**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.

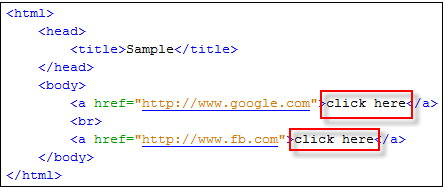
Here is what you will learn-

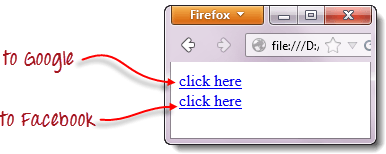
* [Accessing links using Exact Text Match: By.linkText()](https://www.guru99.com/locate-by-link-text-partial-link-text.html#2)
* [Accessing links using Partial Text Match: By.partialLinkText()](https://www.guru99.com/locate-by-link-text-partial-link-text.html#3)
* [How to get Multiple links with the 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()**

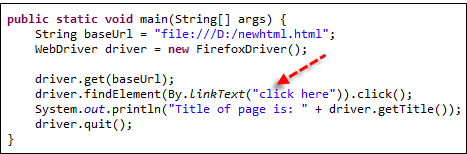
**Accessing links using their exact link text is done through the By.linkText() method**. However, if there are two links that have the very same link text, this method will only access the first one. Consider the HTML code below

Learn Java Programming with Beginners Tutorial





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



**Code:**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class MyClass {

public static void main(String[] args) {

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

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get(baseUrl);

driver.findElement(By.linkText("click here")).click();

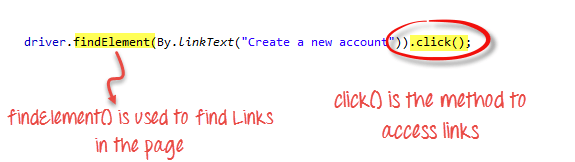
System.out.println("title of page is: " + driver.getTitle());

driver.quit();

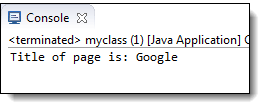
}

}

Here is how it works-

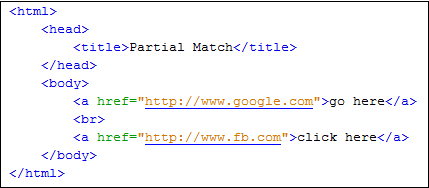


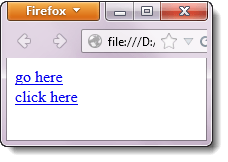
As a result, you will automatically be taken to Google.



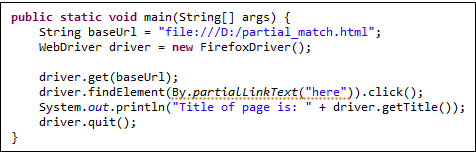
**Accessing links using Partial Text Match: By.partialLinkText()**

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.



**Code:**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class P1 {

public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/accessing-link.html";

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get(baseUrl);

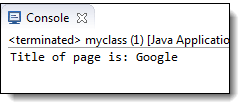
driver.findElement(By.partialLinkText("here")).click();

System.out.println("Title of page is: " + driver.getTitle());

driver.quit();

}

}



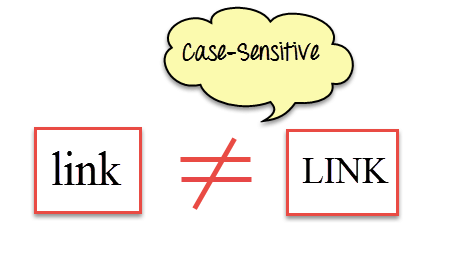
**How to get Multiple links with the same Link Text**

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?

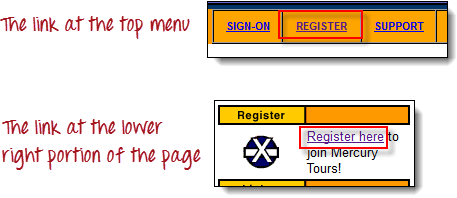
In such cases, generally, different locators viz… By.xpath(), By.cssSelector() or By.tagName() are used.

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

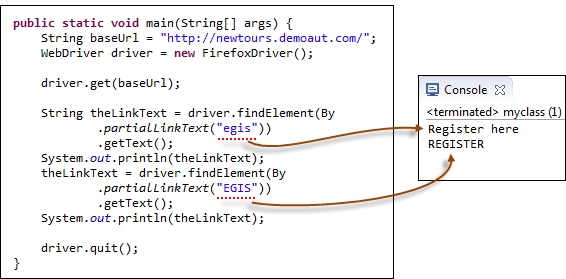
**Case-sensitivity for Link Text**



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.



**Code**

public static void main(String[] args) {

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

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get(baseUrl);

String theLinkText = driver.findElement(By

.partialLinkText("egis"))

.getText();

System.out.println(theLinkText);

theLinkText = driver.findElement(By

.partialLinkText("EGIS"))

.getText();

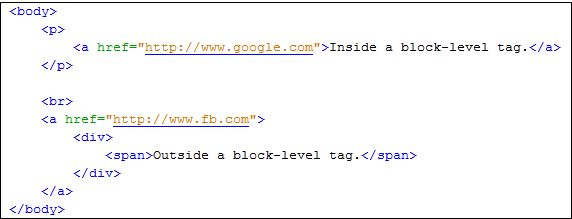
System.out.println(theLinkText);

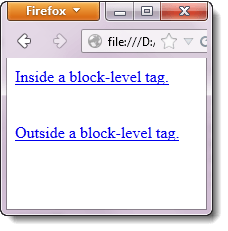
driver.quit();

}

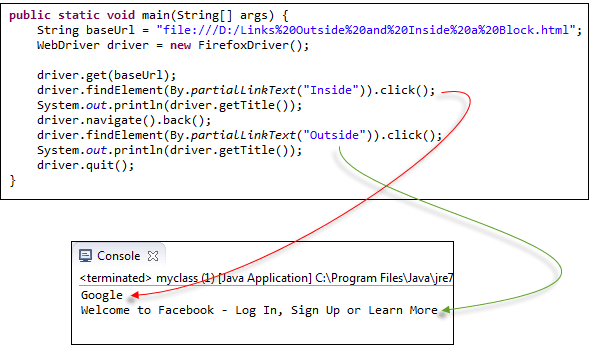
**Links Outside and Inside a Block**

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.



**Code:**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class MyClass {

public static void main(String[] args) {

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

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get(baseUrl);

driver.findElement(By.partialLinkText("Inside")).click();

System.out.println(driver.getTitle());

driver.navigate().back();

driver.findElement(By.partialLinkText("Outside")).click();

System.out.println(driver.getTitle());

driver.quit();

}

}