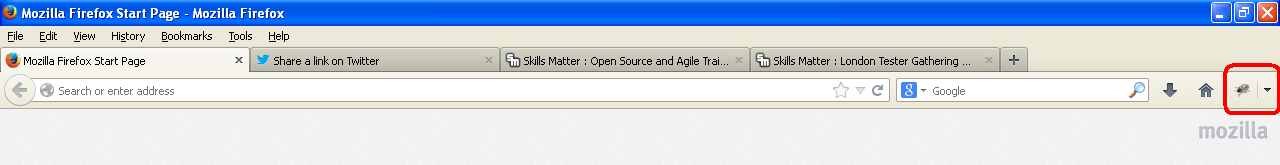
Locators allow us to find elements on a page that can be used in the tests. There are various locating techniques to identify the elements on a webpage. The object/element identification should be based on element identifiers, unique attribute values or other structural information that is stable enough to withstand frequent changes to the web application.

To locate an element based on associated HTML/Css we can use firebug : Firefox addon (in Firefox browser) or Internet explorer developer tools (In IE). In this document, we describe the firebug installation and usage. All the screen shots in the examples are using firebug add on.

## Firebug Installation

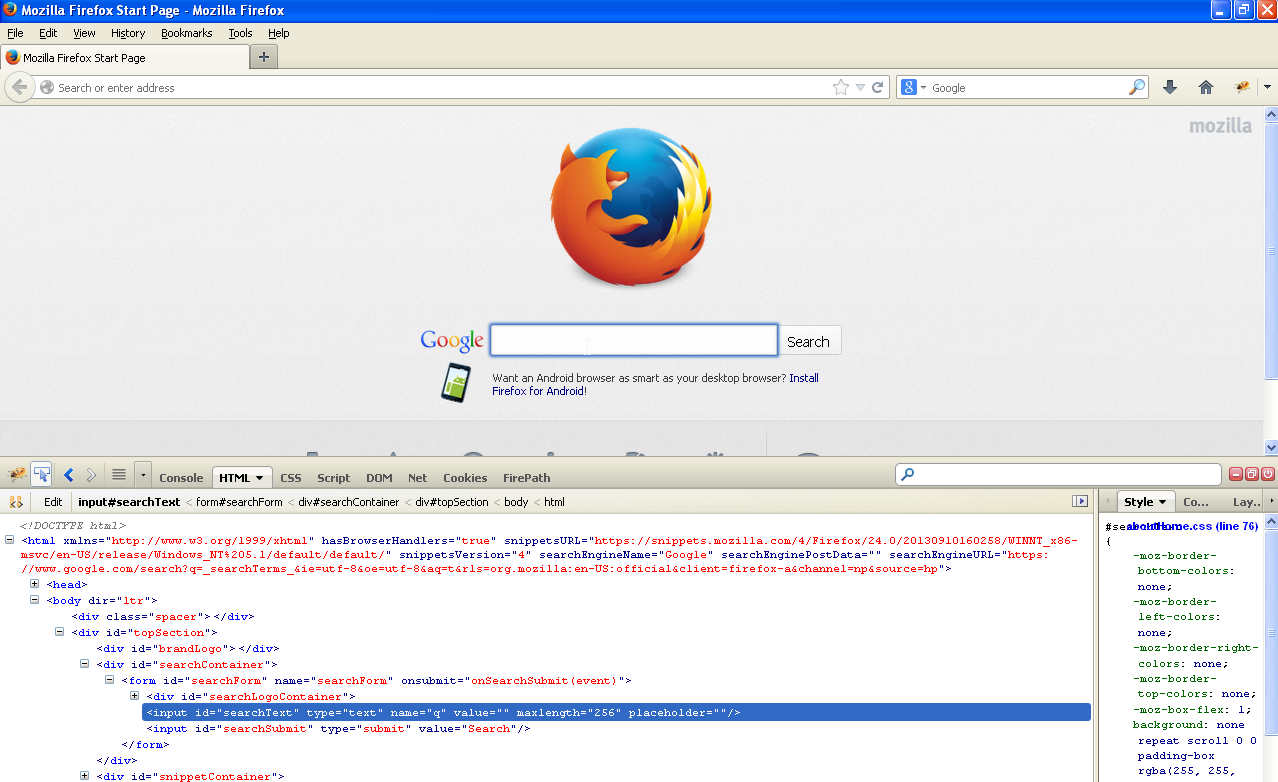
**Prerequisite:** Firefox should be installed.

Open Firefox browser. In the menu bar, go to Tools=> Add-ons. Search for Firebug. Install it. After installation restart the browser and we can see a gray 'Firebug' in the Firefox Navigation toolbar (along the top of the window). Click on the Bug to start. Firebug will open in the bottom of your web page displaying the webpage HTML in the html panel.



To inspect elements, open the desired webpage in the firefox browser. In Firebug, Click on the arrow  ‘Click on element in the page to inspect’ icon. Navigate to the desired object and click on that object. In the html panel it displays the HTML tag information of the selected object.

In the below example we can see the HTML tag information of the Google search box



We can interpret the Google Search text box as: a text box with ‘input’ tag/element with attributes id=”searchText”, type=”text”, name=”q” , present under the div tag with attribute id="searchLogoContainer". Hence, we can locate the Search textbox element with the attributes id or name

Eg: id= searchText or name=q

So, on a webpage we can locate elements with their unique attributes. If there are many elements with same attributes we can use a combination of attributes or select them with their parents and tags.

## Thomas HTML editor

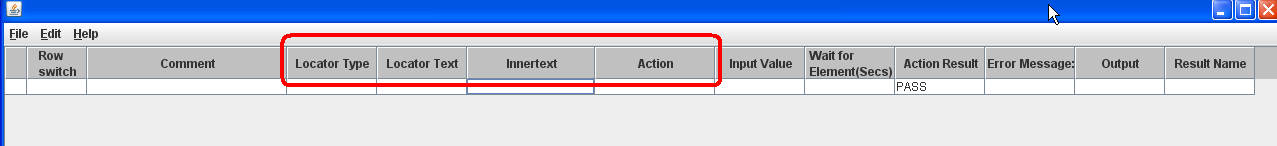
In the HTML editor we use the below columns for locating an element to perform the desired action

**Locator type**: to specify the type of element locator (id, name, class, partiallinktext, xpath, css)

**Locator text**: the description/value of the locator type

**Innertext**: text in addition to the locator description to identify the element uniquely

**Action**: the action we wish to perform on the element



## Locating Techniques

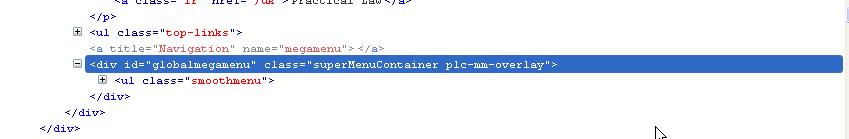
There are many locator techniques depending on the element attributes. When the elements have attributes like- Id/Name/Class /Text it is preferable to use those attributes. If they are not unique enough to identify the elements then we can use xpath or css. Here are some of the locator techniques with examples

1. **By id** – The id attribute can be used to uniquely identify an element within a HTML document

Eg1: <div id="moduleName"> test </div> --- id=moduleName

Eg2: Check you are on the WSR site we can use the id --- id = globalmegamenu

In the HTML editor it should be – Locator Type=id, Locator Text= globalmegamenu, Action=exists



1. **By Name** - The name attribute can be used to identify an element within a HTML document

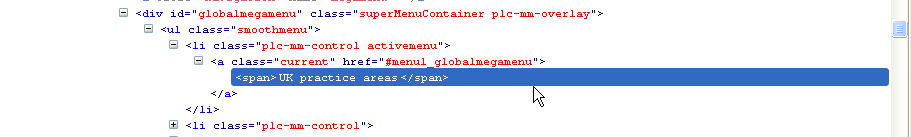
Eg: <a name="topOfPage"></a> -- name=topOfPage

In the HTML editor it should be – Locator Type= name, Locator Text= topOfPage

1. **By classname** – The class name can be used to identify an element within a HTML document

Eg: to click UK practice areas’ link with class name we can target it as Classname= current

In the HTML editor it should be – Locator Type= Classname, Locator Text= current, Action=Click



1. **By PartialLinktext** - This is a simple method of locating a hyperlink in your web page by using the fulltext/partial text of the link.

If two links with the same text are present, then the first match will be used

Eg1: On the header page, identify ‘UK practice areas’ link

‘Partiallinktext= UK practice areas’ or ‘Partiallinktext=UK practice’

To click on different practice areas in the menu we can use partiallinktext=Tax

In the HTML editor it should be – Locator Type= partiallinktext, Locator Text= Tax, Action=click

1. **By Xpath** - Can be used to identify objects that cannot be identified by the above ways like elements in tables, elements without id’s and class
2. **By CSS** – Alternative to XPath to identify elements.

## XPath

**What is XPath?**

XPath is a syntax for defining parts of an XML document. XPath uses path expressions to navigate and select nodes or node-sets in an XML document.

**XPath Terminology**  
<?xml version="1.0" encoding="ISO-8859-1"?>  
<bookstore>   
  <book>  
    <title **lang="en"**>Harry Potter</title>   
    <author>J K. Rowling</author>   
    <year>2005</year>  
    <price>29.99</price>  
  </book>  
</bookstore>

**Atomic Values**

Element Node

**Attribute node**

Root Element Node

**Relationship of Nodes**

1. **Parent** - Each element and attribute has one parent.

Eg: The book element is the parent of the title, author, year, and price

1. **Children -** Element nodes may have zero, one or more children

Eg: The title, author, year, and price elements are all children of the book element

1. **Siblings** - Nodes that have the same parent

Eg: The title, author, year, and price elements are all siblings

1. **Ancestors -** A node's parent, parent's parent, etc

Eg: The ancestors of the title element are the book element and the bookstore element

1. **Descendants** - A node's children, children's children, etc

Eg: Descendants of the bookstore element are the book, title, author, year, and price elements

**Selecting Nodes (basics)**

* / - Selects from root node - If the path starts with a slash ( / ) it always represents an absolute path to an element!

Eg: /bookstore - Selects the root element bookstore

* // - Selects nodes in the document from the current node that match the selection no matter where they are

Eg: //book - Selects all book elements no matter where they are in the document

* @ - selects attributes

Eg: //@lang - Selects all attributes that are named lang

We can select the unknown nodes with the help of wild card characters- \*( Matches any element node),@\*( Matches any attribute node)

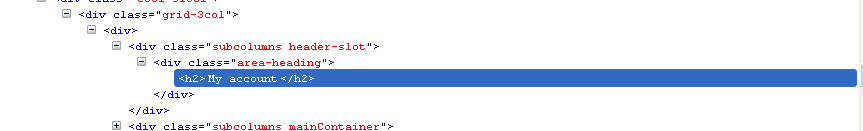
|  |  |
| --- | --- |
|  |  |

1. Element <E> by relative reference - **//E**

Eg1: xpath - //author - Selects all author elements no matter where they are in the document

Eg2: To verify 'My account' is displayed on the page – xpath - //h2 .

In the HTML editor it should be – Locator Type= xpath, Locator Text= //h2, Innertext= My account, Action=exists

When the elements are repetitive and cannot uniquely be identified with just relative reference. Hence we can add inner text property ( as in the above example) or add unique attributes as below

1. Element <E> containing text 't'- **//E[contains(text(),'t')]**

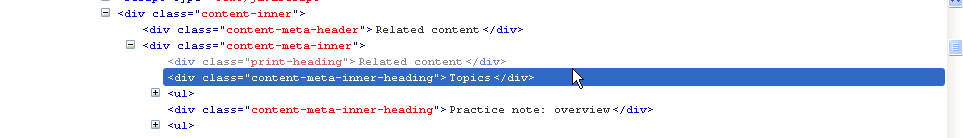
Eg1: On the Search Results page, in the lefthand filter tree, to click on the Resource Type link we can use xpath= //a[contains(text(),'Resource Type')]

In the HTML editor it should be – Locator Type= xpath, Locator Text= //a[contains(text(),'Resource Type')], Action=click



Eg2: On the topic page <http://uk.d100del.dev.practicallaw.com/2-107-3725> Verify that the Related content box is displayed --- Xpath= //div[text()="Topics"]

In HTML Editor, Locator Type= xpath, Locator Text= //div[text()="Topics"], Action=exists



1. Element <E> with attribute A - **//E[@A]**

Eg: xpath - //title[@lang] - Selects all the title elements that have an attribute named lang

In the HTML editor it should be – Locator Type= xpath, Locator Text= //title[@lang]

We can make it more specific by providing the value.

If you need to specify more than one attribute property you do this by just listing them as follows:

//E[@a1 = xxx][@a2=yyy].. . etc

For example:

//div[@style='display:none'][@class='mainPage']

1. Element <E> with attribute A containing text 't' exactly - **//E[@A='t']**

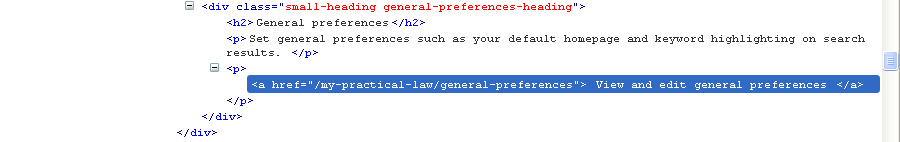
Eg1 //title[@lang='eng'] - Selects all the title elements that have an attribute named lang with a value of 'eng'

Eg2: To indentify General Preferences link --

xpath - //a[@href='/my-practical-law/general-preferences']

In the HTML editor it should be –

Locator Type= xpath, Locator Text= //a[@href='/my-practical-law/general-preferences']

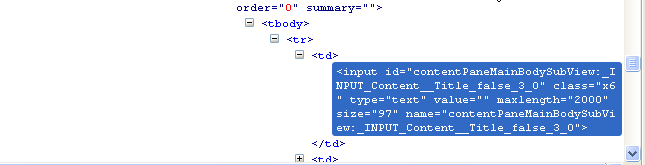


1. If we want to check an element that contains some Text (not an exact match of the whole string)

Element <E> with attribute A containing text 't'- **//E[contains(@A,'t')]**

Eg: In dash, to identify the title textbox while creating a new article

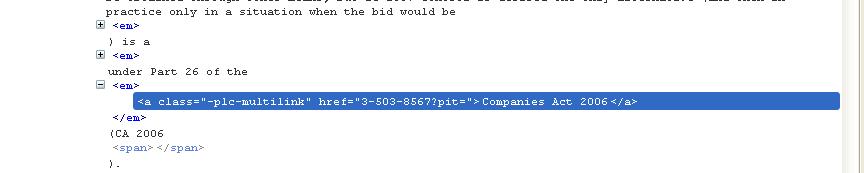
Locator Type= xpath, Locator Text= //input[contains(@id,'Title')]



1. Element <E> whose attribute A begins with 't'- **//E[starts-with(@A, 't')]**

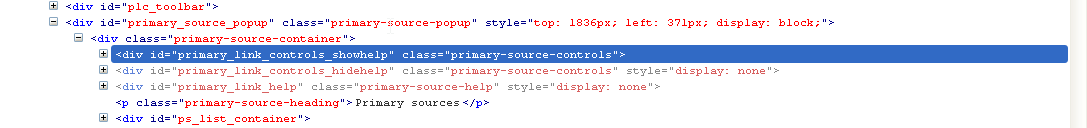
Eg: In the link <http://uk.d100del.dev.practicallaw.com/6-107-3605> to click on [*Companies Act 2006*](http://uk.d100del.dev.practicallaw.com/3-503-8567?pit=) we can use

Locator Type= xpath, Locator Text= //a[starts-with(@href,'3-503-8567')], Action=click



And to click on the primary source popup appears

Locator Type= xpath, Locator Text= //div[starts-with(@id,'primary\_link\_controls\_showhelp')], Action=exists

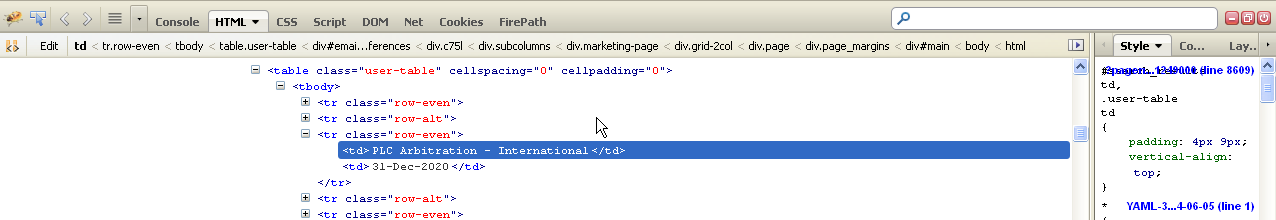


1. **And, or** : Element <E1> with text T1 and T2 , Element <E1> with text T1 or T2

We can use the and/ or attributes to locate the elements

E.g. for below locator

Locator Type= xpath, Locator Text= //tr[(td[1] = 'PLC Arbitration - International') and (td[2] = '31-Dec-2020')] , Action=exists

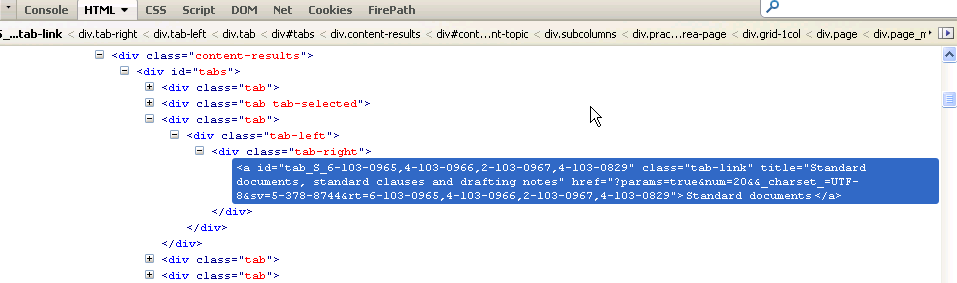


1. **Nth element of a type** , First <E> child, //E[1] , Second <E> child, //E[2] i.e //E[n]

If we want to interact with the nth element then we use E[n]. Note that it works only in case of siblings of an element.

Eg: To check that the third tab on a resource page is standard documents we can use

Locator Type= xpath, Locator Text= //div[@id='tabs']/\*[3], Action= containstext, Input Value= Standard documents



|  |
| --- |
|  |

1. **Xpath axis** : This is an extreme case where we want to find a table cell that had specific text, then traverse the tree backward to find/perform any other operation we can use xpath axis

Following is the listof axis that we can use in our Xpath queries to find elements on the page

**Axis Name** **Result**

ancestor Selects all ancestors (parents, grandparents etc) of the element

descendant Selects all the descendants(children, grandchildren etc) of the element

following Selects all the elements that following the closing tab of the current element

following-sibling Selects all the siblings after the current element

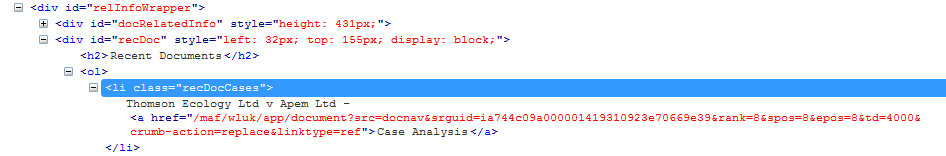
parent Selects the parent of the current element

preceding Selects all the elements that are before the current element

preceding-Sibling Selects all of the siblings before the current element

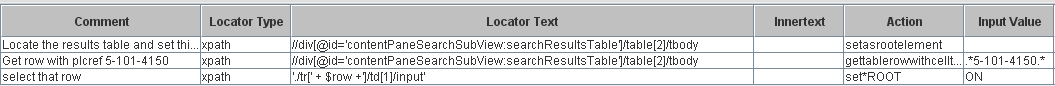
Eg : To get the text of an element with rank=8 property we can give

Locator Type= xpath, Locator Text= //a[contains(@href,'rank=8')]/parent::li, Action= gettext



1. **Table processing :** This is an extreme case where we have to perform any action on a Table.

Eg: In dash when we search for a plc reference and select that row we can use the below stuff





,

## CSS

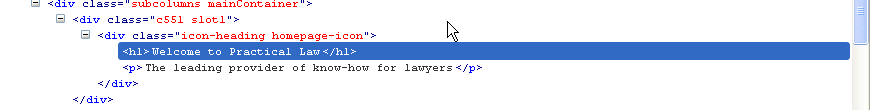
Another way to identify the objects in a webpage is by css.

1. Element selector : Element selectors use the name of the element to locate the element.

Element <E> by relative reference - **css=E**

Eg: To verify the PLC main page is loaded , we can wait for the header as below

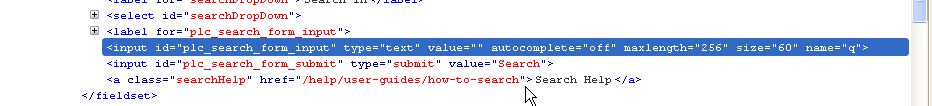
Locator Type= css, Locator Text= h1, InnerText= Welcome to Practical Law ,Action=exists



1. By id: Element <E> with id I : **css=#id**

Eg: Enter the search term in the search box css= #plc\_search\_form\_input. So in the HTML editor

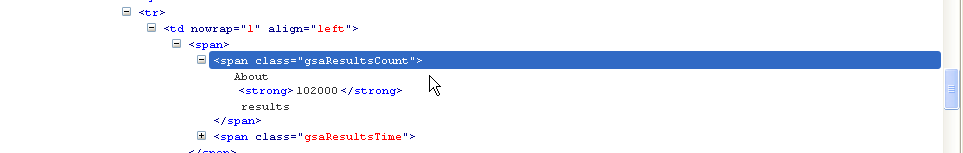
Locator Type= css, Locator Text= #plc\_search\_form\_input,Action=set, Input Value=Takeovers



1. By class : Element with a class C : **css=.C**

Eg: To wait for the Search results page to load we can use the below in the HTML editor

Locator Type= css, Locator Text= .gsaResultsCount,Action=containsText, Input Value=About

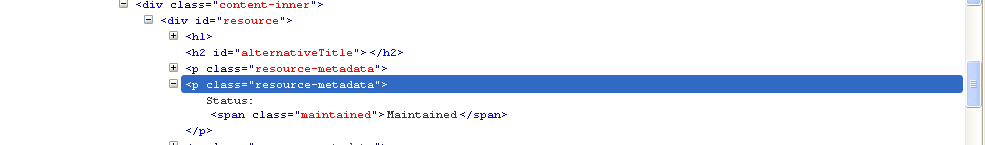


1. **Descendent** : This type of selectors are using to access elements located with other elements when the locator can hit multiple matches. A **space** is needed between each selector leading to the target from the ancestor to the descendant going from left to right. we need to be more specific by using descendant selectors.

Eg1: to locate a heading h1 present under 2 <div> tags, we can use css=div1.class1 div2.class2 h1

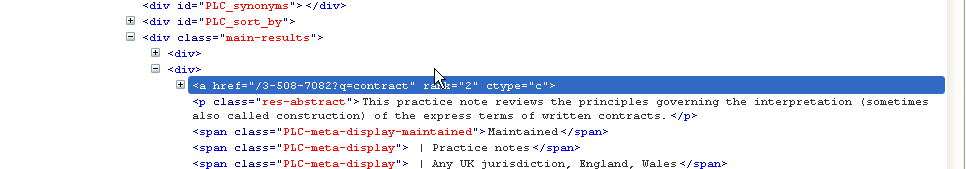
Eg2: On the resource page, to verify the Status : Maintained , we can use the below in the HTML editor

Locator Type= css, Locator Text= #resource.resource-metadata span.maintained, Action=containsText, Input Value=Maintained



Eg3 : On the search results page to click on the second result link we can use

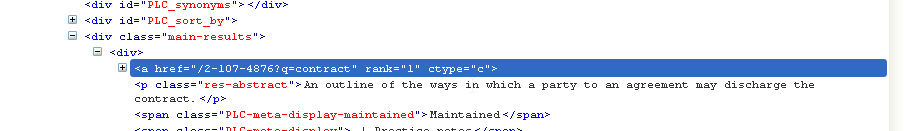
Locator Type= css, Locator Text= div.main-results div a[rank=2], Action=click



1. **CHILD SELECTORS** OR **DIRECT DESCENDANTS :** To target an immediate child, you need to place the greater than sign > between the parent and the immediate child : **parent > child**

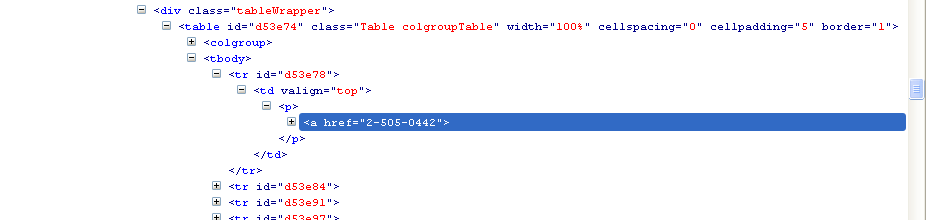
Eg1: On the Search results page if we want to click on the first search result we can say

Locator Type= css, Locator Text= div.main-results > div > a, Action=click



Eg2: On the page <http://uk.d100del.dev.practicallaw.com/4-384-3897?q=takeovers> if you want to click on the ‘Introduction to the Code’ in the table Index of notes we can use

Locator Type= css, Locator Text= #d53e78 >td > p > a, Action=click



1. Targeting elements that contain specific TEXT using **contains :** If we wanted to target the element that contains a specific string we can use contains

Eg: To verify that a page is loaded we can wait for the header to be loaded. So we can use

Locator Type= css, Locator Text= #resource>h1:contains(Takeover Code know-how), Action=exists



1. **Finding nth – child : css=E:nth-child(2)**

If we want to interact with the nth element then we use E[n]. Note that it works only in case of siblings of an element.

Eg: To check that the third tab on a resource page is standard documents we can use

Locator Type= css, Locator Text= div#tabs \*:nth-child(3), Action= containstext, Input Value= Standard documents

