KMS XPath Practical Guide

|  |  |
| --- | --- |
| **Date:** | 10/24/2016 |
| **Version:** | 1.0 |
| **Status:** | Review |
| **Author:** | Minh Hoang |
| **Reviewed by:** |  |

Contents

[1. Purpose 3](#_Toc465267904)

[2. XPath Introduction 3](#_Toc465267905)

[2.1 What is XPath 3](#_Toc465267906)

[2.2 Why is XPath 3](#_Toc465267907)

[2.3 XPath Expressions 4](#_Toc465267908)

[3. XPath Nodes 4](#_Toc465267909)

[3.1 XPath Terminology 4](#_Toc465267910)

[3.2 Node Selection 4](#_Toc465267911)

[3.3 Predicates 5](#_Toc465267912)

[3.4 Wildcards 7](#_Toc465267913)

[4. XPath Axes 8](#_Toc465267914)

[4.1 Ancestor 8](#_Toc465267915)

[4.2 Ancestor-of-self 8](#_Toc465267916)

[4.3 Child 9](#_Toc465267917)

[4.4 Descendant 9](#_Toc465267918)

[4.5 Descendant-or-self 10](#_Toc465267919)

[4.6 Following 10](#_Toc465267920)

[4.7 Following-sibling 11](#_Toc465267921)

[4.8 Parent 11](#_Toc465267922)

[4.9 Preceding 12](#_Toc465267923)

[4.10 Preceding-sibling 12](#_Toc465267924)

[4.11 Self 13](#_Toc465267925)

[4.12 XPath Axes Example 13](#_Toc465267926)

[5. XPath Operators 16](#_Toc465267927)

[6. Location Path Expressions 17](#_Toc465267928)

[6.1 Absolute Location Path 17](#_Toc465267929)

[6.2 Relative Location Path 19](#_Toc465267930)

[7. XPath function 20](#_Toc465267931)

[7.1 Number Function 20](#_Toc465267932)

[7.2 String Function 21](#_Toc465267933)

[7.3 Node Function 23](#_Toc465267934)

[8. Identification Techniques and Optimized Using XPath for Automation 24](#_Toc465267935)

[8.1 Identification Techniques 24](#_Toc465267936)

[8.1.1 Absolute XPath 24](#_Toc465267937)

[8.1.2 Relative XPath 24](#_Toc465267938)

[8.1.3 Relative XPath Combination of Double Slash 25](#_Toc465267939)

[Usage of Single ‘/’ and double ‘//’ in the XPath 25](#_Toc465267940)

[8.1.4 Partial XPath with ‘contains’ Keyword 26](#_Toc465267941)

[8.1.5 Partial XPath with ‘starts-with’ Keyword 27](#_Toc465267942)

[8.1.6 Partial XPath with ‘text()’ Keyword 27](#_Toc465267943)

[8.2 Optimization of XPath 29](#_Toc465267944)

# Purpose

* Help QA for learning and adapting with Automation task.
* This document are compiled from KMS QAs own experiences and by referring to automation guideline on Internet.

# XPath Introduction

## What is XPath

* XPath stands for XML Path Language.
* The 'X' in XPath comes from its roots in XML, the eXtensible Markup Language.
* The 'Path' in XPath comes from the fact that XPath uses a 'path like' syntax.
* With XPath it is possible to identify parts of an XML document and perform computations on data in the XML document.
* XPath in XML or HTML document shows the direction of software web application's element location through nodes and attributes.
* XPath is a W3C recommendation.

Example:

1. Windows

|  |
| --- |
| **C:\Windows\system** |
| This file system path is familiar to Windows operating system users. The path identifies the 'system' directory located within the 'Windows' directory on the 'C:' drive. On a Windows system, 'C:' is the root of the file system. A backslash i.e. '\' is used to separate each level of the directory hierarchy in a Windows file system path. |

1. XPath

|  |
| --- |
| **/person/name** |
| This XPath location path expression starts at the document root (the topmost level of the XML node hierarchy), which is indicated by a '/' at the beginning of the expression. It then identifies the root element 'person' (the topmost level of the XML element hierarchy), and then identifies all 'name' elements which are children of the 'person' element. In addition to indicating the document root (when used at the beginning of a location path), '/' is also used to separate the hierarchical node levels in the location path. |

## Why is XPath

XPath is used to locate a web element based on its XML path. XML stands for Extensible Markup Language and is used to store, organize and transport arbitrary data. It stores data in a key-value pair which is very much similar to HTML tags. Both being markup languages and since they fall under the same umbrella, XPath can be used to locate HTML elements.

The fundamental behind locating elements using XPath is the traversing between various elements across the entire page and thus enabling a user to find an element with the reference of another element.

## XPath Expressions

The basic building block of XPath is the expression. An expression is simply a string of Unicode characters made up of keywords, symbols, and operands.

Example:

|  |
| --- |
| **/html/body/table** |
| This XPath location path expression starts at the document root (the topmost level of the XML node hierarchy), which is indicated by a '/' at the beginning of the expression. It then identifies the root element 'html' (the topmost level of the XML element hierarchy), and then identifies all 'table' elements which are children of the 'body' element. In addition to indicating the document root (when used at the beginning of a location path), '/' is also used to separate the hierarchical node levels in the location path. |

# XPath Nodes

## XPath Terminology

* The document itself is a document node
* All HTML elements are element nodes
* All HTML attributes are attribute nodes
* Text inside HTML elements are text nodes

Example:

|  |
| --- |
| **<div id="content" class="content" role="main">**  **<p>Sample Text Node</p>**  **</div>** |
| HTML Elements = div, p  HTML attributes = ‘id’, ‘class’, ‘role’  HTML attributes value = ‘content’, ‘content’, ‘main’  HTML text nodes = ‘Sample Text Node’ |

## Node Selection

Useful path expressions:

|  |  |
| --- | --- |
| Path Expression | Description |
| nodename | Selects all nodes with the name "nodename". |
| / | Selects from the root node.  **Note:** if the path starts with a slash ( / ) it always represents an absolute path to an element. |
| // | Selects nodes from the current node that match the selection no matter where they are.  **Note:** if the path starts with a double slash ( // ) it always represents and relative path to an element. |
| . | Selects the current node. |
| .. | Selects the parent of the current node. |
| @ | Selects attributes. |

Example:

|  |  |
| --- | --- |
| **<html>**  **<body>**  **<div id="mainContent">**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber">**  **<input name="phoneNumber">**  **<input name="emailAddress">**  **</div>**  **</body>**  **</html>** | |
| Node Selection | Result |
| input | Selects all nodes with the tag-name “input” |
| /html/body/div/input | Selects all nodes with the tag-name “input” |
| //button | Selects all nodes with the tag-name “button” no matter where they are in the document |
| //input[@name=’emailAddress’] | Select all node with tag-name ‘input’ and have name attribute with value is ‘emailAddress’ |
| /html/body/div/input/.. | Select parent of input node, node selection is <div id=”mainContent”>…</div> |

## Predicates

Used to find a specific node once a path expression is specified

Predicates are placed within braces []

Positional Predicates have numbers, last() or position() within the braces []

Examples:

|  |  |
| --- | --- |
| **<html>**  **<html>**  **<body>**  **<div id="mainContent">**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div name=”sectionContent”>**  **<textarea>Give your comment here…</textarea>**  **</div>**  **<div name=”sectionContent” style="margin: auto">**  **<textarea>Give your comment here…</textarea>**  **</div>**  **</div>**  **<footer>**  **<p>Created by: KMS</p>**  **<p>Contact information:**  **<a href="https://www.kms-technology.com">KMS Technology</a>.**  **</p>**  **<p>Email:**  **<a href="mailto:kms@kms.com">kms-technology@kms-technology.com</a>.**  **</p>**  **</footer>**  **</body>**  **</html>** | |
| Node Selection | Result |
| //input[@name=’firstName’] | Select all node with tag-name ‘input’ and have name attribute with value is ‘’firstName’. |
| //a[@href=”mailto:kms@kms.com”] | Select all node with tag-name ‘a’ with a href attribute of mailto:kms@kms.com. |
| //input[2] | Select the second input tag in the page. |
| //textarea/text() | Get the text within the textarea tag. |
| //div[@name='sectionContent' and @style='margin: auto'] | Select all node with tag name ‘div’ have two attributes specified. |

## Wildcards

Wildcards allows several elements to be chosen at the same time

|  |  |
| --- | --- |
| Wildcard | Description |
| \* | Matches any element node |
| @\* | Matches any attribute node |
| node() | Matches any node of any kind |

Examples:

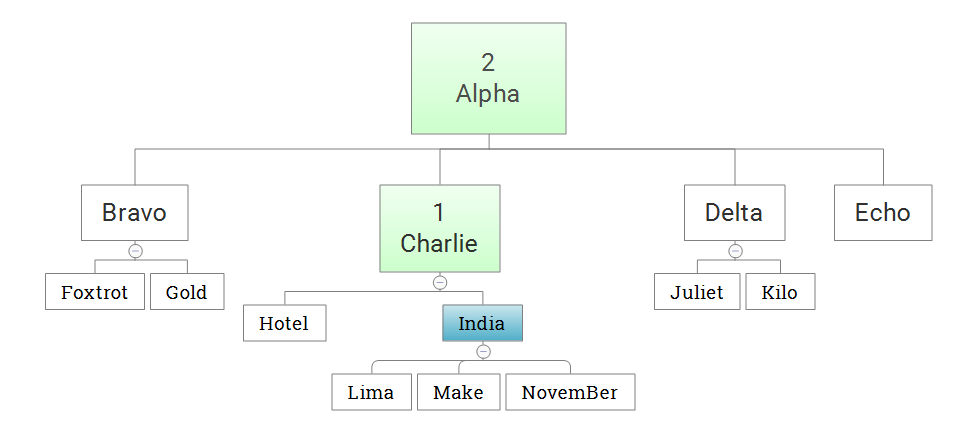
|  |  |
| --- | --- |
| **<html>**  **<html>**  **<body>**  **<div id="mainContent">**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div name=”sectionContent1”>**  **<textarea>Give your comment here…</textarea>**  **<button>Click Me</button>**  **</div>**  **<div name=”sectionContent2” style="margin: auto">**  **<textarea>Give your comment here…</textarea>**  **</div>**  **</div>**  **<footer>**  **<p>Created by: KMS</p>**  **<p>Contact information:**  **<a href="https://www.kms-technology.com">KMS Technology</a>.**  **</p>**  **<p>Email:**  **<a href="mailto:kms@kms.com">kms-technology@kms-technology.com</a>.**  **</p>**  **</footer>**  **</body>**  **</html>** | |
| Node Selection | Result |
| //div[@name=”sectionContent1”]/\* | Selects all the child element nodes of the element with tag-name ‘div’ and has attribute name is ‘sectionContent1’ |
| //\* | Selects all elements in the document |
| //input[@\*] | Selects all input elements which have at least one attribute of any kind |

# XPath Axes

An XPath axis is a path through the node tree making use of particular relationship between nodes.

## Ancestor

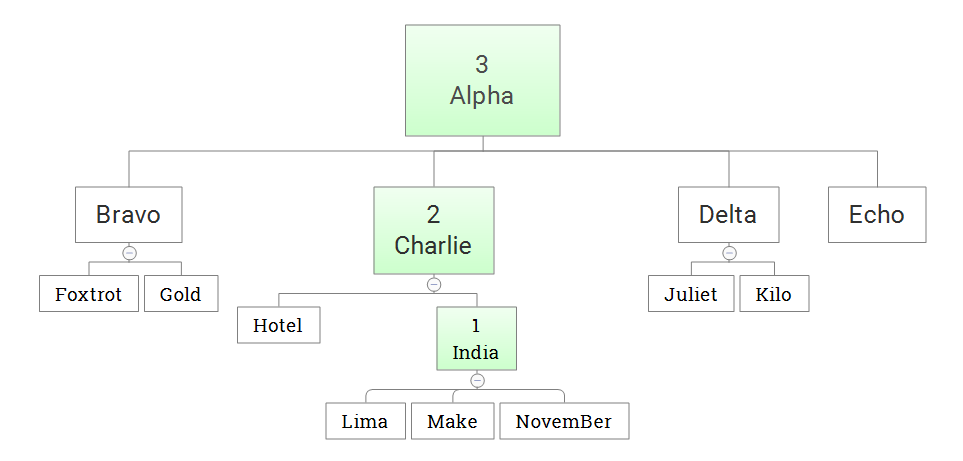
Ancestor axis: Selects all ancestors (parent, grandparent, etc.) of the current node.



India is context node: count (ancestor::\*) returns 2

## Ancestor-of-self

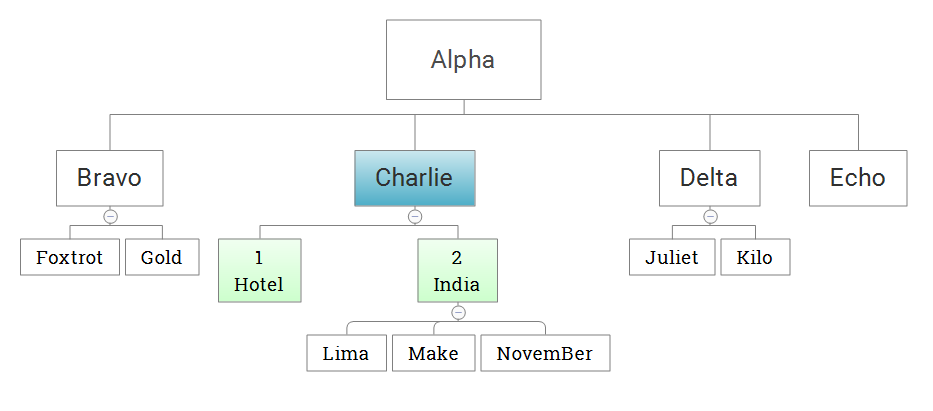
Ancestor-or-self axis: Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself.



India is context node: count (ancestor::\*) returns 3

## Child

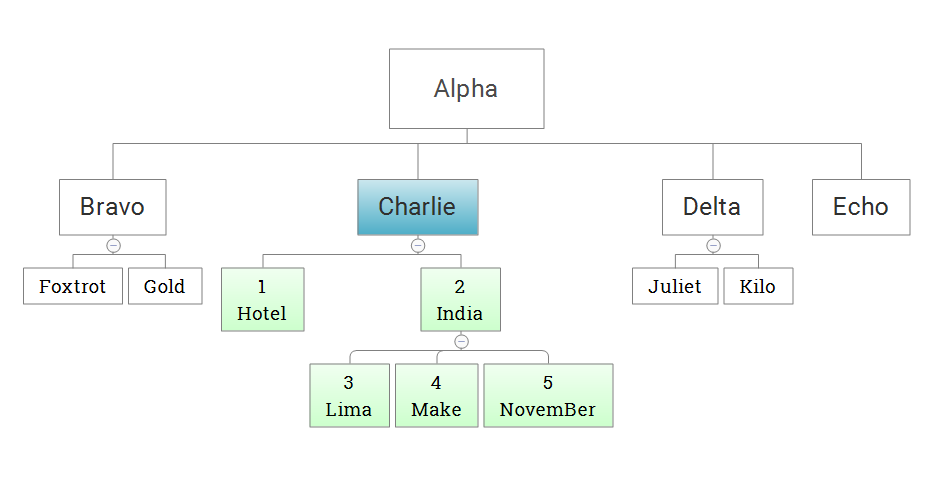
Child axis: Selects all children of the current node.



Charlie is context node, e.g.: count(child::\*) returns 2. Same as count(\*).

## Descendant

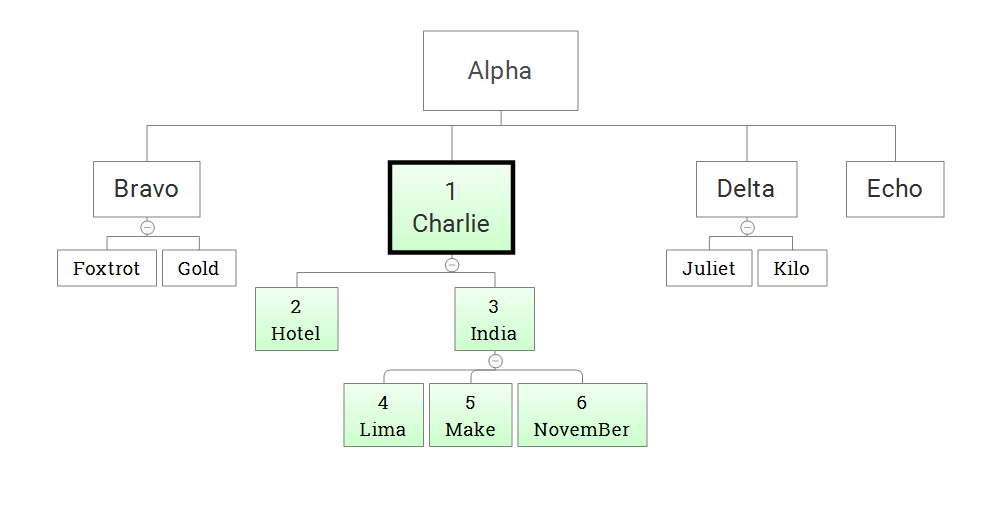
Descendant axis: Selects all descendants (children, grandchildren, etc.) of the current node.



Charlie is context node, e.g.: count(descendant::\*) returns 5.

## Descendant-or-self

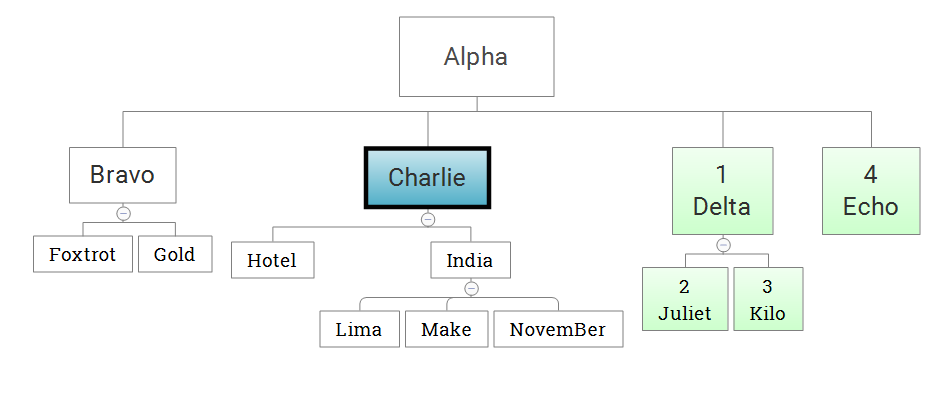
Descendant-or-self axis: Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself.



Charlie is context node, e.g.: count(descendant-or-self::\*) returns 6.

## Following

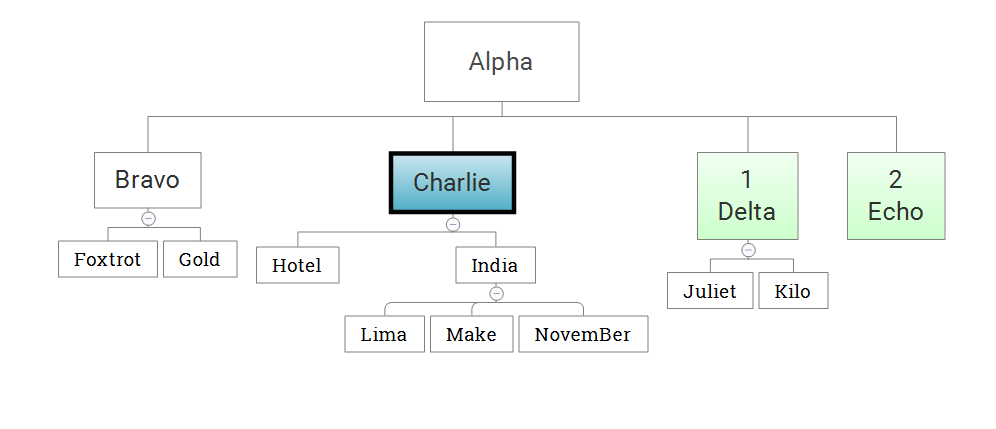
Following axis: Selects everything in the document after the closing tag of the current node.



Charlie is context node, e.g.: count(following::\*) returns 4.

## Following-sibling

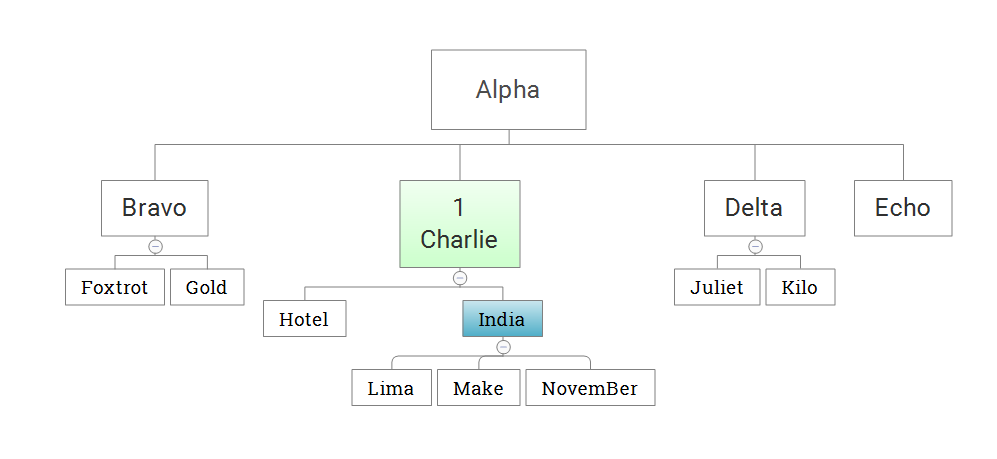
Following-sibling axis: Selects all siblings after the current node.



Charlie is context node, e.g.: count(following::\*) returns 2.

## Parent

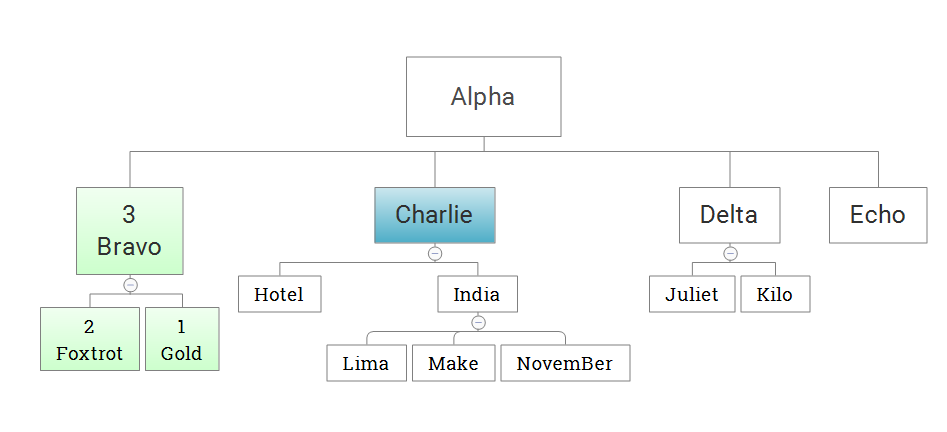
Parent axis: Selects the parent of the current node.



India is context node, e.g.: India/parent::\* return Charlie

## Preceding

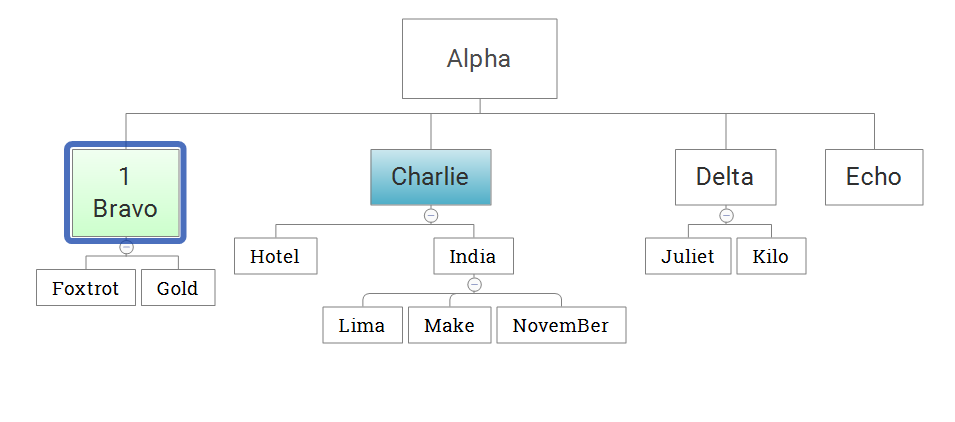
Preceding axis: Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes.



Charlie is context node, e.g.: count(preceding::\*) returns 3

## Preceding-sibling

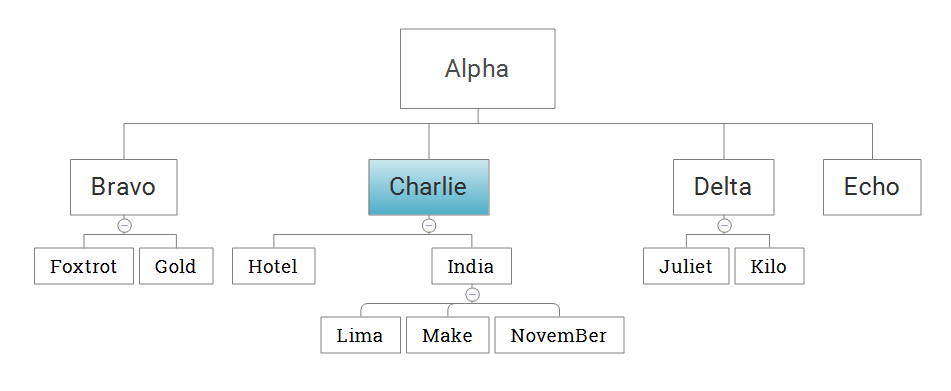
Preceding-sibling axis: Selects all siblings before the current node.



Charlie is context node, e.g.: count(preceding-sibling::\*) returns 1

## Self

Self axis: Selects the current node.



Charlie is context node, e.g.: Charlie/self::\* return Charlie

## XPath Axes Example

|  |  |
| --- | --- |
| **<table>**  **<tbody>**  **<tr id=”1”>**  **<th>Burj Khalifa</th>**  **<td>UAE</td>**  **<td>Dubai</td>**  **</tr>**  **<tr id=”2”>**  **<th>Clock Tower Hotel</th>**  **<td>Saudi Arabia</td>**  **<td>Mecca</td>**  **</tr>**  **<tr id=”3”>**  **<th>Taipei 101</th>**  **<td>Taiwan</td>**  **<td>Taipei</td><**  **</tr>**  **</tbody>**  **</table>** | |
| Node Selection | Result |
| //td[text()='UAE']/ancestor::\* | * <tr id=”1”>…</tr> * <tbody>…</tbody> * <table>…</table> |
| //td[text()='UAE']/ancestor-or-self::\* | * <tr id=”1”>...</tr> * <tbody>…</tbody> * <table>…</table> * <td>UAE</td> |
| //tr[@id='1']/child::\* | * <th>Burj Khalifa</th> * <td>UAE</td> * <td>Dubai</td> |
| //tbody/descendant::\* | * <tr id=”1”>...</tr> * <tr id=”2”>...</tr> * <tr id=”3”>...</tr> * <th>Burj Khalifa</th> * <td>UAE</td> * <td>Dubai</td> * <th>Clock Tower Hotel</th> * <td>Saudi Arabia</td> * <td>Mecca</td> * <th>Taipei 101</th> * <td>Taiwan</td> * <td>Taipei</td> |
| //tbody/descendant-or-self::\* | * <tbody>…</tbody> * <tr id=”1”>...</tr> * <th>Burj Khalifa</th> * <td>UAE</td> * <td>Dubai</td> * <tr id=”2”>...</tr> * <th>Clock Tower Hotel</th> * <td>Saudi Arabia</td> * <td>Mecca</td> * <tr id=”3”>...</tr> * <th>Taipei 101</th> * <td>Taiwan</td> * <td>Taipei</td> |
| //tr[@id='1']/following::\* | * <tr id=”2”>...</tr> * <th>Clock Tower Hotel</th> * <td>Saudi Arabia</td> * <td>Mecca</td> * <tr id=”3”>...</tr> * <th>Taipei 101</th> * <td>Taiwan</td> * <td>Taipei</td> |
| //tr[@id='1']/following-sibling::\* | * <tr id=”2”>...</tr> * <tr id=”3”>...</tr> |
| //tr[@id='1']/parent::\* | * <tbody>…</tbody> |
| //tr[@id='3']/preceding::\* | * <tr id=”2”>...</tr> * <th>Clock Tower Hotel</th> * <td>Saudi Arabia</td> * <td>Mecca</td> * <tr id=”1”>...</tr> * <th>Burj Khalifa</th> * <td>UAE</td> * <td>Dubai</td> |
| //tr[@id='3']/preceding-sibling::\* | * <tr id=”2”>...</tr> * <tr id=”1”>...</tr> |
| //tr[@id='3']/self::\* | * <tr id=”3”>...</tr> |

# XPath Operators

Below is a list of the operators that can be used in XPath expressions:

|  |  |
| --- | --- |
| **Operator** | **Description** |
| | | Computes two node-sets |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| div | Division |
| = | Equal |
| != | Not equal |
| < | Less than |
| <= | Less than or equal to |
| > | Greater than |
| >= | Greater than or equal to |
| or | or |
| and | and |
| mod | Modulus (division remainder) |

Example:

|  |  |
| --- | --- |
| **<html>**  **<body>**  **<div id="mainContent">**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div name="sectionContent">**  **<textarea>Give your comment here inside 1…</textarea>**  **<textarea>Give your comment here inside 2…</textarea>**  **<textarea>Give your comment here inside 3…</textarea>**  **<textarea>Give your comment here inside 4…</textarea>**  **</div>**  **</div>**  **</body>**  **</html>** | |
| Operator | Result |
| //input[@name='firstName'] | //input[@name='lastName'] | * <input name=”firstName”> * <input name=”lastName”> |
| //textarea[1+3] | * <textarea>Give your comment here inside 4…</textarea> |
| //textarea[last()-1] | * <textarea>Give your comment here inside 3…</textarea> |
| //div[@id='mainContent']//\*[position()=count(//div[@id='mainContent']//\*)\*0.5] | * <input name="phoneNumber" value="Mobile Phone 2"> |
| //div[@id='mainContent']//\*[position()=count(//div[@id='mainContent']//\*) div 2] | * <input name="phoneNumber" value="Mobile Phone 2"> |
| count(//div[@id='mainContent']//\*)div 2 = 5 | * true |
| count(//div[@id='mainContent']//\*)div 2 != 5 | * false |
| count(//textarea)>=4 | * true |
| //input[@value='Mobile Phone 1' or @value='Mobile Phone 2'] | * <input name="phoneNumber" value="Mobile Phone 1"> * <input name="phoneNumber" value="Mobile Phone 2"> |
| //input[@name='phoneNumber' and @value='Mobile Phone 2'] | * <input name="phoneNumber" value="Mobile Phone 2"> |
| //div[@name='sectionContent']//\*[position() mod 2=0] | * <textarea>Give your comment here inside 2…</textarea> * <textarea>Give your comment here inside 4…</textarea> |

# Location Path Expressions

## Absolute Location Path

An absolute location path begins with a '/' to signify that it is starting from the document root. The document root is the top level of the XML document's node hierarchy and contains all other nodes in the XML document, including the root element (which is the top level of the XML document's element hierarchy).

An absolute location path can also begin with '//', however '/' and '//' mean different things. The following section includes two examples which illustrate the difference.

Example:

|  |  |
| --- | --- |
| **<html>**  **<body>**  **<div id="mainContent">**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div name="sectionContent">**  **<textarea>Give your comment here inside 1…</textarea>**  **<textarea>Give your comment here inside 2…</textarea>**  **<div name="childContent">**  **<p>Sample Text</p>**  **</div>**  **</div>**  **</div>**  **</body>**  **</html>** | |
| Node Selection | Result |
| /html/body/div/div/div/p | * <p>Sample Text</p>   This XPath expression selects all ‘p’ elements which are children of the 'div' element, which in turn is a descendant element of the document root. |
| /\* | * <html>…</html> * <body>…</body>   This XPath expression selects all element children of the document root. '\*' is an element wildcard. |
| //\* | This XPath expression selects all element descendants of the document root.'//' is the abbreviated form of the 'descendant-or-self' axis. |
| /html/body/div/input/@value | * <input name="phoneNumber" value="Mobile Phone 1"> * <input name="phoneNumber" value="Mobile Phone 2">   This XPath expression selects the ‘value’ attribute of all ‘input’ elements which are children of the 'div' element, which in turn is a descendant element of the document root. |

## Relative Location Path

A relative location path is always evaluated from a context node. A context node can be thought of as the node that the XPath processor is 'currently processing'. The context node can change within an XPath query.

Example:

|  |  |
| --- | --- |
| **<html>**  **<body>**  **<div id="mainContent">**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div name="sectionContent">**  **<textarea>Give your comment here inside 1…</textarea>**  **<textarea>Give your comment here inside 2…</textarea>**  **<div name="childContent">**  **<p>Sample Text</p>**  **</div>**  **</div>**  **</div>**  **</body>**  **</html>** | |
| Node Selection | Result |
| //body//div//button//following-sibling::\* | * <input name="firstName"> * <input name="lastName"> * <input name="phoneNumber" value="Mobile Phone 1"> * <input name="phoneNumber" value="Mobile Phone 2"> * <div name="sectionContent">   This XPath expression first navigates to the parent element of the context node (which is the 'button' element child of the 'div' element). The next step select all following sibling elements. |
| //textarea/../../button | * <button>Click Me</button>   This XPath expression selects the parent element of the parent element of the context node. '..' is an abbreviated form of the 'parent::' axis specifier. |
| //\*[@name=’lastName’] | * <input name="lastName">   This XPath expression selects all child element nodes which have name attribute is ‘lastName’ of the context node. |

# XPath function

## Number Function

XPath defines the following functions on numbers to be used with the XPath expressions.

|  |  |
| --- | --- |
| Function | Description |
| Ceiling() | Returns the smallest integer larger than the value provided. |
| Floor() | Returns the largest integer smaller than the value provided. |
| Round() | Returns the rounded value to nearest integer. |
| Sum() | Returns the sum of two numbers. |

Example:

|  |  |
| --- | --- |
| **<html>**  **<body>**  **<div id="mainContent">**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div id=”count” hidden=”true”> 10</div>**  **<div name="sectionContent">**  **<p id=”count” hidden=”true”>15</p>**  **<textarea>Give your comment here inside 1…</textarea>**  **<textarea>Give your comment here inside 2…</textarea>**  **<div name="childContent">**  **<p>Sample Text</p>**  **</div>**  **</div>**  **</div>**  **</body>**  **</html>** | |
| Function | Result |
| ceiling(count(//div[@id='mainContent']//\*) div 3) | 4 |
| floor(count(//div[@id='mainContent']//\*) div 3) | 3 |
| round(count(//div[@id='mainContent']//\*) div 3) | 3 |
| sum( //\*[@id='count']) | 25 |

## String Function

XPath defines the following functions on string to be used with the XPath expressions.

|  |  |
| --- | --- |
| Function | Description |
| starts-with(string1, string2) | Returns true when first string starts with the second string. |
| contains(string1, string2) | Returns true when the first string contains the second string. |
| substring(string, offset, length?) | Returns a section of the string. The section starts at offset up to the length provided. |
| substring-before(string1, string2) | Returns the part of string1 up before the first occurrence of string2. |
| substring-after(string1, string2) | Returns the part of string1 after the first occurrence of string2. |
| string-length(string) | Returns the length of string in terms of characters. |
| normalize-space(string) | Trims the leading and trailing space from string. |
| translate(string1, string2, string3) | Returns string1 after any matching characters in string2 have been replaced by the characters in string3. |
| concat(string1, string2, ...) | Concatenates all strings. |

Example:

|  |  |
| --- | --- |
| **<html>**  **<body>**  **<div id="mainContent">**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div id=”count” hidden=”true”> 10</div>**  **<div name="sectionContent">**  **<p id=”count” hidden=”true”>15</p>**  **<textarea>Give your comment here inside 1…</textarea>**  **<textarea>Give your comment here inside 2…</textarea>**  **<div name="childContent">**  **<p>Sample Text</p>**  **</div>**  **</div>**  **</div>**  **</body>**  **</html>** | |
| Function | Result |
| starts-with(//textarea[1]/text(),'Give') | true |
| contains(//textarea[1]/text(),'Give this') | false |
| substring(//textarea[1]/text(),0,10) | Give your |
| substring-before(//textarea[1]/text(),'inside') | Give your comment here |
| substring-after(//textarea[1]/text(),'inside') | 1… |
| string-length(//textarea[1]/text()) | 32 |
| normalize-space(//textarea[2]/text()) | Give your comment here inside 2… |
| translate(//div[@name='childContent']//p/text(),'T','t') | Sample text |
| concat(//div[@name='childContent']//p/text(),' With Another String') | Sample Text With Another String |
| //\*[contains(text(),'Give')] | * <textarea>Give your comment here inside 1…</textarea> * <textarea>Give your comment here inside 2…</textarea> |
| //\*[contains(@name,'Name')] | * <input name="firstName"> * <input name="lastName"> |
| //\*[strarts-with(text(),'Give')] | * <textarea>Give your comment here inside 1…</textarea> * <textarea>Give your comment here inside 2…</textarea> |
| //\*[starts-with(@name,'phone')] | * <input name="phoneNumber" value="Mobile Phone 1"> * <input name="phoneNumber" value="Mobile Phone 2"> |

## Node Function

XPath defines the following functions on nodes to be used with the XPath expressions.

|  |  |
| --- | --- |
| Function | Description |
| comment() | Selects nodes which are comments. |
| node() | Selects all kinds of nodes. |
| processing-instruction() | Selects nodes which are processing instruction. |
| text() | Selects a text node. |
| name() | Provides the name of the node. |
| position() | Provides the position of the node. |
| last() | Selects the last node relative to current node. |

Example:

|  |  |
| --- | --- |
| **<html>**  **<body>**  **<div id="mainContent">**  **<!--This is first comment-->**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div id=”count” hidden=”true”> 10</div>**  **<div name="sectionContent">**  **<!--This is second comment-->**  **<p id=”count” hidden=”true”>15</p>**  **<textarea>Give your comment here inside 1…</textarea>**  **<textarea>Give your comment here inside 2…</textarea>**  **<div name="childContent">**  **<!--This is third comment-->**  **<p>Sample Text</p>**  **</div>**  **</div>**  **</div>**  **</body>**  **</html>** | |
| Function | Result |
| //comment() | * <!--This is first comment--> * <!--This is second comment--> * <!--This is third comment--> |
| //div[@name='childContent']/node() | * <!--This is third comment--> * <p>Sample Text</p> |
| //\*[text()='Sample Text'] | * <p>Sample Text</p> |
| name(//\*[text()='Sample Text']) | * Result return tag name is ‘p’ |
| //div[@id='mainContent']//input[position()<5 and position()>3] | * <input name="phoneNumber" value="Mobile Phone 2"> |
| //div[@id='mainContent']//input[last()] | * <input name="phoneNumber" value="Mobile Phone 2"> |

# Identification Techniques and Optimized Using XPath for Automation

## Identification Techniques

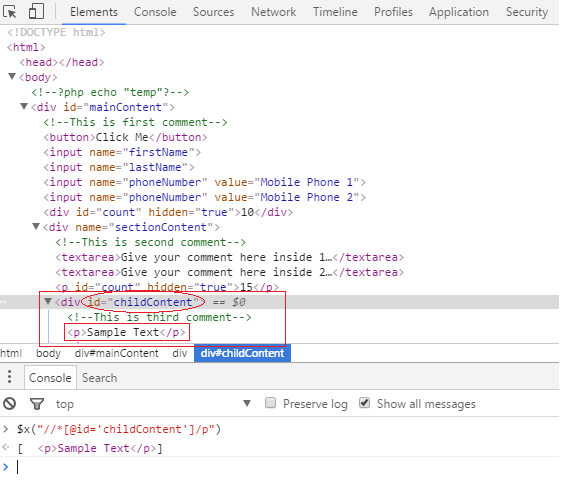
### Absolute XPath

The easiest way of finding the XPath is to use the Browser Inspector tool to locate an element and get the XPath of it:

XPath generated by the tool is: /html/body/div/div/div/p

### Relative XPath

At times XPath generated by tool are too lengthy and you see there is a possibility of getting a shorter XPath. Above XPath will technically work, but each of those nested relationships will need to be present 100% of the time, or the locator will not function. Above choose XPath is known as Absolute XPath. There is a good chance that your XPath will vary in every release. It is always better to choose Relative XPath, as it helps us to reduce the chance of element not found exception. To choose the relative XPath, it is advisable to look for the recent Id attribute. Look below at the HTML code of the above screen shot.



**Absolute** XPath: /html/body/div/div/div/p

**Relative** XPath: //\*[@id=’childContent’]/p

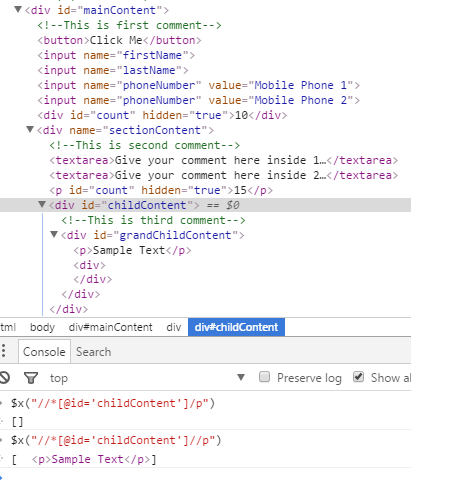
### Relative XPath Combination of Double Slash

Relative XPath can be choose in many ways and to understand that, it is required to understand the usage of single & double slashes in the XPath.

Usage of Single ‘/’ and double ‘//’ in the XPath

A single slash ‘/’ anywhere in XPath signifies to look for the element immediately inside its parent element.

A double slash ‘//’ signifies to look for any child or any grand-child element inside the parent element.

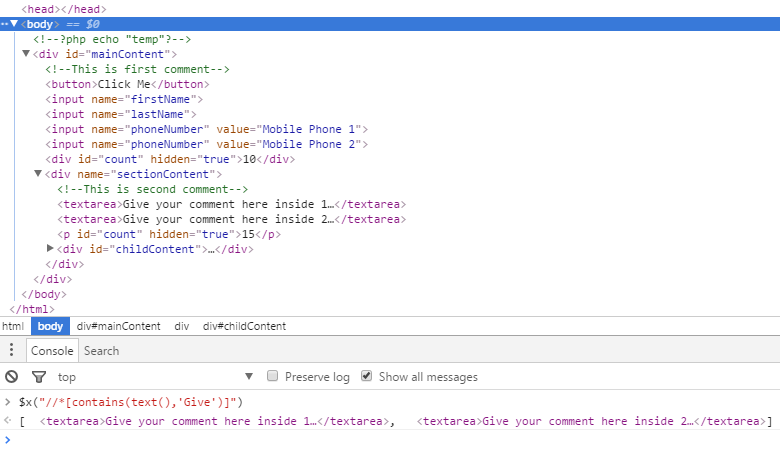


**Relative** XPath without using double slash: //\*[@id=’childContent’]/div/p

**Relative** XPath with using double slash: //\*[@id=’childContent’]//p

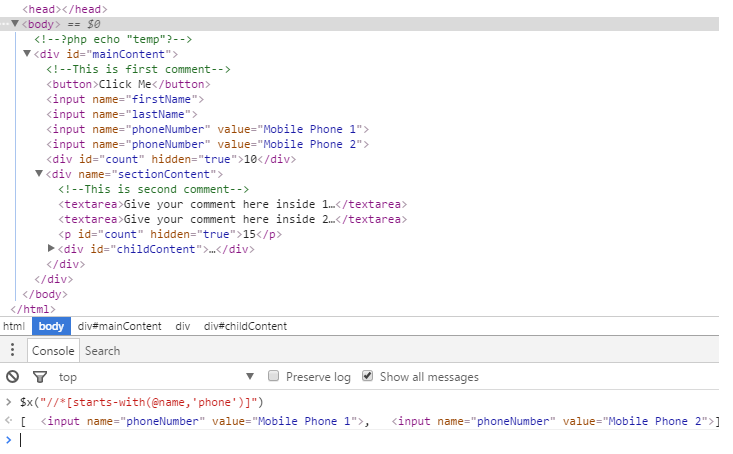
### Partial XPath with ‘contains’ Keyword

Contains method is used when partial attribute value or partial text associated with the web element.



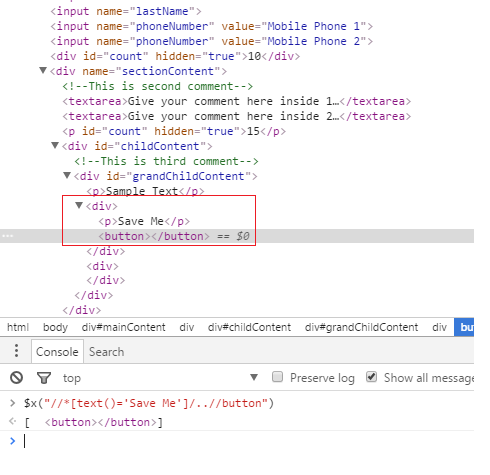
### Partial XPath with ‘starts-with’ Keyword

‘starts-with()’ method is used when the initial partial attribute value or initial partial text associated with the web element. User can also use this method to locate web elements those are consist of both the static (initial) and dynamic (trailing) values.



### Partial XPath with ‘text()’ Keyword

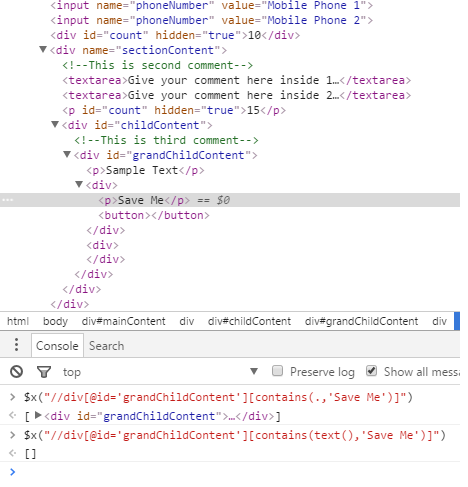
‘text()’ method is used when the text nodes that are children of the context node, using this method for locate relative element which is contain text nodes.



The difference between dot (“ . ”) and text():

* The dot (".") in XPath is called the "context item expression" because it refers to the context item. This could be match with a node (such as an element, attribute, or text node) or an atomic value (such as a string, number, or Boolean). While text() refers to match only element text which is in string form.
* The dot (".") notation is the current node in the DOM. This is going to be an object of type Node while Using the XPath function text() to get the text for an element only gets the text up to the first inner element. If the text you are looking for is after the inner element you must use the current node to search for the string and not the XPath text() function.

Example:



XPath expression //div[@id='grandChildContent'][contains(text(),'Save Me')] can find div element by using text ‘Save Me’.  
XPath expression //div[@id='grandChildContent'][contains(.,'Save Me')] cannot find div element by using text ‘Save Me’.

## Optimization of XPath

There many cases where simple changes are made to the application and previously identified XPath Statements won’t work. In order to avoid such problems, it’s necessary to optimize the XPath Statements before using them in Selenium Automation.

The Advantage of Optimizing the XPath Statements are to get the shortest and least breakable XPath Statements. Following are the few strategies in order to optimize the XPath:

1. Use id/name attribute if available (in single page web applications use text() method to locate element).
2. Use the combination of attributes to make the XPath more specific.
3. Use the Relative XPath instead of Absolute XPath Statements.
4. Always avoid using indexes in XPath.
5. Verify the XPath using Selenium IDE commands or using chrome developer tools ($x(“ [your XPath expression]”)).
6. Using [..] to move to parent of the present node.
7. Use XPath functions in XPath wherever necessary to better identification.
8. Use Preceding-sibling or Following-sibling wherever applicable.
9. Identification of objects with same attribute values.
10. Handling dynamic attribute Values.

Example:

|  |
| --- |
| **<html>**  **<body>**  **<div id="mainContent">**  **<!--This is first comment-->**  **<button>Click Me</button>**  **<input name="firstName">**  **<input name="lastName">**  **<input name="phoneNumber" value="Mobile Phone 1">**  **<input name="phoneNumber" value="Mobile Phone 2">**  **<div id=”count” hidden=”true”> 10</div>**  **<div name="sectionContent">**  **<!--This is second comment-->**  **<p id=”count” hidden=”true”>15</p>**  **<textarea>Give your comment here inside 1…</textarea>**  **<textarea>Give your comment here inside 2…</textarea>**  **<div name="childContent">**  **<!--This is third comment-->**  **<p>Sample Text</p>**  **<input type=”text” value=”Input Text”></input>**  **</div>**  **</div>**  **</div>**  **<table>**  **<tbody>**  **<tr><th>No.</th><th>First Name</th><th>Last Name</th></tr>**  **<tr><td>1</td><td>A1</td><td>B1</td></tr>**  **<tr><td>2</td><td>A2</td><td>B2</td></tr>**  **<tr><td>3</td><td>A3</td><td>B3</td></tr>**  **</tbody>**  **</table>**  **</body>**  **</html>** |
|  |

* Use ID/Name attribute if available

If the ID/Name attribute is present for an object, use it in XPath even though the object can be identified with any other attributes. ID/Name attribute should have the highest priority.

Example:

//input[@name=’firstName’]

//div[@name=’childContent’]

* Use combination of attribute in XPath

If ID/Name attribute is not available, any other attribute which uniquely identifies the object can be used. Also combination of 2 or more attributes can be used to identify the object such as name, type, class, value etc. This will make the XPath more specific.

Example:

//input[@type=’text’ and @value=’Input Text’]

* Use Relative XPath instead of Absolute XPath

Absolute XPath refers that XPath starts from root i.e. from ‘/’ to till the desired html element is reached such as /html/body/p[2]. Relative path refers to the XPath that starts from specific point to the desired element. It starts with ‘//’.It is always recommended to use Relative XPath then Absolute XPath for following reasons.

* Absolute XPath will be very long compare to Relative XPath and hence difficult to manage
* In case of Absolute XPath, there are high chances that XPath may break even if there are
* small changes introduced
* Absolute XPath has the disadvantage of typing the expression to a particular input
* Structure

Example:

Absolute XPath: /html/body/div/div/div/input

Relative XPath: //input[@value='Input']

* Avoid Using Indexes in XPath

In XPath always avoid using indexes to identify a specific node. These are not reliable and with slightest modification XPath may break. It always advisable to locate element that in relation to an existing stable element rather than to rely on meaningless indexes. This means, when reading through your elements you are more likely to understand where this becomes more obvious.

Example:

In HTML sample above all text in table will not change. Below is sample XPath for finding element contains ‘B3’ text.

Relative XPath: //\*[text()='B3']/parent::\*

Relative XPath with indexes: //table//tr[4]

Second XPath in example above cannot find correct element if SUT change position of table row.

* Using .. (double dot symbol) to move to parent of the present node

Assume that you are in a specific node and want to move to parent of that node then it is advisable to use .. (double dot symbol) to navigate to parent node then using Parent::<node type> or ancestor::<node type>. This will reduce the length of the XPath and identification of node faster during execution.

In XPath . (A Single dot) refers to present node and ..(a double dot) refers to parent of present node.

Example:

//\*[text()='B3']/ /..

* Use XPath function XPath wherever necessary to better identification

As XPath is a Programming Language, it has many built-in functions which user can use in XPath Statements. During selenium automation, sometime there are problems of identifying the objects on a page which have same attributes there is no way of distinguishing them. In such cases XPath function comes very handy, some of the XPath functions used are:

* Position() –This function is for locating specific html tag such as //input[position()=5] to locate
* 5th element
* Last() – This function is for locating last element node of the specific node type such as //input[last()]. Further it can be specified as //input[last()-1] to identify an node before the last node
* Starts-with()- This is one of the predefined methods of XPath Language which is used in XPath Statements to locate the element starting with the specified text or to locate the element node containing an attribute value which is starting with the specified text.
* Contains () –This is one of the predefined methods of XPath Language which is used in XPath Statement to locate the element node containing specified text on the page or to locate the element node containing specific text in its attribute value.
* Use Preceding-sibling or Following-sibling wherever applicable

If it is needed to navigate to immediate following or preceding node of an element, then it is advisable to use following-sibling or preceding-sibling as applicable instead of just using following or preceding axes. This will enable faster/efficient identification of such elements

Example:

//div//input[following-sibling::input[@value='Mobile Phone 1']] – take all the input elements which has following-sibling element is ‘input’ has attribute value is ‘Mobile Phone 1’

* Identification of objects with same attribute values

There are cases were two objects have same attribute value in a specific web page. In such cases identify a nearby object with unique identification and then navigate to intended object.

* Handling dynamic attribute Values

In case finding element need to use value of an attribute which is dynamic, then we can use Selenium.getAttribute() or selenium.getText() or any other methods to fetch the values and use it in XPath.