**Locators in Selenium**

There are a total of 8 locators in Selenium WebDriver-

1. **By Id** - Locates element using id attribute of the web element.

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

1. **By className** - Locates the web element using className attribute.

WebElement element = driver.findElement(By.className("elementsClass"));

1. **By tagName** - Locates the web element using its html tag like div, a, input etc.

WebElement element = driver.findElement(By.tagName("a"));

1. **By name** - Locates the web element using name attribute.

WebElement element = driver.findElement(By.name("male"));

1. **By linkText** - Locates the web element of link type using their text.

WebElement element = driver.findElement(By.linkText("Click Here"));

1. **By partialLinkText** - Locates the web element of link type with partial matching of text.

WebElement element = driver.findElement(By.partialLinkText("Click"));

1. **By cssSelector** - Locates the web element using css its CSS Selector patterns(explained in detailed here - [CSS Locators](http://artoftesting.com/automationTesting/css-selector-in-selenium-webdriver.html)).

WebElement element = driver.findElement(By.cssSelector("div#elementId"));

1. **By xpath** - Locates the web element using its XPaths(explained in detailed here [XPath Locators](http://artoftesting.com/automationTesting/xpath-in-selenium-tutorial.html)).

WebElement element = driver.findElement(By.xpath("//div[@id=’elementId’]"));

## Opening a URL

### Using Get method-

The driver.get() method is used to navigate to a web page by passing the string URL as parameter. Syntax-

driver.get("http://artoftesting.com");

Using Navigate method-

The driver.navigate().to() method does the task of opening a web page like driver.get() method. Syntax-

driver.navigate().to("http://artoftesting.com");

## Clicking on WebElements

The click() method in Selenium is used to perform the click operation on web elements. In our previous tutorial [Locators in Selenium WebDriver](http://artoftesting.com/automationTesting/locatorsInSelenium.html), we studied about locating the webElements in Selenium. The click() method is applied on the webElements identified, to perform the click operation.

*//Clicking an element directly*

driver.findElement(By.id("button1")).click();

*//Or by first creating a WebElement and then applying click() operation*

WebElement submitButton = driver.findElement(By.id("button2"));

submitButton.click();

**Writing in a Textbox**

The sendKeys() method can be used for writing in a textbox or any element of text input type.

*//Creating a textbox webElement*

WebElement element = driver.findElement(By.name("q"));

*//Using sendKeys to write in the textbox*

element.sendKeys("ArtOfTesting!");

**Clearing text in a Textbox**

The clear() method can be used to clear the text written in a textbox or any web element of text input type.

*//Clearing the text written in text fields*

driver.findElement(By.name("q")).clear();

**Fetching text written over any web element**

In automation, many a times we need to fetch the text written over a web element for performing some assertions or debugging. For this, we have getText() method in selenium webDriver.

*//Fetching the text written over web elements*

driver.findElement(By.id("element123")).getText();

**Navigating backwards in a browser**

Selenium provides navigate().back() command to move backwards in the browser's history.

*//Navigating backwards in browser*

driver.navigate().back();

**Navigating forward in a browser**

Selenium provides navigate().forward() command to move forward in a browser.

*//Navigating forward in browser*

driver.navigate().forward();

**Refreshing the browser**

There are multiple ways to refresh a page in Selenium WebDriver-

* Using **driver.navigate().refresh()** command
* Using **sendKeys(Keys.F5)** on any textbox on the webpage
* Using **driver.get("URL")** with current URL
* Using **driver.navigate().to("URL")** with current URL

*//Refreshing browser using navigate().refresh()*

driver.navigate().refresh();

*//By pressing F5 key on any textbox element*

driver.findElement(By.id("id123")).sendKeys(Keys.F5);

*//By opening the current URL using get() method*

driver.get("http"*//artoftesting.com");*

*//By opening the current URL using navigate() method*

driver.navigate().to("http://artoftesting.com");

**Closing browser**

Selenium provides two commands to close browsers close() and quite(). The driver.close() command is used to close the browser having focus. Whereas, the driver.quite command is used to close all the browser instances open.

*//To close the current browser instance*

driver.close();

*//To close all the open browser instances*

driver.quit();

**Sample Script**

You can use the sample script below to automate the dummy [webpage](http://www.artoftesting.com/sampleSiteForSelenium.html). The comments mentioned throughout the script will guide you through whole automation process. Also, note that the Thread.sleep() used in the sample script is to pause the automation in between events. This Thread.sleep() is not required and is only included to provide you some time to see the automation events.  
Download this java file here [seleniumBasicCommands.java](http://artoftesting.com/automationTesting/sampleScript/seleniumBasicCommands.java) (right click on 'seleniumBasicCommands.java' and click on 'save link as...' to save the sample selenium test script).

**package** myTestPackage;

**import** **java.util.concurrent.TimeUnit**;

**import** **org.openqa.selenium.By**;

**import** **org.openqa.selenium.WebDriver**;

**import** **org.openqa.selenium.WebElement**;

**import** **org.openqa.selenium.firefox.FirefoxDriver**;

**import** **org.openqa.selenium.support.ui.Select**;

**import** **org.testng.Assert**;

**import** **org.testng.annotations.Test**;

**public** **class** **seleniumBasicCommands** {

**public** **static** **void** **main**(String Args[]) **throws** InterruptedException{

//Create Firefox driver's instance

WebDriver driver = **new** FirefoxDriver();

//Set implicit wait of 10 seconds

//This is required for managing waits in selenium webdriver

driver.manage().timeouts().implicitlyWait(**10**, TimeUnit.SECONDS);

//Launch sampleSiteForSelenium

driver.get("http://www.artoftesting.com/sampleSiteForSelenium.html");

//Fetch the text "This is sample text!" and print it on console

//Use the id of the div to locate it and then fecth text using getText() method

String sampleText = driver.findElement(By.id("idOfDiv")).getText();

System.out.println(sampleText);

//Waiting for 3 seconds just for user to efficiently check automation

//Its not mandatory though

Thread.sleep(**3000**);

//Using linkText locator to find the link and then using click() to click on it

driver.findElement(By.linkText("This is a link")).click();

Thread.sleep(**3000**);

//Finding textbox using id locator and then using send keys to write in it

driver.findElement(By.id("fname")).sendKeys("Kuldeep Rana");

Thread.sleep(**3000**);

//Clear the text written in the textbox

driver.findElement(By.id("fname")).clear();

Thread.sleep(**3000**);

//Clicking on button using click() command

driver.findElement(By.id("idOfButton")).click();

Thread.sleep(**3000**);

//Find radio button by name and check it using click() function

driver.findElement(By.name("male")).click();

Thread.sleep(**3000**);

//Find checkbox by cssSelector and check it using click() function

driver.findElement(By.cssSelector("input.Automation")).click();

Thread.sleep(**3000**);

//Using Select class for for selecting value from dropdown

Select dropdown = **new** Select(driver.findElement(By.id("testingDropdown")));

dropdown.selectByVisibleText("Database Testing");

Thread.sleep(**50000**);

//Close the browser

driver.close();

}

}

## Press Enter/Return Key in Selenium

For pressing Enter key over a textbox we can pass Keys.ENTER or Keys.RETURN to the sendKeys method for that textbox.

WebElement textbox = driver.findElement(By.id("idOfElement"));

textbox.sendKeys(Keys.ENTER);

or

WebElement textbox = driver.findElement(By.id("idOfElement"));

textbox.sendKeys(Keys.RETURN);

Similarly, we can use Keys enum for different non-text keys and pass them to the sendKeys method. The following table has an entry for each of the non-text key present in a keyboard.

|  |  |
| --- | --- |
| **Keyboard's Key** | **Keys enum's value** |
| Arrow Key - Down | Keys.ARROW\_DOWN |
| Arrow Key - Up | Keys.ARROW\_LEFT |
| Arrow Key - Left | Keys.ARROW\_RIGHT |
| Arrow Key - Right | Keys.ARROW\_UP |
| Backspace | Keys.BACK\_SPACE |
| Ctrl Key | Keys.CONTROL |
| Alt key | Keys.ALT |
| DELETE | Keys.DELETE |
| Enter Key | Keys.ENTER |
| Shift Key | Keys.SHIFT |
| Spacebar | Keys.SPACE |
| Tab Key | Keys.TAB |
| Equals Key | Keys.EQUALS |
| Esc Key | Keys.ESCAPE |
| Home Key | Keys.HOME |
| Insert Key | Keys.INSERT |
| PgUp Key | Keys.PAGE\_UP |
| PgDn Key | Keys.PAGE\_DOWN |
| Function Key F1 | Keys.F1 |
| Function Key F2 | Keys.F2 |
| Function Key F3 | Keys.F3 |
| Function Key F4 | Keys.F4 |
| Function Key F5 | Keys.F5 |
| Function Key F6 | Keys.F6 |
| Function Key F7 | Keys.F7 |
| Function Key F8 | Keys.F8 |
| Function Key F9 | Keys.F9 |
| Function Key F10 | Keys.F10 |
| Function Key F11 | Keys.F11 |
| Function Key F12 | Keys.F12 |