

# One of many approaches to build XPath for HTML element on a web page

This is very basic description of building XPath just for quick start.

Please, find more information about locators and XPath expressions by following links:

- <https://www.guru99.com/xpath-selenium.html>
- <http://www.softwaretestingclass.com/complete-guide-on-xpath-in-selenium/>
- <http://www.softwaretestinghelp.com/using-selenium-xpath-and-other-locators-selenium-tutorial-5/>
- <http://toolsqa.com/selenium-webdriver/choosing-effective-xpath/>

# XPath

- XPath is defined as XML path. It is a syntax or language for finding any element on the web page using XML path expression
- These XPath expressions look very much like the path expressions you use with traditional computer file systems:



```
<html debug="true">
  <head>
  <body>
    <div id="wrapper">
      <div id="leftnav">
        <div id="logo">
          
        </div>
        <div id="userpermit">
        </div>
        <div id="navmenu">
        </div>
        <div id="footer">
        </div>
      </div>
    </div>
  </body>
</html>
```

# Build XPath

- Basically, each HTML element looks like

**<tag attribute="value", attribute1="value1"></tag>**


- For this element XPath will look like

**//tag[contains(@attribute, 'value')]**

where contains is a key word

For test scripts this XPath will be something like  
instructions: look for the element with tag 'tag' which  
contains 'attribute' that is equal to 'value'

# Example of element in HTML



The screenshot shows a web browser window with a login page. The page has a sidebar on the left with a logo and links: 'Log In', 'Forgot Password', and 'Contact Us'. The main content area has a heading 'Welcome to Employment Services', a link for 'Français', and a 'Log In' section with input fields for 'Email address' and 'Password', and a 'Log In' button. A red box highlights the 'Email address' input field, and a red arrow points from the HTML code below to it.

Inspect

Console HTML CSS Script DOM

```
<div id="page" style="height: 254px;">
  <div id="feedbackmessages"/>
  <div class="post" id="main">
    <h1>Welcome to Employment Services</h1>
    <div/>
    <div>
      <h2>Log In</h2>
      <form id="HomeForm" method="post" action="./homepage?12-1.IFormSubmitListener-HomeForm">
        <div style="width: 0px; height: 0px; position: absolute; left: -100px; top: -100px; overflow: hidden;">
          <label class="homepagel">Email address</label>
          <input type="email" id="emailAddress" autofocus="" maxlength="64" value="" name="emailAddress"/>
          <br/>
          <label class="homepagel">Password</label>
          <input type="password" id="password" maxlength="64" onfocus="starve()" value="" name="password"/>
          <br/>
          <input class="subbutton" type="submit" id="subform" value="Log In"/>
          <br/>
        </div>
      </form>
    </div>
  </div>
</div>
```

# Example of element in HTML

- On the previous slide the web element 'Email address' text field looks like

```
<input  
type="email" id="emailAddress" autofocus="" maxle  
ngth="64" value="" name="emailAddress"/>
```

where **input** is called tag,

**type**, **id**, **autofocus**, **maxlength**, **value** and **name** are called attributes,

and “**email**”, “**emailAddress**”, “**64**” are called values

# XPath for 'Email address' text field

Using template from slide 3 for 'Email address' text field  
XPath can be

```
//input[contains(@type, 'email')]
```

or

```
//input[contains(@name, 'emailAddress')]
```

or

```
//input[contains(@maxlength, '64')]
```

# Check XPath in console

Before using XPath in your code check if it is valid

- Open console in your browser (in Chrome it is in View=>Developer=>Developer Tools)
- Copy XPath to the console in the following format:

`$x("your XPath")`

Example: `$x("//input[contains(@type, 'email')]")`

and hit 'Enter'

- If XPath is valid, console returns an element

# Check XPath in console

The screenshot displays a web browser's developer console on the left and a login page on the right. The console shows the following log entries:

```
> 19:00:00.000 $x("//input[contains(@type, 'email')]")  
< 19:00:00.000 ▶ [input#emailAddress]
```

Red arrows and text provide context for these elements:

- An arrow points from the XPath query `$x("//input[contains(@type, 'email')]")` to the text: **element XPath in format \$x("xpath")**.
- An arrow points from the console output `[input#emailAddress]` to the text: **console returns 'Email address' test field**.
- An arrow points from the XPath query to a red-bordered input field on the login page labeled "Email address".

The login page, titled "Welcome to Employment Services", includes a sidebar with links for "Log In", "Forgot Password", and "Contact Us", and a main form area with fields for "Email address" and "Password", along with a "Log In" button. The "Email address" field is highlighted with a red border.



# List of elements with the same XPath

Sometimes a few elements may have the same XPath. For example, on Home Page there are three elements with XPath `//a[contains(@id, 'link')]` because there are three html elements that have 'a' tag with attribute 'id' and value 'link'

The screenshot illustrates the relationship between the HTML structure and the user interface. On the left, the HTML structure is shown, with the navigation menu links highlighted in a red box. The links are:

- `<a href="/home" id="link">Log In</a>`
- `<a href="/home" id="link">Forgot Password</a>`
- `<a href="/home" id="link">Contact Us</a>`

On the right, the user interface shows the corresponding buttons: 'Log In', 'Forgot Password', and 'Contact Us'. Red arrows point from the HTML links to the UI buttons, demonstrating that these three elements share the same XPath `//a[contains(@id, 'link')]`.

# List of elements with the same XPath

- To select the exact element from the list you may use index of this element
- For previous example 'Log In' link has index 1, 'Forgot Password' link - index 2, and 'Contact Us' link - index 3
- To select 'Log In' link put XPath in between parentheses and add index 1 at the end:

**`(//a[contains(@id, 'link')])[1]`**

The screenshot shows a web browser's developer console on the left and a web page on the right. The console has tabs for 'Elements', 'Console', and 'Sources'. The 'Console' tab is active, showing a log of messages. The first message is a timestamp followed by an XPath query: `$x("(//a[contains(@id, 'link')])[1]")`. The second message is a timestamp followed by the result: `[a#link]`. A red arrow points from the first element of the result, `a#link`, to the 'Log In' link on the web page. The web page has a dark blue sidebar on the left with the 'Employment Ontario Partner' logo and three links: 'Log In', 'Forgot Password', and 'Contact Us'. The 'Log In' link is highlighted with a red box. The main content area on the right is titled 'Welcome to Employment Services' and contains a login form with fields for 'Email address' and 'Password', and a 'Log In' button. At the bottom of the sidebar, there is copyright information: 'Copyright © 2017 Lingua Classica All rights reserved Hosted by Vivio Technologies.'