Recent

Topics

Tutorials

Highlights

Settings

Feedback (http://community.safa

Sign Out

PREV
Uploadi

NEXT
g a t...

Identifying elements based on anchors

In some cases, the same web element is used more than once in a page. For example, suppose that the application uses a toggle button implemented using an image to change the value or the state of other elements, such as input controls (for example, `WebEdit`). Further, let us assume that these elements possess exactly the same set of attributes, and that they have no unique ID or name to reckon upon for unequivocal identification. In such a case, the task of identifying these objects during runtime can be achieved using alternative options given by UFT, such as the index or location of the object. However, this solution is not robust enough, as I discovered in one of the projects in which I was involved.

The problem was found precisely with such toggle buttons. At first, work was done relying on the index (because the page layout was fixed), but soon I discovered that in some cases, QTP clicked on the wrong button (the project was done in 2007-2008 with QTP 9.5). After investigating the issue, it turned out that there was a bug in the application. Though the intended toggle operation was indeed executed, in some cases, the image of the toggled button did not change. As a result, all elements with an index greater than the unchanged element remained with an index greater than expected. To illustrate the problem, suppose that we have 10 such controls in a page, and we click on the third control (index = 2). Now, as we expected the original image collection to decrease by one, reference to index = 2 should have led to click on the **Next Image**

nage was not replaced to reflect the new state
= 2 continued to reference the same element!
ı that control, clicking again and again. There
was also a checkpoint that verified that clicking on the toggle button actually switched the values of two input elements, and of course, these never changed, as the wrong toggle button was clicked!

After facing such a bug (which was reported, of course) a dilemma arose. Should we stop the run session or find a workaround to continue after reporting the problem? The decision was quite easy, because the bug was considered minor. So we opted for the second alternative.

How to do it...

In automation, we usually refrain from identifying TOs based on their location on the screen. Absolute coordinates are really bad identifiers, as the screen resolution can change (and also modern browsers have zooming capabilities). In addition, using *abs_x* and *abs_y* would be sensitive to any tiny layout change. However, if we know how to identify an object based on its name, ID, or inner text, to name a few valid properties, then we can use it as an anchor to identify other related objects in its vicinity. The idea is similar to that of attached text, which was accustomed in older technologies. Actually, a label was identified, and then, the target input control was identified based on a search algorithm for text to its left or top.

The idea was quite simple. First, get the collection of objects to search (in our case, it was a collection of images). Second, get a reference to the target anchor object (input element `WebEdit` by QTP/UFT). Finally, in the collection, find the one object that is closest to the anchor (in our case, they were always aligned vertically, that is, at about the same height, so the proximity was calculated along the *x* axis).

From these lines of reasoning, the following code emerged:

```
Const C_SEARCH_RANGE = 35 'Pixels
```

Settings

Feedback
(http://community.safaribooksonline.com/)

Sign Out

```
Function getObjectByAnchor(oParent, oTargetDesc, oAnchor)
    Dim i, oCollection
    Dim AnchorX, AnchorY, TargetX, TargetY

    'Set the function to return nothing in case of failure
    Set getObjectByAnchor=nothing

    'Get the collection of candidate target objects
    Set oCollection = oParent.ChildObjects(oTargetDesc)

    'Check if the given description yielded an empty collection
    If oCollection.count = 0 Then
        Reporter.ReportEvent micWarning, "getObjectByAnchor", "No object
        Exit function
    End If

    'Get the Anchor's position
    With oAnchor
        AnchorX=.GetROProperty("abs_x")
        AnchorY=.GetROProperty("abs_y")
    End With

    'Search the collection of candidate objects
    For i = 0 To oCollection.count-1
        Set oTarget=oCollection(i)
        'Get the object's position
        With oTarget
            TargetX=.GetROProperty("abs_x")
            TargetY=.GetROProperty("abs_y")
        End With
        'Check if the objects are vertically aligned (along the Y axis)
        If TargetY=AnchorY Then
            'Check if it the objects are close enough (within a range de
            If abs(TargetX-AnchorX) <= C_SEARCH_RANGE Then
                'Return the current candidate object as target
                set getObjectByAnchor=oTarget
                Exit Function
            End If
        End If
    Next
End Function
```

In `Action1`, we would call the `GetObjectByAnchor` function as follows:

```
dim ele, oParent, oTargetDesc, oAnchor

Set oParent=Browser("title:=.*Advanced QTP.*").Page("title:=.+")
Set oTargetDesc=Description.Create()
oTargetDesc("micclass").Value="Image"
oAnchor=oParent.WebEdit("name:=s")
set ele = getObjectByAnchor(oParent, oTargetDesc, oAnchor)
If not ele is nothing Then
    ele.click
else
    reporter.ReportEvent micFail, "Click on Image", "No matching object
    exittest
End If
```

Note

Recall that in VBScript, the value `Nothing` is an object, so it is not enough to check the returned value with `If Not IsObject(ele) Then`. Therefore, we used a less common syntax, namely, `If Not ele Is Nothing Then`.

How it works...

The `GetObjectByAnchor` function accepts the following three arguments:

- `objParent`: This is a reference to the parent object or container of the elements from which we need to find our target element
- `objTargetDesc`: This is a description object carrying the properties and values pairs used to retrieve the collection of candidate target elements
- `objAnchor`: This is a reference to the element to which the target element is expected to be aligned

This function requires the execution of the following steps:

- Get the collection of candidate target objects. If empty, report and exit the function while returning the VBScript value of `Nothing`.
- Get the position of `objAnchor`.
- Loop through the collection of candidate objects and do as follows for each of them:
    - Get the position of `oTarget` (which is assigned the current `oCollection` item `i`).
    - Check if `oTarget` is vertically aligned (along the *y* axis) with respect to `objAnchor`.
    - If it is vertically aligned, then check if, with respect to `objAnchor`, `oTarget` is within the search range we defined in the `C_SEARCH_RANGE` global constant.

- If it's within the range, then `oTarget` is most probably the object we are looking for. So exit the function while returning `oTarget`; if not, continue searching.

In `Action1`, we get the value returned by the function and proceed accordingly. If the value is not `Nothing`, then we click on the element. Otherwise, we report on a failure to find a matching object and stop the test.

PREV
Uploading a file u…

NEXT
Synchronizing a t…