed as (0, 0).

*Syntax:*

Point location = getLocation();

**getSize()... Dimension**

This method is applicable on all visible elements. It will return the width and height of the rendered WebElement.

*Syntax:*

Dimension size = getSize();

**getText()... String**

This method is applicable on any element. This method will returns the visible inner text string of the web element if anything is available or else will return an empty string.

*Syntax;*

String elementText = getText();

**getTagName()... String**

This method is applicable on any web element. This will returns the tag name of the target web element.

*Syntax:*

String tagName = getTagName();

**isDisplayed... boolean**

This method is applicable on any web element. This method verifies if an element is displayed on the web page.

*Syntax:*

boolean value = isDisplayed();

**isEnabled()... boolean**

This method is applicable on any web element. This method verifies if an element is enabled on web page or not.

*Syntax:*

boolean value = isEnabled();

**isSelected()... boolean**

This method is applicable on radio button, options in select and checkbox web elements. This method verifies if an element is selected right now on the web page. It returns true if an element is selected or returns false.

*Syntax:*

boolean value = isSelected();

*Note: To interact with dropdown lists and list boxes, WebDriver support class called “Select” which provides useful methods to select and de-select the options. The following are the methods of Select class to select options.*

**selectByVisibleText(String arg0)... void**

Select an option that display visible text (text display in drop down) matching the given argument.

*Syntax:*

selectByVisibleText(text of an option which you want to select in dropdown);

**selectByIndex(int arg0)... void**

Select an option based on the index of given by the user.

*Syntax:*

selectByIndex(index of the option which you want to select in dropdown);

**selectByValue(String arg0)... void**

Select an option that matches the value of the given argument.

*Syntax:*

selectByValue(value of the option which you want to select in dropdown);