Xpath Video Tutorials

Thanks a lot for purchasing the Xpath Video Tutorials Series. I hope you learn something new and it will help you with Xpath.

If you have questions, or didn't understand something, please send me a support ticket: support@bot-factory.com

As you know, I'm a practical and solution oriented guy. That means I focus 100% on the things that work and are necessary to get the stuff done. No theoretical ballast or useless stuff ☺

Everything I teach you in this course has helped me to generate Xpath expressions for every project I was involved with so far.

And please keep in mind. Your Xpath Expressions doesn't need to always solve the job completely. You can still format and modify the data later as well. In Ubot Studio, php, python or C#. Wherever you work with Xpath.

So don't waste too much time building the 100% correct xpath expression. If you could extract a data block and then extract the data you need later via Regex or looping through a list for example. The result is what matters. Always focus on the result.

Ok, let’s start:

1. Html Structure

Html code is structured. Elements within other elements.

```
<a>
        <div>
        </div>
</a>
```

So in this example the A element is the first level. And the div element is within the A element. So the div element is the second level.

Xpath works in a similar way. You select an element, or multiple elements. And then you either move in(deeper into the levels), move out (to a higher level), or you extract data from the level you selected.

A html element can have attributes

```
<a data="xx" data2="x2">
        <div>
        </div>
</a>
```

In this case the A element has two attributes. Data and data2.

Also important is the fact, that it doesn’t matter how the code is formatted.

This:

```
 <a data="xx" data2="x2">
        <div>
        </div>
</a>
```

Is similar to:

```
<a data="xx" data2="x2"><div></div></a>
```

Of course the readability is different, but the html code is 100% identical.

2. Select an element

```
<a>
        <div>
        </div>
</a>
```

So let’s write our first Xpath Expression. You can start at the beginning of the html code, or you can tell the Xpath Engine to „search the whole thing“. Which means search through all levels.

Let’s say we want to select the „a“ element in our example:

/a

This will work fine because the „a“ element is on what I call level 0. So it’s not within something else.

In our example you could not write:

/div

This won’t work, because the div element is within the „a“ element.

But you can write

/a/div

That would select the „div“ element.

Or you use // which means „search the whole document and select everything that matches.

So if I write: //div

This will select the div element. No matter where it is in the code.

And if there are multiple div elements, //div will select all of them.

If you have code like:

```
<a>
        <div>
        </div>
</a>
<div>
</div>
```

And you writhe:

//a/div

This will search the whole thing, select all A attributes and then look 1 level down for a div element. If there is one, that div element will be selected.

In our example only the first div element would get selected, because it’s the only one that is 1 level below an a element.

Let’s say we have something like this:

```
<a>
    <xx>
        <div>
        </div>
    </xx>
</a>
<div>
</div>
```

We want to only select the div element that is within the „a“ element. But we don’t know where (what level) the „div“ element is within the „a“ element. Then we can use:

```
//a//div
```

This will select all a elements, and then select all div elements who are somewhere (any level) within those a elements.

3. Select by attributes

```
<a data="x1">
        <a data="x2">
        </a>
        <a data="y2">
        </a>
</a>
```

Let’s say we want to select the A element where the data attribute equals y2.

```
//a[@data="y2"]
```

If we want to select all of them where the data attribute contains a 2 we would use:

```
//a[contains(@data,"2")]
```

That will select the x2 and y2 one in our example.

We can also search for innertext. Innertext is the text that is written within an element:

```
<a data="x1">
	<a data="x2">
	Innertext 1
	</a>
	<a data="y2">
	Innertext 2
	</a>
</a>
```

To search for innertext we would use:

```
//a[contains(text(),"Innertext")]
```

4. Inside out Search

Sometimes you have to select an element and then move up a level.

```
<a data="x1">
    <x>
    </x>
    <a data="x2">
        <a data="y2">
        </a>
    </a>
</a>
```

Let’s say we want to select the 2nd level (data=x2). But we can’t do it directly. Because we don'' know the attribute name or the attribute value.

But we can select the y2 one.

```
//a[@data="y2"]/..
```

This one will select the a element where the data attribute equals y2. And then move one level up. /.. will move one level up.

/../.. will move 2 levels up. And so on.

You can also do stuff like:

```
//a[@data="y2"]/../../x
```

Which will select the a element where the data attribute equals y2. Then it will move 2 levels up. Which in our example is the first element (data=x1). And then it will move 1 level down to the x element (/x).

5. Conditions

You can also use the not condition.

If you want to select all a elements where the data attribute is NOT y2:

```
//a[not(@data="y2")]
```

Or if you you want to select all a elements where the data attribute does NOT contain 2 for example:

```
//a[not(contains(@data,"2"))]
```

We can also use the conditions “and” and “or”

```
//a[@data1="test1" and @data2="test2"]

//a[@data1="test1" or @data2="test2"]

//a[contains(text(),"innertext") and @data2="test2"]

//a[contains(text(),"innertext ") and contains(@data2,"2")]

//a[contains(text(),"innertext ") or contains(@data2,"2")]
```

6. Wildcards

To match any element you can use:

```
//*
```

```
//*[@data="Test1"]
```

This will select all elements where the data attribute equals Test1

Works with all the other conditions as well:

```
//*[contains(text(),"innertext")]
//*[contains(@data,"2")]
//*[not(contains(@data,"2"))]
```

We can also use a wildcard for attributes:

```
//a[@*="2"]
```

This will select all elements where one of the attributes has the value 2

Also works for the other conditions:

```
//a[contains(@*,"2")]
//a[not(contains(@*,"2"))]
```

And you can combine it:

```
//*[@*="data1"]
```

Select all elements where one attribute equals data1.

7. Number Conditions

If we have something like:

```
<p1 price1="10" price2="99">

</p1>

<p1 price1="20" price2="55">

</p1>
```

We can use things like:

```
//p1[@price1<11]
```

This will select all p1 elements where the price1 attribute is higher than 11.

You can of course also use > >= <=

And you can combine it with the and or conditions as well:

```
//p1[@price1<11 and @price2>70]
```

8. Additional Information’s

and has a higher priority than or and both operators have lower priority than the relational and equality operators (=, !=, >, >=).

So, it is safe to write: A = B and C = D

Some most frequent mistakes made:

People write AND and/or OR. Remember, XPath is case-sensitive.

People use the | (union) operator instead of or

9. Xpath Expression Cheat Sheet

You need an expression. Here's the list of everything we covered in the tutorials:

`//a`

Search the whole thing and select all a elements

`/a`

Start at the beginning and select all A elements who are on level 1.

`//a/div`

Search the whole thing for a elements. Then look one level deep into the a elements and select all div elements who are below an a element.

`//a//div`

Search the whole thing for a elements. Then search through all levels of the selected a elements and select all div elements who are somewhere within an a element

`//p[@title]`

Select all p elements who have a title attribute

`//p[@title="Test2"]`

Select all p elements where the title attribute has the value Test2

`//p[contains(@title,"yyy")]`

Select all p elements where the title attribute contains yyy

`//p[contains(text(),"HTML")]`

Select all p elements where the innertext contains HTML

`//a/p[@title="Test2"]`

Search all p elements where the title attribute equals Test2 and who are one level below an a element.

`//a//p[@title="Test2"]`

Search all p elements where the title attribute equals Test2 and who are below an a element (any level deep within the a element).

`//a[contains(@href,"ebay.com")]`

Select all a elements where the href attribute contains ebay.com

`//a[contains(@href,"ebay.com")]/..`

Select all a elements where the href attribute contains ebay.com.

Then move one level up (/..) and select all the elements who are one level above the a elements we just selected.

`(//a)[2]`

Select the second a element from all the a elements that have been found.

```
(//a[contains(@href,"ebay.com")])[5]
```

Select the fifth a element from all the a elements that have been found where the href attribute contains ebay.com.

```
//p1[@data1="xx1" and @data2="yy1"]
```

Select all p1 elements where the data1 attribute matches xx1 and the data2 attribute matches yy1

```
//p1[@data1="xx1" or @data2="yy1"]
```

Select all p1 elements where the data1 attribute matches xx1 or the data2 attribute matches yy1

```
//p1[not(@data1="xx2")]
```

Select all p1 elements where the data1 attribute doesn’t match xx2

```
//p1[not(contains(text(),"HTML"))]
```

Select all p1 elements where the innertext doesn’t contain HTML

```
//p1[contains(text(),"HTML") and @data2="yy1"]
```

Select all p1 elements where innertext contains HTML and where the data2 attribute matches yy1

```
//p1[contains(text(),"HTML") and contains(@data2,"y1")]
```

Select all p1 elements where innertext contains HTML and where the data2 attribute contains y1

```
//*[@data2="yy2"]
```

Select all elements where the data2 attribute equals yy2

```
//*[contains(@data2,"2")]
```

Select all elements where the data2 attribute contains 2

```
//*[contains(text(),"HTML")]
```

Select all elements where the innertext contains HTML

```
//p1[@*="yy2"]
```

Select all p1 elements where any attribute equals yy2

```
//p1[contains(@*,"2")]
```

Select all p1 elements where any attribute contains 2

```
//p1[@price1<13]
```

Select all p1 elements where the price1 attribute is below 13

```
//p1[@price1>13]
```

Select all p1 elements where the price1 attribute is above 13

```
//p1[@price1>=13]
```

Select all p1 elements where the price1 attribute is above or equal to 13