

Selenium Locating Strategy

Installing Add-on

- ▶ Fire bug
- ▶ Fire path
- ▶ ChroPath

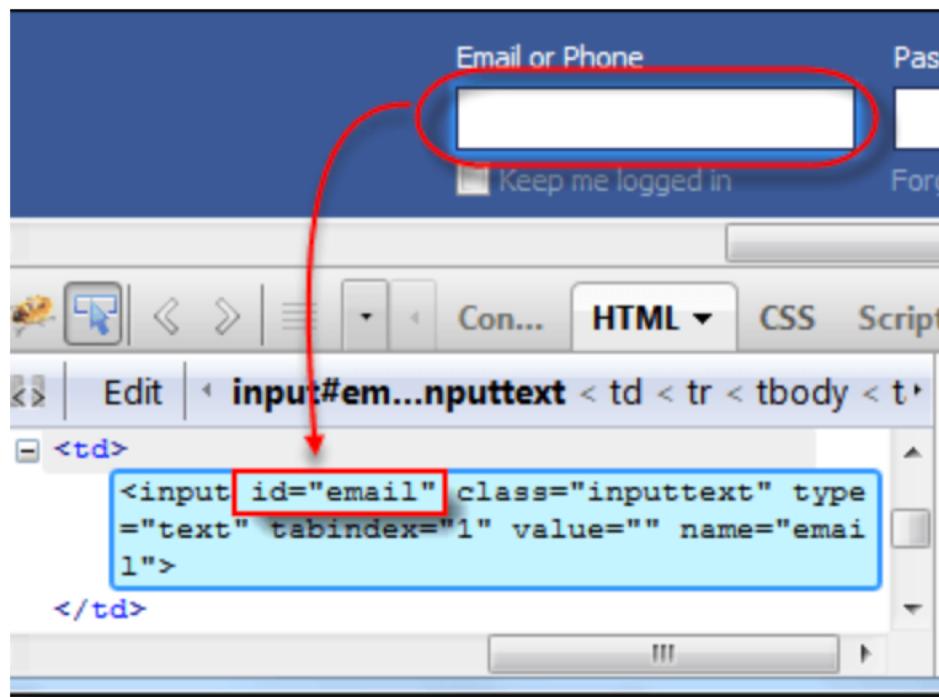
Type of Locators

- ▶ **ID Locator**
- ▶ **Name Locator**
- ▶ **Class name**
- ▶ **Link Locator**
- ▶ **Partial Link text**
- ▶ **Tag Name**
- ▶ **XPath Locator**
- ▶ **CSS**

- ▶ **id** Select element with the specified @*id* attribute.
- ▶ **Name** Select first element with the specified @*name* attribute.
- ▶ **Linktext** Select link (anchor tag) element which contains text matching the specified link text
- ▶ **Partial Linktext** Select link (anchor tag) element which contains text matching the specified partial link text
- ▶ **Tag Name** Locate Element using a Tag Name .
- ▶ **Class name** Locate Element using a class Name ..
- ▶ **Css** Select the element using css selectors. You can check here for refer [W3C CSS Locatros](#)
- ▶ **Xpath** Locate an element using an XPath expression.

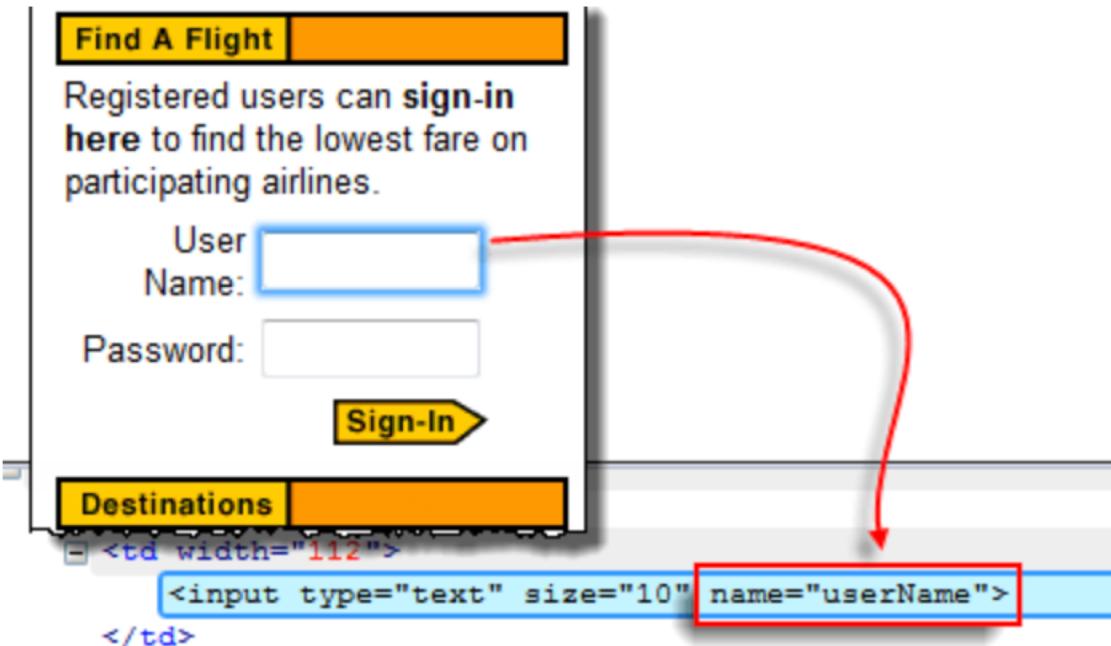
Locating by ID

- ▶ Target Format: *id=id of the element*



Locating by Name

- ▶ Target Format: name=*name of the element*



Locating by Link Text

Target Format: link=*link_text*

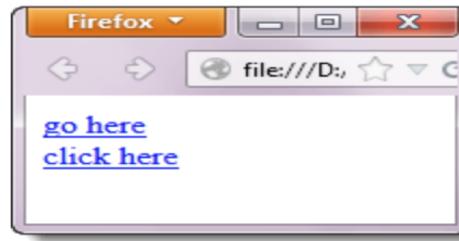


The screenshot shows a website header for "one cool summer ARUBA". Below the header is a horizontal menu bar with four items: "SIGN-ON", "REGISTER", "SUPPORT", and "CONTACT". The "REGISTER" button is highlighted with a red arrow. Below the menu, a code editor displays the HTML code for the menu items. The "REGISTER" link is highlighted with a blue box.

```
'mouseOver')" onmouseout="changeStyle(this, 'mouseOut')">
    <a href="mercuryregister.php">REGISTER </a>
</td>
+ <td class="mouseOut" width="73" height="
```

Locating by Partial Link Text

```
<html>
  <head>
    <title>Partial Match</title>
  </head>
  <body>
    <a href="http://www.google.com">go here</a>
    <br>
    <a href="http://www.fb.com">click here</a>
  </body>
</html>
```



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

```
public static void main(String[] args) {
    String baseUrl = "file:///D:/partial_match.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    driver.findElement(By.partialLinkText("here")).click();
    System.out.println("Title of page is: " + driver.getTitle());
    driver.quit();
}
```

Locating by XPath

- ▶ Syntax for Xpath : Xpath= //tagname[@attribute='value']

Xpath= //tagname[@Attribute='Value']

The diagram shows the XPath expression `Xpath= //tagname[@Attribute='Value']` enclosed in a black-bordered box. Five orange speech bubbles point to different parts of the expression:

- Select Current node (points to the first `//`)
- Selects Attribute (points to the `@Attribute`)
- Value of the attribute (points to the `'Value'`)
- Tagname like Input, Div, Img etc. (points to the `tagname`)
- Attribute Name (points to the `Attribute`)

Types of X-path

- ▶ There are two types of XPath:
 - 1) Absolute XPath .
 - 2) Relative XPath .

Absolute XPath :

- ▶ It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.
- ▶ The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node.
- ▶ **Absolute xpath:**
`html/body/div[1]/section/div[1]/div/div/div/div[1]/div/div/div/div/div[3]/div[1]/div/h4[1]/b`

Absolute xpath:

The screenshot shows a web page with two main sections: 'TESTING' and 'SAP'. The 'TESTING' section contains links for 'Learn Software Testing', 'QTP (Quick Test Professional)', 'Learn Selenium', and 'Learn Mobile App Testing'. The 'SAP' section contains links for 'Learn SAP Beginner', 'Learn SAP ABAP', 'Learn SAP HR/HCM', and 'Learn SAP FICO'. A speech bubble labeled 'Element' points to the 'TESTING' section. Another speech bubble labeled 'Absolute Path' points to the FirePath interface.

The FirePath toolbar at the top includes icons for Inspect, Console, HTML, CSS, Script, DOM, Net, Cookies, and FirePath. The FirePath dropdown menu is open, showing 'Top Window' and 'Highlight'. The 'XPath:' field contains the absolute XPath path: `html/body/div[1]/section/div[1]/div/div/div[1]/div/div/div/div[3]/div[1]/div/h4[1]/b`. This path is highlighted with a red box. Below the path, the FirePath tree displays the corresponding HTML structure:

```
<br/>


#### Testing


```

A blue bar highlights the text 'Testing' in the `Testing` node. At the bottom left of the FirePath interface, a red box highlights the text '1 matching node'.

Relative xpath:

- ▶ For Relative Xpath the path starts from the middle of the HTML DOM structure. It starts with the double forward slash (//), which means it can search the element anywhere at the webpage.
- ▶ You can start from the middle of the HTML DOM structure and no need to write long xpath.
- ▶ Relative xpath: `//*[@class='featured-box']//*[text()='Testing']`

Relative xpath:

The screenshot shows a web page with two main sections: 'TESTING' and 'SAP'. The 'TESTING' section contains links for 'Learn Software Testing', 'QTP (Quick Test Professional)', 'Learn Selenium', and 'Learn Mobile App Testing'. The 'SAP' section contains links for 'Learn SAP Beginner', 'Learn SAP ABAP', 'Learn SAP HR/HCM', and 'Learn SAP FICO'. A large orange speech bubble labeled 'Element' points to the 'TESTING' section. Another orange speech bubble labeled 'Relative Path' points to the FirePath toolbar. The FirePath toolbar includes icons for Top Window, Highlight, and XPath. The XPath field contains the expression `//*[@class='featured-box']//*[text()='Testing']`. Below the toolbar, a tree view shows the DOM structure with the matching node highlighted in blue. The message '1 matching node' is displayed at the bottom.

Element

TESTING

SAP

Learn Software Testing

QTP (Quick Test Professional)

Learn Selenium

Learn Mobile App Testing

Learn SAP Beginner

Learn SAP ABAP

Learn SAP HR/HCM

Learn SAP FICO

Live Tes

Live Sel

Live Ecc

Live UF

Top Window Highlight XPath: //*[@class='featured-box']//*[text()='Testing']

Relative Path

1 matching node

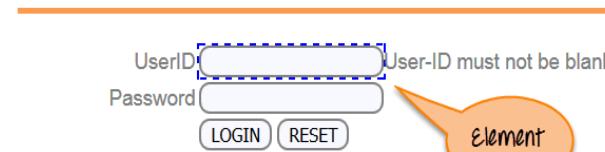
Using XPath Handling complex & Dynamic elements in Selenium

► What are XPath axes.

- XPath axes search different nodes in XML document from current context node. XPath Axes are the methods used to find dynamic elements, which otherwise not possible by normal XPath method having no ID , Classname, Name, etc.
- Axes methods are used to find those elements, which dynamically change on refresh or any other operations. There are few axes methods commonly used in Selenium Webdriver like child, parent, ancestor, sibling, preceding, self, etc.

1. Basic XPath:

- ▶ XPath expression select nodes or list of nodes on the basis of attributes like **ID**, **Name**, **Classname**, etc. from the XML document as illustrated below.
- ▶ Syntax: **Xpath=//input[@name='uid']**



Some more basic xpath expressions:

```
Xpath=//input[@type='text']
Xpath= //label[@id='message23']
Xpath= //input[@value='RESET']
Xpath=//*[@class='barone']
Xpath=//a[@href='http://demo.guru99.com/']
Xpath= //img[@src='//cdn.guru99.com/images/home/java.png']
```



2. Contains()

- ▶ Contains() is a method used in XPath expression. It is used when the value of any attribute changes dynamically, for example, login information.
- ▶ The contain feature has an ability to find the element with partial text as shown in below example.
- ▶ In this example, we tried to identify the element by just using partial text value of the attribute. In the below XPath expression partial value 'sub' is used in place of submit button. It can be observed that the element is found successfully.
- ▶ Complete value of 'name' is 'btnLogin' but using only partial value 'btn'.
 - ▶ `Xpath=//*[contains(@name,'btn')]`
- ▶ Complete value of 'Type' is 'submit' but using only partial value 'sub'.
 - ▶ `Xpath=//*[contains(@type,'sub')]`

► Xpath=//*[contains(@id,'message')]

The screenshot shows a browser developer tools window with the FirePath tab selected. The XPath query `//*[contains(@id,'message')]` is entered in the search bar and highlighted with a red box. The results pane displays the HTML structure of the page with two nodes matching the query, both of which are highlighted with blue boxes. A callout box on the left indicates "2 Nodes Matched". A green arrow points from the highlighted text in the screenshot to the highlighted text in the results pane.

User-ID User-ID must not be blank

Password Password must not be blank

LOGIN RESET

Top Window ▾ Highlight XPath: `//*[contains(@id,'message')]`

2 Nodes Matched

2 matching nodes

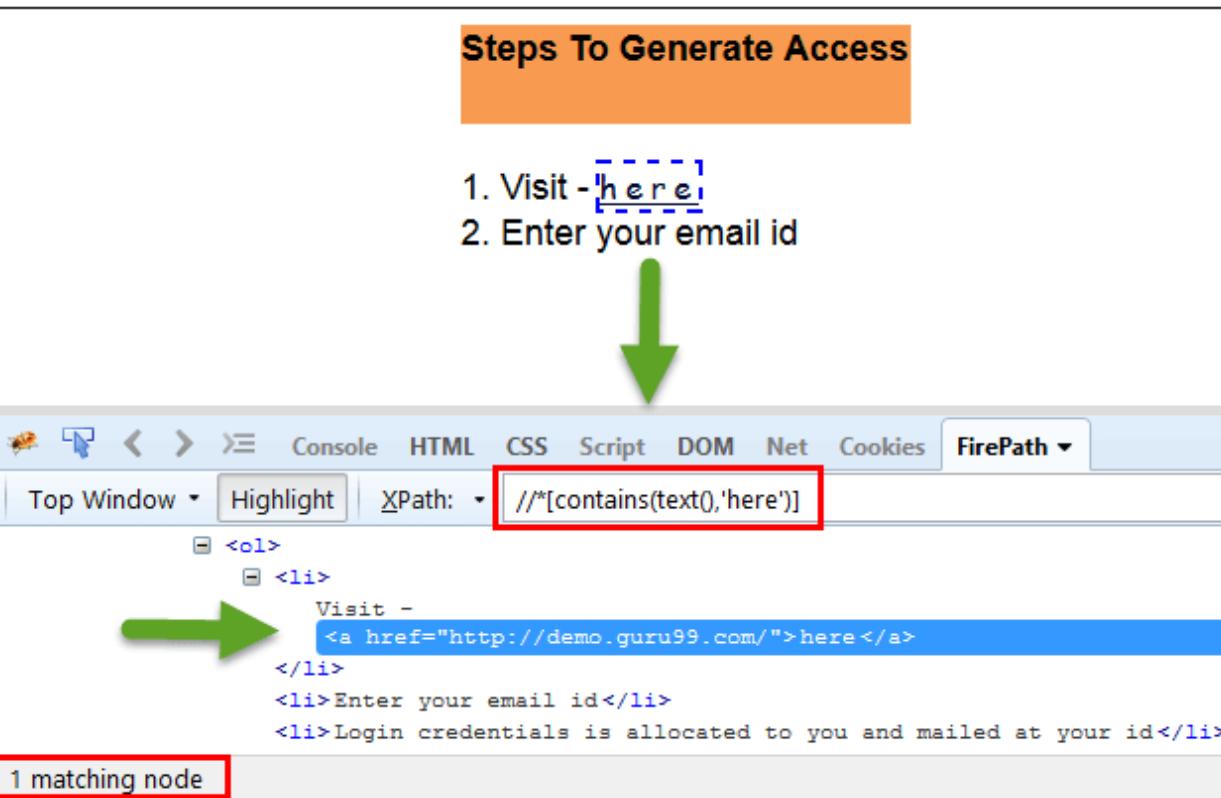
```
<td>
  <input type="text" onblur="validateuserid();" onkeyup="validateuserid();" maxlength="10" name="uid"/>
  <label id="message23" style="visibility: visible;">User-ID must not be blank</label>
</td>
<tr>
  <!-- Display Password and its text box-->
  <tr>
    <td align="right">Password</td>
    <td>
      <input type="password" onblur="validatepassword();" onkeyup="validatepassword();" name="password"/>
      <label id="message18" style="visibility: visible;">Password must not be blank</label>
    </td>
  </tr>
```

- In the below expression, we have taken the "text" of the link as an attribute and 'here' as a partial value as shown in the below screenshot. This will find the link ('here') as it displays the text 'here'.
 - Xpath=//*[contains(text(),'here')]
 - Xpath=//*[contains(@href,'guru99.com')]

Steps To Generate Access

1. Visit - [here](#)
2. Enter your email id

↓



The screenshot shows the FirePath extension for Firefox. The toolbar has tabs for Console, HTML, CSS, Script, DOM, Net, Cookies, and FirePath (selected). The XPath tab is active, displaying the query `//*[contains(text(),'here')]`. The results pane shows the HTML structure of the page with one matching node highlighted in blue. A green arrow points from the 'here' link in the steps above to the highlighted node in the results.

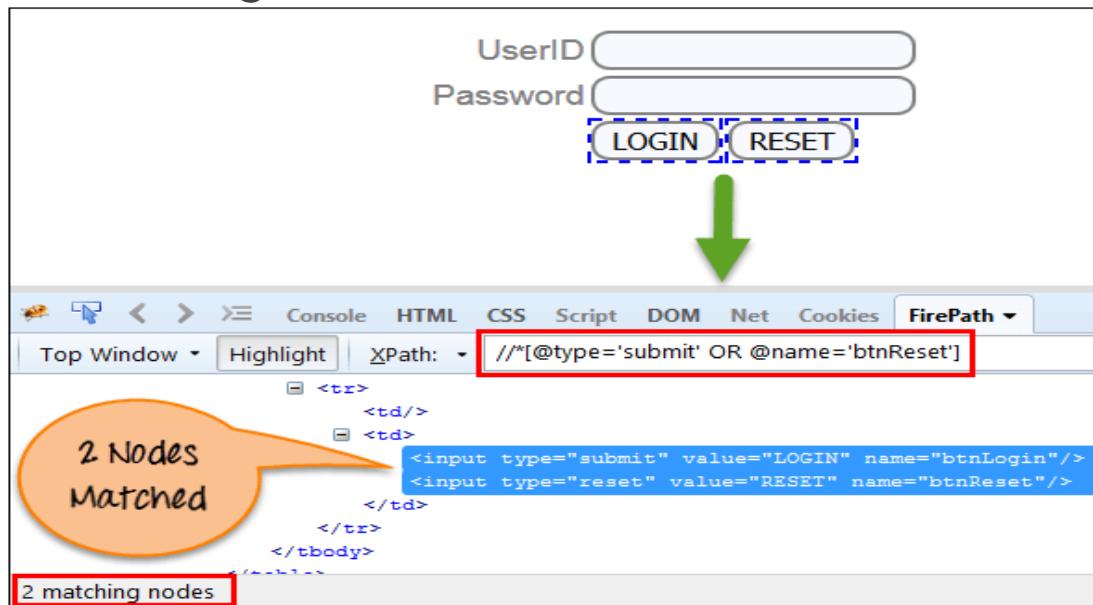
```
Top Window ▾ Highlight | XPath: //*[contains(text(),'here')]

<ol>
  <li>Visit - <a href="http://demo.guru99.com/">here</a></li>
  <li>Enter your email id</li>
  <li>Login credentials is allocated to you and mailed at your id</li>
</ol>
```

1 matching node

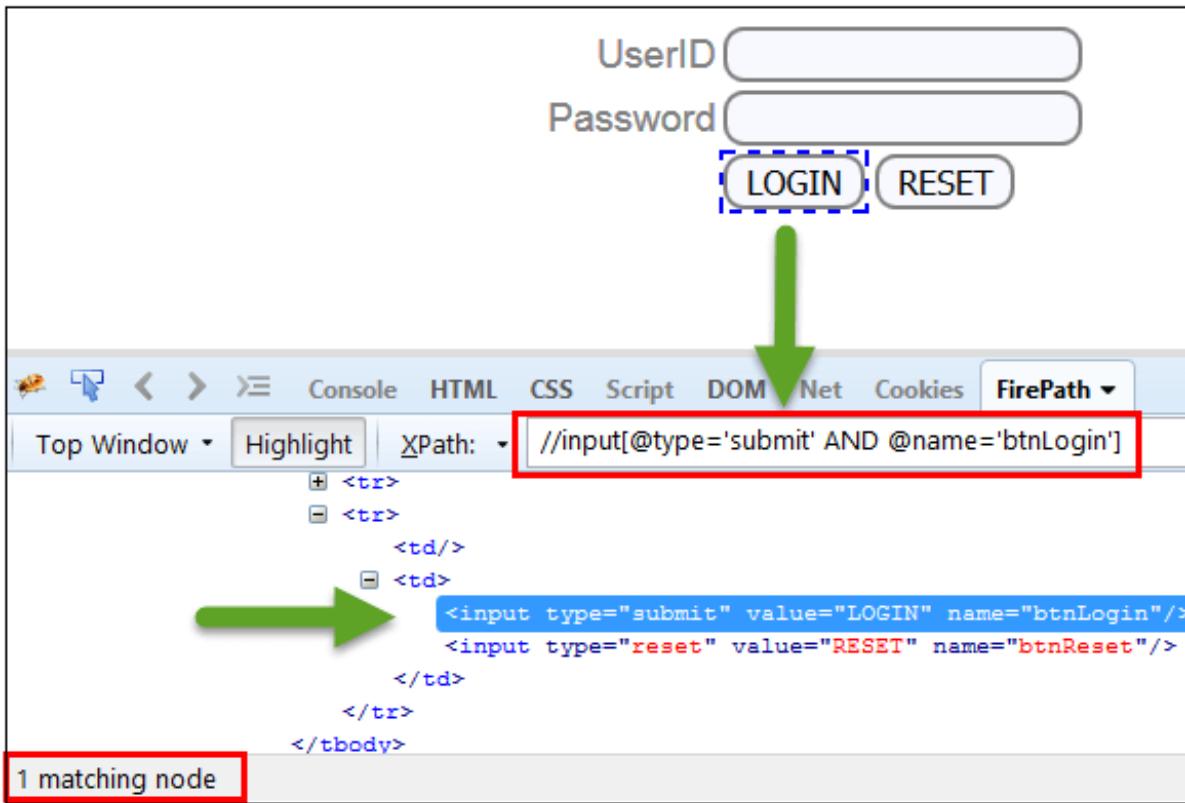
3. Using OR & AND:

- In OR expression, two conditions are used, whether 1st condition OR 2nd condition should be true. It is also applicable if any one condition is true or maybe both. Means any one condition should be true to find the element.
- In the below XPath expression, it identifies the elements whose single or both conditions are true.
 - Xpath= //*[@type='submit' OR @name='btnReset']
- Highlighting both elements as "LOGIN " element having attribute 'type' and "RESET" element having attribute 'name'.



AND

- ▶ In AND expression, two conditions are used, both conditions should be true to find the element. It fails to find element if any one condition is false.
- ▶ Syntax **Xpath= //input[@type='submit' AND @name='btnLogin']**



4. Start-with function:

- ▶ Start-with function finds the element whose attribute value changes on refresh or any operation on the webpage. In this expression, match the starting text of the attribute is used to find the element whose attribute changes dynamically. You can also find the element whose attribute value is static (not changes).
- ▶ For example :- Suppose the ID of particular element changes dynamically like:
 1. Id=" message12"
 2. Id=" message345"
 3. Id=" message8769"
- ▶ and so on.. but the initial text is same. In this case, we use Start-with expression.

- In the below expression, there are two elements with an id starting "message"(i.e., 'User-ID must not be blank' & 'Password must not be blank'). In below example, XPath finds those element whose 'ID' starting with 'message'.
- Syntax: Xpath= //label[starts-with(@id,'message')]

The screenshot shows a web application interface with two input fields: 'UserID' and 'Password'. Each field has an associated validation message: 'User-ID must not be blank' and 'Password must not be blank' respectively. Below the fields are 'LOGIN' and 'RESET' buttons. The FirePath toolbar is open, with the 'Highlight' tab selected. The XPath expression `//label[starts-with(@id,'message')]` is entered in the search bar. A green dashed arrow points from this search bar to the corresponding line in the DOM tree where the label elements are highlighted. A callout bubble on the left side of the DOM tree contains the text 'Id starting with \'message\''. At the bottom left of the FirePath interface, it says '2 matching nodes'.

```

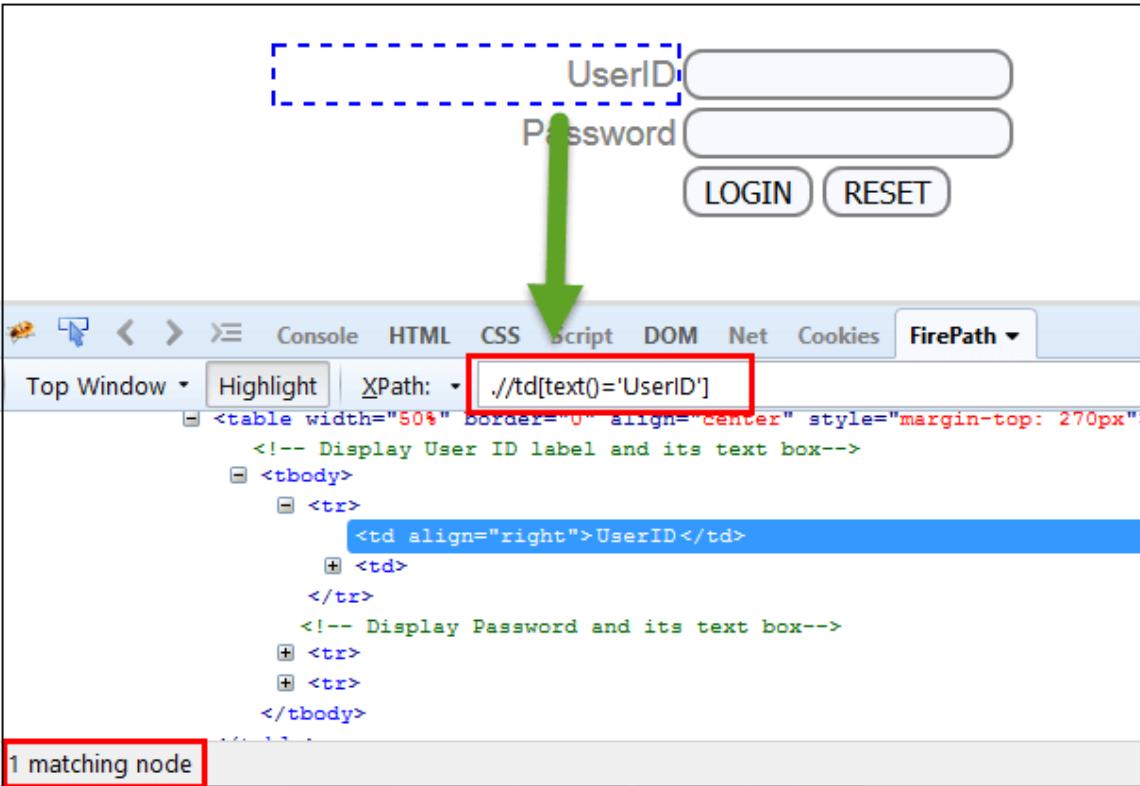
<tr>
  <td align="right">UserID</td>
  <td>
    <input type="text" onblur="validateuserid();" onkeyup="validateuserid();" maxlength="10" name="uid"/>
    <label id="message23" style="visibility: visible;">User-ID must not be blank</label>
  </td>
</tr>
<!-- Display Password and its text box--&gt;
&lt;tr&gt;
  &lt;td align="right"&gt;Password&lt;/td&gt;
  &lt;td&gt;
    &lt;input type="password" onblur="validatepassword();" onkeyup="validatepassword();" name="password"/&gt;
    &lt;label id="message18" style="visibility: visible;"&gt;Password must not be blank&lt;/label&gt;
  &lt;/td&gt;
&lt;/tr&gt;
</pre>


2 matching nodes


```

5. Text()

- In this expression, with text function, we find the element with exact text match as shown below. In our case, we find the element with text "UserID".
- Syntax: **Xpath=//td[text()='UserID']**



6.XPath axes methods

- ▶ These XPath axes methods are used to find the complex or dynamic elements. Below we will see some of these methods.
- ▶ a) **Following**: Selects all elements in the document of the current. node() [UserID input box is the current node] as shown in the below screen.
- ▶ Syntax: Xpath=//*[@type='text']//following::input



- ▶ There are 3 "input" nodes matching by using "following" axis- password, login and reset button. If you want to focus on any particular element then you can use the below XPath method:
- ▶ Syntax: **Xpath=//*[@type='text']//following::input[1]**
- ▶ You can change the XPath according to the requirement by putting [1],[2].....and so on.

Guru99 Bank

An orange speech bubble points to the password input field, which is highlighted with a dashed blue border. The text inside the bubble says "Showing particular Node".

The browser's developer tools are open, specifically the FirePath tab. The XPath expression `//*[@type='text']//following::input[1]` is highlighted in red in the search bar. The FirePath tree shows the following structure:

```

+ <tr>
  <!-- Display Password and its text box-->
  <tr>
    <td align="right">Password</td>
    <td>
      <input type="password" onblur="validatepassword(); onkeyup="validatepassword(); message18"/>
      <label id="message18"/>
    </td>
  </tr>

```

b. Ancestor:

- ▶ The ancestor axis selects all ancestors element (grandparent, parent, etc.) of the current node as shown in the below screen.
- ▶ In the below expression, we are finding ancestors element of the current node("ENTERPRISE TESTING" node).
- ▶ Syntax: **Xpath=//*[text()='Enterprise Testing']/ancestor::div**

The screenshot shows a "Tutorials Library" interface with four main sections: TESTING, SAP, LIVE PROJECTS, and MUST LEARN!. The SAP section is highlighted with an orange oval containing the text "13 Nodes matched". A speech bubble points from the bottom right towards the oval, containing the text "Xpath using ancestor". The FirePath toolbar at the top has a dropdown set to "XPath" with the query `//*[text()='Enterprise Testing']/ancestor::div`. The FirePath panel at the bottom shows the XML structure of the page, with the SAP section being expanded.

Tutorials Library

TESTING

- Learn Software Testing
- QTP (Quick Test Professional)
- Learn Selenium
- Learn Mobile App Testing
- Learn Cucumber Testing
- Learn SoapUI
- Learn Agile Testing

SAP

- Learn SAP Beginner
- Learn SAP ABAP
- Learn SAP UI5
- Learn SAP Data Services
- Learn SAP SD
- Learn SAP CRM

LIVE PROJECTS

- Live Testing Project
- Live Selenium Project
- Live Ecommerce Project
- Live UFT Testing
- Live HP ALM Exercise
- Live Mobile Testing
- Live Security Testing

MUST LEARN!

- Learn Excel Tutorials
- Learn Accounting
- Learn Ethical Hacking
- Cloud Computing for Beginners
- Learn Photoshop CC
- Learn BigData
- Learn Digital Marketing

FirePath

Top Window Highlight XPath: `//*[text()='Enterprise Testing']/ancestor::div`

```
<document>
  <html lang="en-gb" xml:lang="en-gb" slick-uniqueid="3">
    <head>
```

c. Child

- ▶ Selects all children elements of the current node (Java) as shown in the below screen.
- ▶ Syntax: **Xpath=//*[@id='java_technologies']/child::li**

The screenshot shows a web page with four main sections: TESTING, SAP, LIVE PROJECTS, and MUST LEARN!. Each section contains a list of learning topics. The TESTING section includes QTP, Learn Selenium, Learn Mobile App Testing, Learn Cucumber Testing, Learn SoapUI, and Learn Agile Testing. The SAP section includes SAP ABAP, HR/HCM, FICO, Basis, SD, CRM, MM, CO, and Payroll. The LIVE PROJECTS section includes Testing Project, Selenium Project, Ecommerce Project, UFT Testing, IIP ALM Exercise, Mobile Testing, Security Testing, PHP Project, Scrum(Agile) Testing, and Insurance Testing. The MUST LEARN! section includes Excel Tutorials, Accounting, Ethical Hacking, Cloud Computing, Photoshop CC, BigData, Digital Marketing, Business Analyst, Informatica, and Project Management.

A speech bubble in the bottom-left corner contains the text "XPath using child".

The FirePath toolbar at the top has tabs for Console, HTML, CSS, Script, DOM, Net, Cookies, and FirePath. The FirePath tab is selected, showing the XPath expression `//*[@id='java_technologies']/child::li`. The DOM tree below the toolbar shows the structure of the page, with the `ul` element under `#java_technologies` expanded to show its 11 child `li` elements.

71 matching nodes

```
<h4>
<ul id="java_technologies" class="menu">
    <li>
    <li>
    <li>
    <li>
    <li>
    <li>
    <li>
    <li>
    <li>
    <li>
</ul>
```

d. Preceding:

- ▶ Select all nodes that come before the current node as shown in the below screen.
- ▶ Syntax: Xpath=//*[@type='submit']//preceding::input

The screenshot shows a login interface with two text input fields ('UserID' and 'Password') and two buttons ('LOGIN' and 'RESET'). A blue dashed box highlights the 'Password' input field. An orange callout bubble says 'Showing 2 Nodes'. Below the interface is the FirePath tool interface. The 'XPath' field contains the expression `//*[@type='submit']//preceding::input`. The DOM tree shows the HTML structure of the login form. At the bottom left of the FirePath interface, a red box highlights the text '2 matching nodes'.

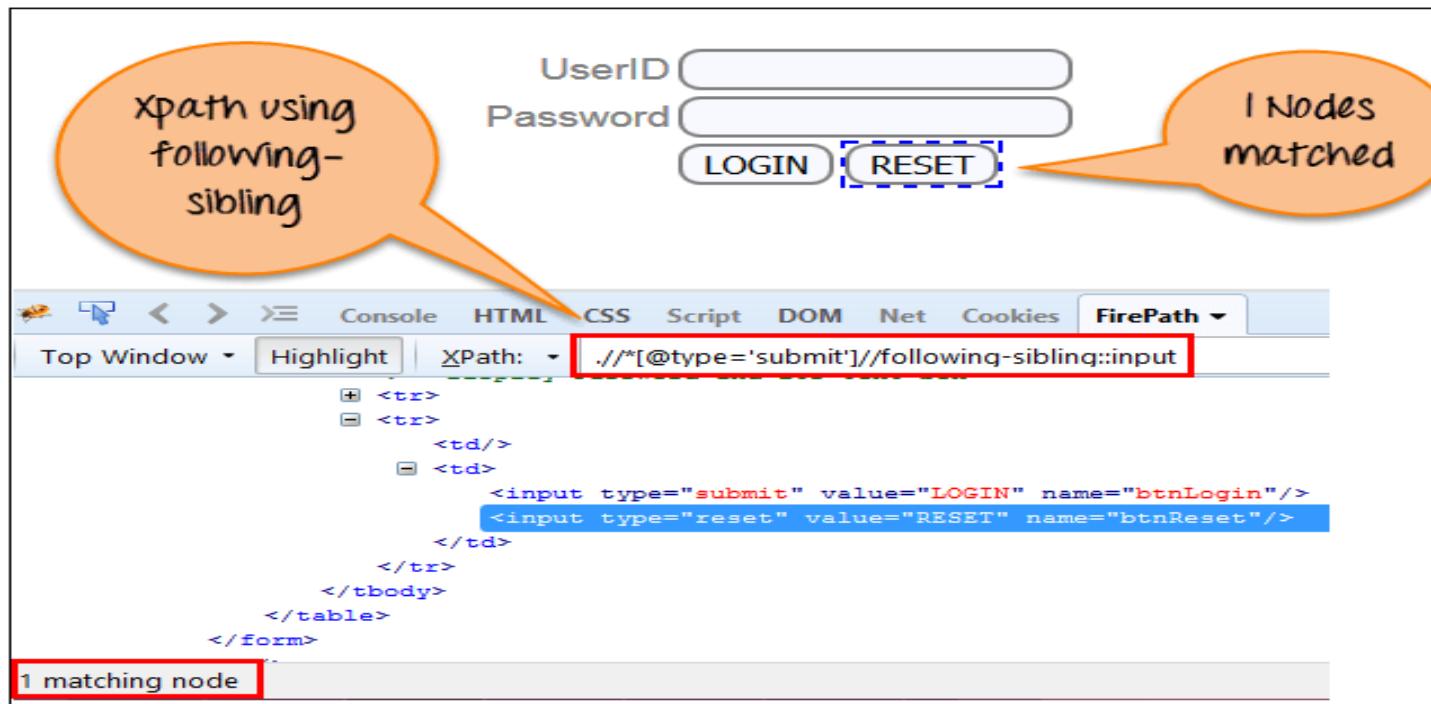
Showing 2 Nodes

Xpath using preceding

2 matching nodes

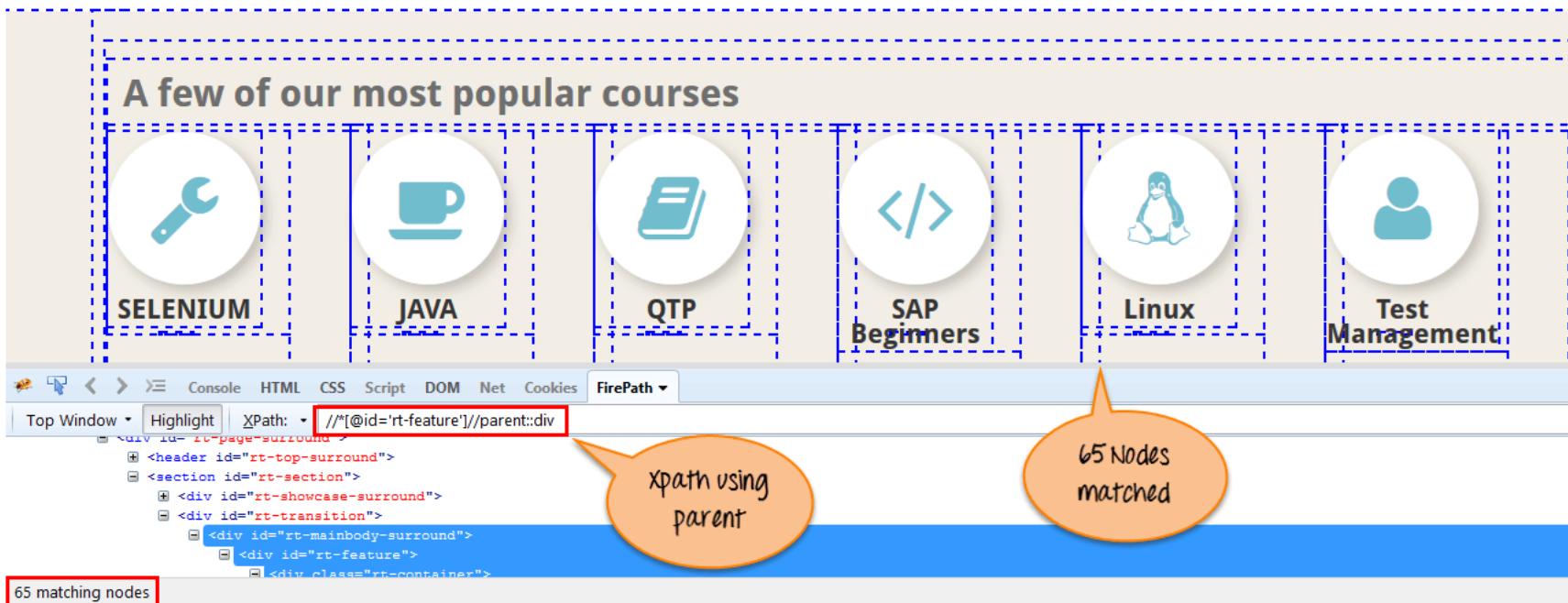
e. Following-sibling:

- ▶ Select the following siblings of the context node. Siblings are at the same level of the current node as shown in the below screen. It will find the element after the current node.
- ▶ Syntax: **xpath=//*[@type='submit']//following-sibling::input**



f. Parent:

- ▶ Selects the parent of the current node as shown in the below screen.
- ▶ Syntax: **Xpath=//*[@id='rt-feature']//parent::div**



g. Self:

- ▶ Selects the current node or 'self' means it indicates the node itself as shown in the below screen.
- ▶ Syntax: Xpath = //*[@type='password']//self::input

Guru99 Bank

UserID

Password (highlighted with a dashed blue border)

LOGIN RESET

Showing 1 Node

Xpath using self

FirePath

XPath: //*[@type='password']//self::input

```
<tr>
  <td align="right">Password</td>
  <td>
    <input type="password" onblur="validatepassword(); onkeyup="validatepassword(); name='password'/" name="password" />
    <label id="message18"/>
  </td>
</tr>
```

h. Descendant:

- ▶ Selects the descendants of the current node as shown in the below screen.
- ▶ In the below expression, it identifies all the element descendants to current element ('Main body surround' frame element) which means down under the node (child node , grandchild node, etc.).
- ▶ Syntax: **Xpath=//*[@id='rt-feature']//descendant::a**

The screenshot shows a web page titled "A few of our most popular courses" with six course icons: SELENIUM, JAVA, QTP, SAP Beginners, Linux, and Test Management. Each icon has a dashed blue border around its circular container. Below the page is the FirePath toolbar, with the "XPath" tab selected. The XPath expression `//*[@id='rt-feature']//descendant::a` is entered in the input field. A status message "12 matching nodes" is displayed at the bottom left. The FirePath interface shows the DOM tree with the following expanded section:

```
<canvas height="130px" width="125px"/>


- + a href="/selenium-tutorial.html">
  - + h4>
          <a style="color:#343434;" href="/selenium-tutorial.html">SELENIUM</a>


```

Annotations include an orange speech bubble pointing to the "XPath using descendant" text with the label "12 matching nodes". Another orange speech bubble points to the "12 Nodes matched" text in the DOM tree.

Locating by CSS Selector

- ▶ CSS Selectors are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. Locating by CSS Selector is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.
- ▶ CSS Selectors have many formats, but we will only focus on the most common ones.
 - ✓ Tag and ID
 - ✓ Tag and class
 - ✓ Tag and attribute
 - ✓ Tag, class, and attribute
 - ✓ Inner text

Locating by CSS Selector - Tag and ID

- ▶ Again, we will use Facebook's Email text box in this example. As you can remember, it has an ID of "email," and we have already accessed it in the "Locating by ID" section. This time, we will use a CSS Selector with ID in accessing that very same element.
- ▶ Syntax: `d.findElement(By.cssSelector("#twotabsearchtextbox"))`
- ▶ **Keep in mind that the ID is always preceded by a hash sign (#).**

Syntax	Description
--------	-------------

`css=tag#id`

- tag = the HTML tag of the element being accessed
- # = the hash sign. This should always be present when using a CSS Selector with ID
- id = the ID of the element being accessed

Locating by CSS Selector - tag, class, and attribute

Syntax	Description
<code>css=tag.class[attribute=value]</code>	<ul style="list-style-type: none">• tag = the HTML tag of the element being accessed• . = the dot sign. This should always be present when using a CSS Selector with class• class = the class of the element being accessed• [and] = square brackets within which a specific attribute and its corresponding value will be placed• attribute = the attribute to be used. It is advisable to use an attribute that is unique to the element such as a name or ID.• value = the corresponding value of the chosen attribute.

Locating by CSS Selector - Class

- ▶ Syntax: `d.findElement(By.cssSelector(".twotabsearchtextbox"))`
- ▶ To find unique element by using more attributes
 - ▶ `Input.formBtn[name=go]`

Locating by CSS Selector - inner text

- ▶ As you may have noticed, HTML labels are seldom given id, name, or class attributes. So, how do we access them? The answer is through the use of their inner texts. **Inner texts are the actual string patterns that the HTML label shows on the page.**

Syntax	Description
--------	-------------

`css=tag:contains("inner text")`

- tag = the HTML tag of the element being accessed
- inner text = the inner text of the element

There are there important special characters:

1. '^' symbol, represents the starting text in a string.

Syntax: `css=input[id^='ema']`

2. '\$' symbol represents the ending text in a string.

Syntax: `css=input[id$='mail']`

3. '*' symbol represents contains text in a string.

Syntax: `css=input[id*='mai']`