Optimized Identification Techniques Using XPath

7/11/2013
IBM
Gururaja Rao & Arvind Pachunoori

Abstract: Selenium is the web automation tool which is reliable platform, independent and compatible with the different environments. As many of the teams in IBM are using Selenium, one of the grey area that discourage the users from adapting to selenium is, poor identification mechanism used. This has a huge consequence such as unreliable, inconsistent, incompatible scripts and hence leading to lot of maintenance efforts and false quality concerns. These obstacles were overcome by adopting various strategies. This article shares optimized techniques for identification of objects in selenium automation.

Gururaja Rao is working as staff software engineer for Atlas product in ECM. From past two years he is working on automation testing using Selenium. Reach out to him at gururajarao@in.ibm.com

Arvind Pachunoori is working as an advisory software engineer (Technical) QA for Atlas team under ECM. The team uses selenium for their application to automate Web UI. This team is one of the first teams to integrate the Selenium with RQM. Reach out to him at apachuno@in.ibm.com

Introduction

This article basically concentrates on how we can identify the objects in an Optimized manner using XPath which are reliable and consistent. We observed that there are applications which deal with different kind of objects; not having any id/name(s) and many dynamic objects not having any unique identification. Also there are lots of technologies integrated in an application such as GWT, DOJO, Flash etc which makes it difficult to develop a consistent code.

So handling such complex scenarios and making scripts run successfully, is a big challenge. We used accurate methodology with xpath and Xpath axes to overcome such difficulties. Additionally we have identified the ways it works for IE, Firefox and other browsers.

So we are going to demonstrate various techniques that are used for identifying the objects especially in critical scenarios.

Basis of XPath

In Selenium automation, the XPath is a very powerful way to parse the HTML code of any web page and find out the XPath path for identifying the elements on the web page. If properly used, it can produce very reliable and low maintenance locators. But when used an inefficient xpath, it can lead to very inconsistent results.

XPath is a syntax for defining parts of an XML document. It uses the path expressions to navigate in XML documents. It is like a small programming language, it has functions, expressions, wild card characters etc. XML documents consist of trees of nodes. The topmost element of the tree is called the root element followed by descendant nodes.

Let's look into the terminology used to identify these nodes:

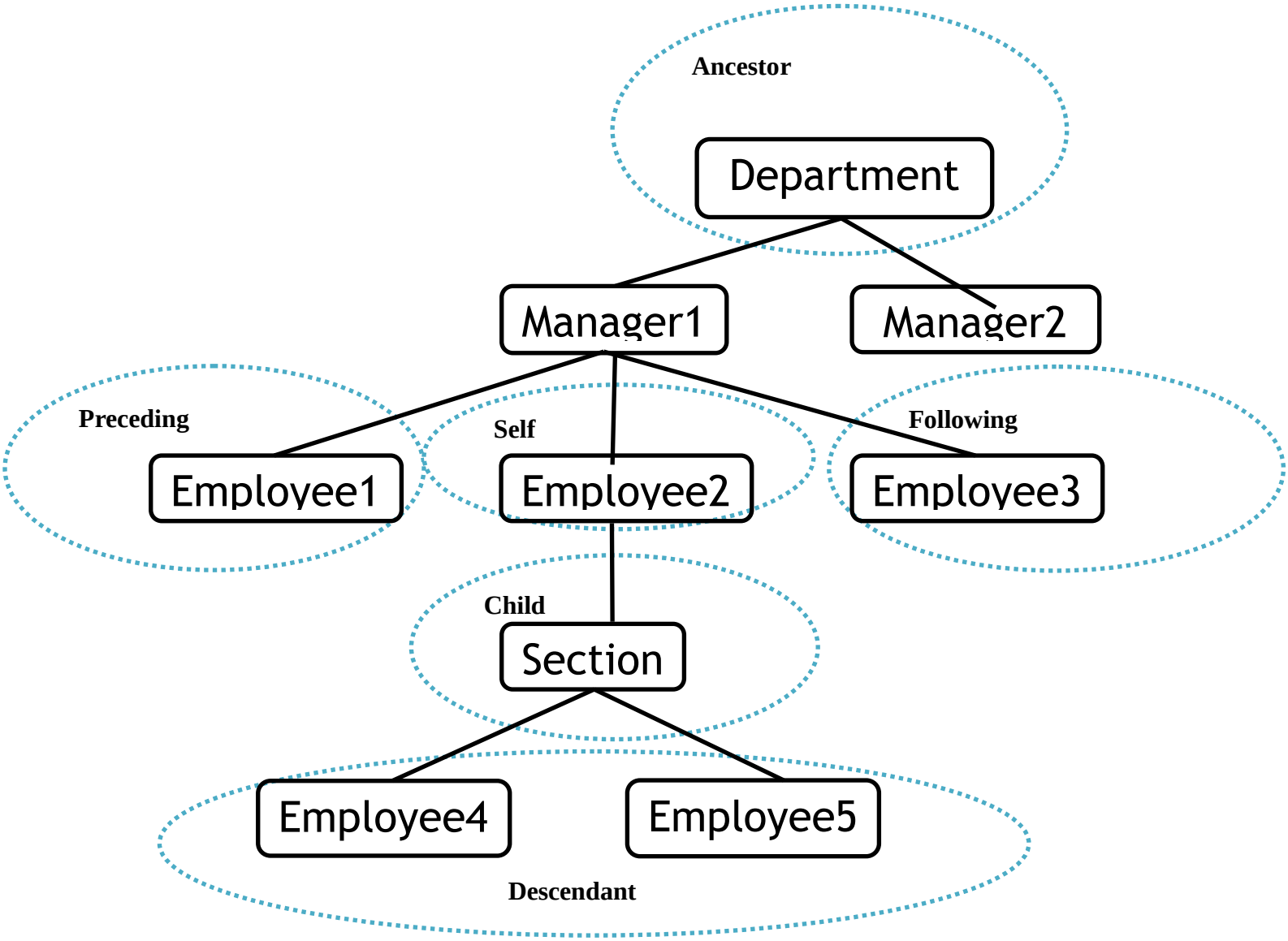
- Each element has one parent except root
- Each element may have zero or one or more child elements referred as **child**
- Nodes can have same parent called siblings. Among siblings, the nodes which are to the immediate left are referred as **preceding-sibling** and other left nodes as **preceding**. The nodes to the immediate right are referred as **following-sibling** and other right nodes are referred as **following**.
- A nodes parent, parent's parent etc is referred as **Ancestor**
- A nodes children, children's children is referred as **Descendant**

XPath Axes

An axis defines a node-set relative to the current node.

Axis Name	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

Following example explains the various facets of Xpath identification



In the above example we illustrate by considering **Employee2** as node for reference for referring to other nodes

Employee2 node has

- Employee3 as following sibling
- Employee1 as Preceding sibling.

For Employee2,

- Manager1 is the parent
- Department is the ancestor (Parent of Parent).

For Employee2,

- Section is the Child node
- Employee4 and Employee5 are Descendant nodes

Optimization of XPath

We observed that 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 attribute if available
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
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. Identification of image objects
11. Handling dynamic attribute Values

Let's look into each of them in detail:
Consider the following XML example "Example1" for illustration of these strategies
Example-1

```
</pre>
<table>
<tbody>
<tr id="item1">
<td class="name item">Mp3 Download: foobar.mp3</td>
<td class="name item">
        <input class="name item form field disabled" type="text" name="qty" />
```

```
</td>
</tr>
<tr id="item2">
<td class="name item">Mp3 Player</td>
<td class="name item">
 <input id="item2_quantity" class="name item form field required" type="text" name="qty"/>
</td>
</tr>
</tbody>
</table>
<pre>
```

Use ID attribute if available

If the ID attribute is present for an object, use it in XPath even though the object can be identified with any other attributes. ID attribute should have the highest priority. For example : Based on Example1, It can be used as

//tr[@id='item1']

@ symbol indicate following is an attribute of the object.

Use combination of attributes in XPath

If ID 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. For example, based on Example1, we can have

//input[@type='text' and @name='qty']

In the above example Boolean operator 'and 'operator is used in XPath statement to combine more than one attributes.

Using combination of properties in XPath is useful in following cases

- There are cases where there does not exist any single attribute which uniquely identify the object
- Even if there is exist single attribute which uniquely identifies the object, we can use combination of attributes to make Xpath more robust and hence there is less chances of failure in identification

Use Relative XPath instead of Absolute Xpath

Absolute Path 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

For example, based on Example1, we can have an absolute path as follows

/pr/table/tbody/tr[@id='item1']

Using Relative path we can have

//tr[@id='item1']

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.

Let us see example Exmaple1:

You can target the highlighted input field using
xpath=//table/tr[2]/td/input

xpath=(//table[@name='cart']//input)[2]
xpath=//input[@id='item2_quantity']

xpath=//input[contains(@class='required')]
xpath=//input[contains(@class='required')] and @type='text']

As you can guess, the first two expressions are not as reliable as others. Of these //table/tr[2]/td/input is the worst because it would break even with slight modifications to the page structure. In any case do not rely on tools, including selenium IDE, to generate the right xpath expression for you – they are usually the worst.

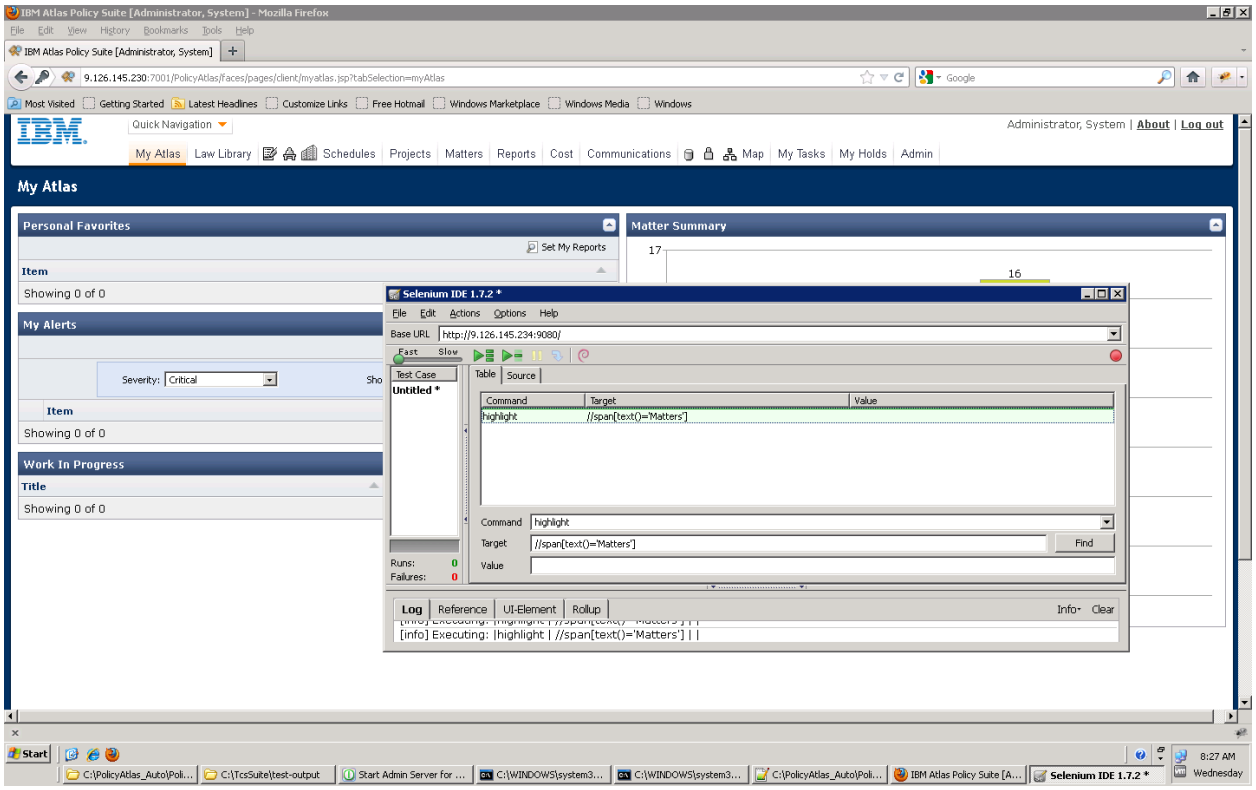
It always advisable to code the Selenium test to find that item in relation to an existing item that *has* been programmed properly rather than to rely on meaningless indexes. This means, when reading through your script you are more likely to understand where

//input[@id='BottomTotal']/following-sibling::input[contains(@value,'Add')]
is compared to
//li[6]/input[4].

This becomes more obvious when you have a large number of tests in your test suite.

Verify the xpath using Selenium IDE commands

Once Xpath is formed, verify it using Selenium IDE. There are cases where a specific Xpath may satisfy two or more objects. Use “highlight” command in Selenium IDE to highlight the intended object as shown below. This will ensure that Xpath is accurate.



Further we can also use “click”, “verifyVisible” , “type” etc. command in selenium IDE to verify the object.

It is advisable to use both “highlight” as well as “verifyVisible” commands to verify the as some of the objects may successfully highlight but not visible.

“click” command can be used for objects such as “Button” and “type” can be used to type the characters into a text box

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. Consider the following example
Based on Example1, the parent of the node td i.e. tr can be reached using

//td[@class='name item']/../[@id='item1]

Use XPath functions in XPath wherever necessary to better identification

As XPath is a Programming Language, it has many built-in functions which we 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.
For Example
//div[starts-with(text(),'Approver')] -Matches the element with string which begins with 'Approver'
//div[starts-with(@id,'test')] – Matches the element with attribute begins with string 'test'
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.
For example
//span[contains(text(),'Approver')]—Matches any element with text contains the string 'Approver'
//span[contains(@id,'testing')] – Matches any element with id attribute containing string 'testing'

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 o preceding axes. This will enable faster/efficient identification of such elements
Based on Example1, we can have
//tr[@id='item2']/preceding-sibling::tr[@id='item1']
//tr[@id='item1']/following-sibling::tr[@id='item2']

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.

Consider the following example:
In the below example there are 2 checkboxes with same name as **resendChkBox**, so to identify such cases we can initiate relative X-Path with Escalation Rule value and proceed to that checkbox object.

//td[text()='Escalation Rule #1']/parent::tr/following-sibling::tr/descendant::span/input[@name='resendChkBox']

//td[text()='Escalation Rule #2']/parent::tr/following-sibling::tr/descendant::span/input[@name='resendChkBox']

Identification of image objects

The objects of type Image can be usually identified by src and alt attributes such as

//img[contains(@src,'cancel')]
//img[@alt='Search']

Handling dynamic attribute Values

If we 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 a XPath

As for example:
String fetchText=selenium.getAttribute("//a[contains(text(),'click')]/@href")
In this case the value of href attribute will be stored in a variable fetchText .
Similarly in the following case the the text value of element with name containing 'testing' will be stored in the variable p.
String p=selenium.getText("//p[contains(@name,'testing')]")

Some Illustration on usage of XPath

//a[text()='ApproverName']/ancestor::td/preceding-sibling::td/input

//span[text()='Confirmation Instructions:']/ancestor::td/following-sibling::td/descendant::a/img

//div[text()='Send Reminder Every']/parent::td/following-sibling::td/descendant::td/select

//td[text()='Escalation Rule #1']/../following-sibling::tr/descendant::td/div[contains(text(),' Use for the first')]/input

//div[text()='Attorney']/parent::td/preceding-sibling::td/child::div[text()='Initial Receipt']

//td[text()='Created/Updated']/following-sibling::td[text()='From']/following-sibling::td/div/input