Locate Elements by Link Text & Partial Link Text in Selenium Webdriver

**What is Link Text in Selenium?**

A **Link Text in Selenium** is used to identify the hyperlinks on a web page. It is determined with the help of an anchor tag. For creating the hyperlinks on a web page, we can use an anchor tag followed by the link Text.

### Links Matching a Criterion

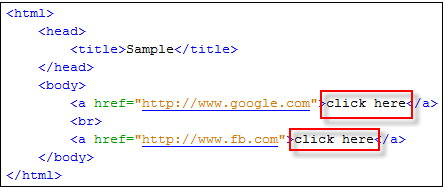
Links can be accessed using an exact or partial match of their link text. The examples below provide scenarios where multiple matches would exist and would explain how WebDriver would deal with them.

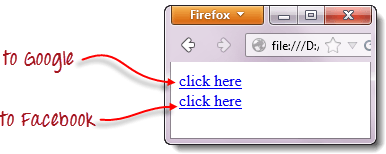
In this tutorial, we will learn the available methods to find and access the Links using Webdriver. Also, we will discuss some of the common problems faced while accessing Links and will further discuss on how to resolve them.

* [Accessing links using Exact Text Match: By.linkText()](https://www.guru99.com/locate-by-link-text-partial-link-text.html#2)
* [Accessing links using Partial Text Match: By.partialLinkText()](https://www.guru99.com/locate-by-link-text-partial-link-text.html#3)
* [How to get Multiple links with the same Link Text](https://www.guru99.com/locate-by-link-text-partial-link-text.html#5)
* [Case-sensitivity for Link Text](https://www.guru99.com/locate-by-link-text-partial-link-text.html#4)
* [Links Outside and Inside a Block](https://www.guru99.com/locate-by-link-text-partial-link-text.html#6)

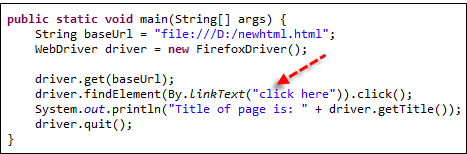
## Accessing links using Exact Text Match: By.linkText()

**Accessing links using their exact link text is done through the By.linkText() method**. However, if there are two links that have the very same link text, this method will only access the first one



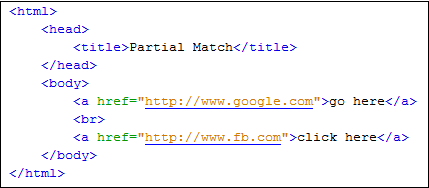


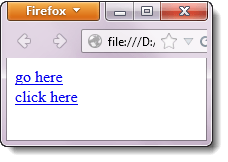
When you try to run the WebDriver code below, you will be accessing the first “click here” link



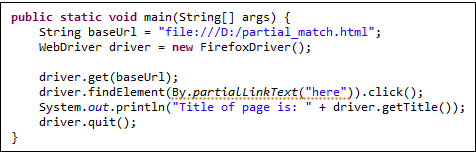
## Accessing links using Partial Text Match: By.partialLinkText()

Accessing links using a portion of their link text is done using the **By.partialLinkText()**method. If you specify a partial link text that has multiple matches, only the first match will be accessed. Consider the HTML code below.





When you execute the WebDriver code below, you will still be taken to Google.



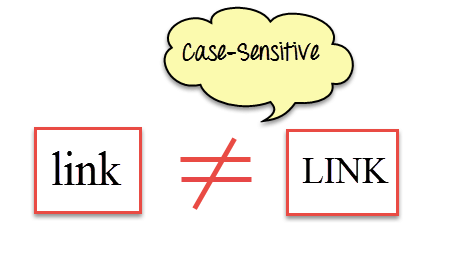
## How to get Multiple links with the same Link Text

So, how to get around the above problem? In cases where there are multiple links with the same link text, and we want to access the links other than the first one, how do we go about it?

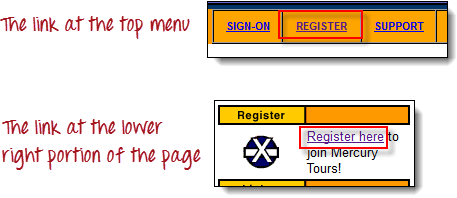
In such cases, generally, different locators viz… By.xpath(), By.cssSelector() or By.tagName() are used.

Most commonly used is By.xpath(). It is the most reliable one but it looks complex and non-readable too.

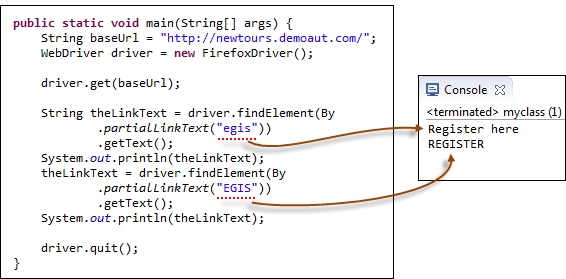
## Case-sensitivity for Link Text



The parameters for **By.linkText()** and **By.partialLinkText()**are both case-sensitive, meaning that capitalization matters. For example, in Mercury Tours’ homepage, there are two links that contain the text “egis” – one is the “REGISTER” link found at the top menu, and the other is the “Register here” link found at the lower right portion of the page.

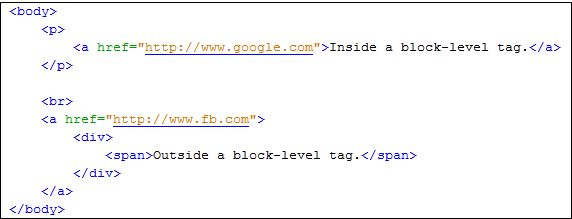


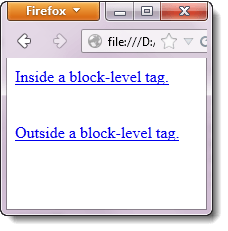
Though both links contain the character sequence “egis,” one is the “By.partialLinkText()” method will access these two links separately depending on the capitalization of the characters. See the sample code below.



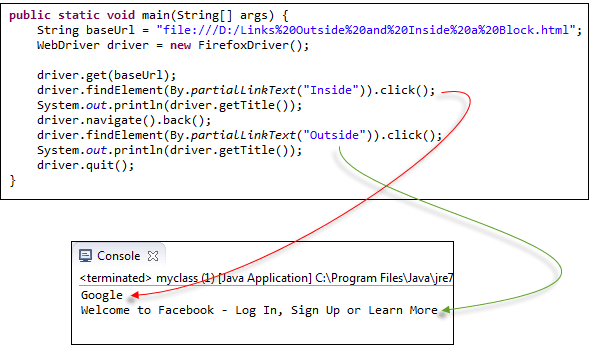
## Links Outside and Inside a Block

The latest HTML5 standard allows the <a> tags to be placed inside and outside of block-level tags like <div>, <p>, or <h3>. The “By.linkText()” and “By.partialLinkText()” methods can access a link located outside and inside these block-level elements. Consider the HTML code below.





The WebDriver code below accesses both of these links using By.partialLinkText() method.



# **Mouse Click & Keyboard Event: Action Class in Selenium Webdriver**

## Action Class in Selenium

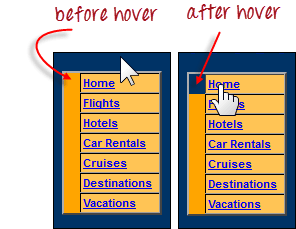
**Action Class in Selenium** is a built-in feature provided by the selenium for handling keyboard and mouse events. It includes various operations such as multiple events clicking by control key, drag and drop events and many more. These operations from the action class are performed using the advanced user

**Handling Keyboard & Mouse Events**

Handling special keyboard and mouse events are done using the **Advanced User Interactions API**. It contains the **Actions** and the **Action** classes that are needed when executing these events. The following are the most commonly used keyboard and mouse events provided by the Actions class.

|  |  |
| --- | --- |
| **Method** | **Description** |
| **clickAndHold()** | Clicks (without releasing) at the current mouse location. |
| **contextClick()** | Performs a context-click at the current mouse location. (Right Click Mouse Action) |
| **doubleClick()** | Performs a double-click at the current mouse location. |
| **dragAndDrop(source, target)** | Performs click-and-hold at the location of the source element, moves to the location of the target element, then releases the mouse.  **Parameters:**  source- element to emulate button down at.  target- element to move to and release the mouse at. |
| **dragAndDropBy(source, x-offset, y-offset)** | Performs click-and-hold at the location of the source element, moves by a given offset, then releases the mouse.  **Parameters**:  source- element to emulate button down at.  xOffset- horizontal move offset.  yOffset- vertical move offset. |
| **keyDown(modifier\_key)** | Performs a modifier key press. Does not release the modifier key – subsequent interactions may assume it’s kept pressed.  **Parameters**:  modifier\_key – any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL) |
| **keyUp(modifier \_key)** | Performs a key release.  **Parameters**:  modifier\_key – any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL) |
| **moveByOffset(x-offset, y-offset)** | Moves the mouse from its current position (or 0,0) by the given offset.  **Parameters**:  x-offset- horizontal offset. A negative value means moving the mouse left.  y-offset- vertical offset. A negative value means moving the mouse down. |
| **moveToElement(toElement)** | Moves the mouse to the middle of the element.  **Parameters**:  toElement- element to move to. |
| **release()** | Releases the depressed left mouse button at the current mouse location |
| **sendKeys(onElement, charsequence)** | Sends a series of keystrokes onto the element.  **Parameters**:  onElement – element that will receive the keystrokes, usually a text field  charsequence – any string value representing the sequence of keystrokes to be sent |

In the following example, we shall use the moveToElement() method to mouse-over on one Mercury Tours’ table rows. See the example below.



The cell shown above is a portion of a <TR> element. If not hovered, its color is #FFC455 (orange). After hovering, the cell’s color becomes transparent. It becomes the same color as the blue background of the whole orange table.

**Step 1:**Import the **Actions** and **Action** classes.

Keyboard & Mouse Event using Action Class in Selenium Webdriver

**Step 2:**Instantiate a new Actions object.

Keyboard & Mouse Event using Action Class in Selenium Webdriver

**Step 3:**Instantiate an Action using the Actions object in step 2.

Keyboard & Mouse Event using Action Class in Selenium Webdriver

In this case, we are going to use the moveToElement() method because we are simply going to mouse-over the “Home” link. The build() method is always the final method used so that all the listed actions will be compiled into a single step.

**Step 4:**Use the perform() method when executing the Action object we designed in Step 3.

Keyboard & Mouse Event using Action Class in Selenium Webdriver

Below is the whole WebDriver code to check the background color of the <TR> element before and after the mouse-over.

package newproject;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Action;

import org.openqa.selenium.interactions.Actions;

public class PG7 {

public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/newtours/";

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

WebElement link\_Home = driver.findElement(By.linkText("Home"));

WebElement td\_Home = driver

.findElement(By

.xpath("//html/body/div"

+ "/table/tbody/tr/td"

+ "/table/tbody/tr/td"

+ "/table/tbody/tr/td"

+ "/table/tbody/tr"));

Actions builder = new Actions(driver);

Action mouseOverHome = builder

.moveToElement(link\_Home)

.build();

String bgColor = td\_Home.getCssValue("background-color");

System.out.println("Before hover: " + bgColor);

mouseOverHome.perform();

bgColor = td\_Home.getCssValue("background-color");

System.out.println("After hover: " + bgColor);

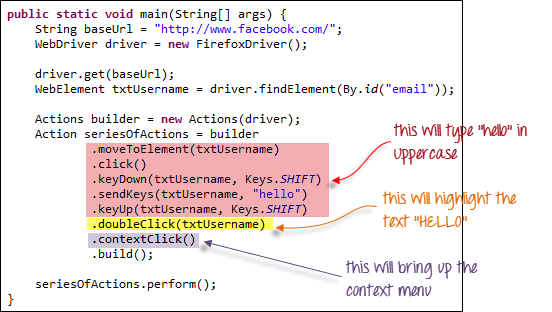
driver.close();

}

}

## Building a Series of Multiple Actions

**You can build a series of actions using the Action and Actions classes**. Just remember to close the series with the build() method. Consider the sample code below.



public static void main(String[] args) {

String baseUrl = "http://www.facebook.com/";

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

WebElement txtUsername = driver.findElement(By.id("email"));

Actions builder = new Actions(driver);

Action seriesOfActions = builder

.moveToElement(txtUsername)

.click()

.keyDown(txtUsername, Keys.SHIFT)

.sendKeys(txtUsername, "hello")

.keyUp(txtUsername, Keys.SHIFT)

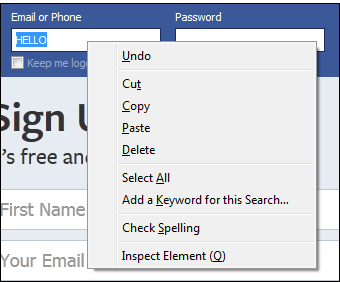
.doubleClick(txtUsername)

.contextClick()

.build();

seriesOfActions.perform() ;

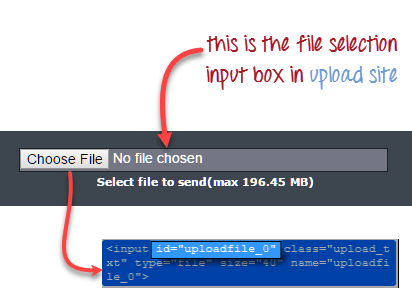
}



# How to Upload a File using Selenium Webdriver

**Uploading Files**

**Uploading files in WebDriver is done by simply using the sendKeys() method on the file-select input field to enter the path to the file to be uploaded.**

Handle File upload popup in Selenium Webdriver

handle file upload popup in selenium webdriver

package newproject;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

public class PG9 {

public static void main(String[] args) {

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

String baseUrl = "http://demo.guru99.com/test/upload/";

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

WebElement uploadElement = driver.findElement(By.id("uploadfile\_0"));

// enter the file path onto the file-selection input field

uploadElement.sendKeys("C:\\newhtml.html");

// check the "I accept the terms of service" check box

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

// click the "UploadFile" button

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

}

}