Managing multiple browser windows

In particular cases, we may face a requirement to handle multiple browser windows or tabs. A typical situation would be when clicking on a link or button, which leads to the opening of a page in a pop-up browser window, or in another tab within the same window. This new page might be a standard form, a Terms of Use page, or similar, and usually, this would either close automatically upon completing a data-filling process (as in the case of a form), after reading the document, or approving the terms, for instance. One of the challenges with dynamically created pages, which are generated on the fly, is that we do not wish to clutter our OR with such objects, but rather detect their presence during runtime, perform some checkpoints to verify if the content is correct, and proceed with the test flow (usually by closing the newly opened window first).

In other cases, we may need to test a complex web application with an administrator, client-side GUI and an end user client-side GUI. For instance, we might want to test how changes made by an administrator affect the way users use the application. A typical case would be, for example, one in which a power user makes policy changes and so restricts features to specific groups of users. Another related case example would be when a user needs to be banned. In such cases, one would like to perform some operations on the admin side, and test how they reflect on the end user side. To save time, we may wish to have both GUIs open in separate browser windows.

The suggested method involves using a global dictionary. For background on this topic, please refer to the *Using a global dictionary for fast shared data access* recipe in Chapter 1, *Data-driven Tests*, to learn how to define and use such a dictionary.

How to do it...

The general method to manage multiple browsers is to use an hWnd (window handle). First, we will get an hWnd for each browser window that opens and store it in a dedicated global dictionary that we will declare. Though it changes from one run session to another, the hWnd is always unique for an object, so it is the ultimate identifying property (though it is, of course, a chicken and egg problem, as you first need to identify the object using other properties). Yet, it is a good practice, especially because web applications quite often change the title of the page according to the current context. Basically, UFT identifies the object using the title. Other methods, such as `CreationTime`, are not robust enough, as `CreationTime` is a dynamic property that changes as browser windows open and close. The hWnd property, on the other hand, will remain constant as long as the browser window stays open.

So, while the browser window or tab is open, we will be able to refer to it through its associated key in the dictionary. When closing it, we shall remove the key-item pair from the dictionary. In such a way, we will be able to track the open browsers and access them using a key that reflects their function within the application context, without being sensitive to the content of the currently loaded page.

In the `Web_RegisteredFunctions.vbs` function library, put the following code:

```
Dim oBrowsers

Function initBrowsers()
```

```
    Set oBrowsers = CreateObject("scripting.dictionary")
End Function

Function disposeBrowsers()
    Set oBrowsers = nothing
End Function
```

This code will take care of initializing and disposing of the `objBrowsers` global variable.

In `Action1`, we will execute the following logic for each browser:

1. Open a browser window with a specified URL.
2. Identify the object using the `openurl` property, making sure the URL has opened. We will use a regular expression to suppress specific parameters that may be added automatically to the URL.
3. Retrieve the window handle (the `hWnd` property) and push it to `objBrowsers`.
4. Highlight each browser window by accessing the keys in `oBrowser`.
5. Close each browser window and remove each associated key.

The code is as follows:

```
Dim arrURL

initBrowsers()

arrURL = Array("advancedqtp.com", "taaas.net", "relevantcodes.com")

For i = 0 To ubound(arrURL)
    SystemUtil.Run "IExplore.exe", arrUrl(i)
    If Browser("openurl:=.*"&arrURL(i)&".*").Exist Then
        oBrowsers.Add arrURL(i), Browser("openurl:=.*"&arrURL(i)&".*").Ge
        reporter.ReportEvent micFail, "Open Browser", "Browser didn't ope
    End If
Next

'Show the Browsers
For i = 0 To ubound(arrURL)
    print "hwnd:="&oBrowsers(arrURL(i))
    Browser("hwnd:="&oBrowsers(arrURL(i))).highlight
Next

'Close the Browsers
For i = 0 To ubound(arrURL)
    print "Closing "& arrURL(i)
    Browser("hwnd:="&oBrowsers(arrURL(i))).close
    if not Browser("hwnd:="&oBrowsers(arrURL(i))).Exist(0) then
        oBrowsers.Remove arrURL(i)
        print oBrowsers.count
    End if
Next

disposeBrowsers()
```

A good alternative would be to actually add a reference to the `Browser` object itself. The rest of the logic remains the same. The code is as follows:

```
Dim arrURL

initBrowsers()

arrURL = Array("advancedqtp.com", "taaas.net", "relevantcodes.com")

For i = 0 To ubound(arrURL)
    SystemUtil.Run "IExplore.exe", arrUrl(i)
    If Browser("openurl:=.*"&arrURL(i)&".*").Exist Then
        oBrowsers.Add arrURL(i), Browser("openurl:=.*"&arrURL(i)&".*")
    else
        reporter.ReportEvent micFail, "Open Browser", "Browser didn't ope
    End If
Next

'Show the Browsers
For i = 0 To ubound(arrURL)
    oBrowsers(arrURL(i)).highlight
Next

'Close the Browsers
For i = 0 To ubound(arrURL)
    print "Closing "& arrURL(i)
    oBrowsers(arrURL(i)).close
    if not oBrowsers(arrURL(i)).Exist(0) then
        oBrowsers.Remove arrURL(i)
        print oBrowsers.count
    End if
Next

disposeBrowsers()
```

How it works...

First, we initialize the `objBrowsers` global object, which is actually a dictionary. Next, we open three browser windows using `SystemUtil.Run`, invoking `IExplore.exe` (Internet Explorer's executable) for each URL, as defined in our `arrURL` array variable. For each browser that opens, we store a key with the URL and either assign it `hWnd` or a reference to a `Browser` TO. We then traverse the items in `objBrowsers` with the keys, access each TO using descriptive programming with

`Browser("hwnd:="&objBrowsers(arrURL(i)))`, and highlight them to demonstrate the correct identification. Finally, we close each browser using its `objBrowsers` key, and after verifying that it is closed, we remove the key from `objBrowsers` to keep our list updated.

PREV
Deleting cookies

NEXT
Handling pop-up …