Implementing a simple search class

In this recipe, we will see how to create a class that can be used to execute a search on Google.

Getting ready

From the **File** menu, navigate to **New** | **Test**, and name the new test `SimpleSearch`. Then, create a new function library by navigating to **N** | **Function Library**, or use the *Alt* + *Shift* + *N* shortcut. Name the ne function library `cls.Google.vbs` and associate it with your test.

How to do it...

Proceed with the following steps:

1. Define an environment variable as `OPEN_URL`.
2. Insert the following code in the new library:

```
Class GoogleSearch
  Public Function DoSearch(ByVal sQuery)
    With me.Page_
      .WebEdit("name:=q").Set sQuery
      .WebButton("html id:=gbqfba").Click
    End With
    me.Browser_.Sync

    If me.Results.WaitProperty("visible", 1, 10000) Then
      DoSearch = GetNumResults()
    Else
      DoSearch = 0
      Reporter.ReportEvent micFail, TypeName(Me), "Search did not
    End If
  End Function

  Public Function GetNumResults()
    Dim tmpStr

    tmpStr = me.Results.GetROProperty("innertext")
    tmpStr = Split(tmpStr, " ")
    GetNumResults = CLng(tmpStr(1)) 'Assumes the number is always
  End Function

  Public Property Get Browser_()
    Set Browser_ = Browser(me.Title)
  End Property
  Public Property Get Page_()
    Set Page_ = me.Browser_.Page(me.Title)
  End Property
  Public Property Get Results()
    Set Results = me.Page_.WebElement(me.ResultsId)
  End Property
  Public Property Get ResultsId()
    ResultsId = "html id:=resultStats"
  End Property
  Public Property Get Title()
    Title = "title:=.*Google.*"
  End Property

  Private Sub Class_Initialize
    If Not me.Browser_.Exist(0) Then
      SystemUtil.Run "iexplore.exe", Environment("OPEN_URL")
      Reporter.Filter = rfEnableErrorsOnly
      While Not Browser_.Exist(0)
        Wait 0, 50
      Wend
      Reporter.Filter = rfEnableAll
      Reporter.ReportEvent micDone, TypeName(Me), "Opened browser"
    Else
      Reporter.ReportEvent micDone, TypeName(Me), "Browser was alre
    End If
  End Sub

  Private Sub Class_Terminate
    If me.Browser_.Exist(0) Then
      me.Browser_.Close
      Reporter.Filter = rfEnableErrorsOnly
      While me.Browser_.Exist(0)
```

```
        wait 0, 50
      Wend
      Reporter.Filter = rfEnableAll
      Reporter.ReportEvent micDone, TypeName(Me), "Closed browser"
    End If
  End Sub
End Class
```

3. In `Action`, write the following code:

```
Dim oGoogleSearch
Dim oListResults
Dim oDicSearches
Dim iNumResults
Dim sMaxResults
Dim iMaxResults

'--- Create these objects only in the first iteration
If Not LCase(TypeName(oListResults)) = "arraylist" Then
  Set oListResults = CreateObject("System.Collections.ArrayList")
End If

If Not LCase(TypeName(oDicSearches)) = "Dictionary" Then
  Set oDicSearches = CreateObject("Scripting.Dictionary")
End If

'--- Get a fresh instance of GoogleSearch
Set oGoogleSearch = GetGoogleSearch()

'--- Get search term from the DataTable for each action iteration
sToSearch = DataTable("Query", dtLocalSheet)
iNumResults = oGoogleSearch.DoSearch(sToSearch)

'--- Store the results of the current iteration
'--- Store the number of results
oListResults.Add iNumResults
'--- Store the search term attached to the number of results as key
If Not oDicSearches.Exists(iNumResults) Then
  oDicSearches.Add iNumResults, sToSearch
End If
'Last iteration (assuming we always run on all rows), so perform th

If CInt(Environment("ActionIteration")) = DataTable.LocalSheet.GetR
  'Sort the results ascending
  oListResults.Sort
  'Get the last item which is the largest
  iMaxResults = oListResults.item(oListResults.Count-1)
  'Print to the Output pane for debugging
  Print iMaxResults
  'Get the search text which got the most results
  sMaxResults = oDicSearches(iMaxResults)
  'Report result
  Reporter.ReportEvent micDone, "Max search", sMaxResults & " got "
  'Dispose of the objects used
  Set oListResults = Nothing
  Set oDicSearches = Nothing
  Set oGoogleSearch = Nothing
End If
```

4. In the local datasheet, create a parameter named `Query` and enter several values to be used in the test as search terms.
5. Next, from the UFT home page navigate to **View** | **Test Flow**, and then right-click with the mouse on the Action component in the graphic display, then select **Action Call Properties** and set the Action to run on all rows.

How it works...

The Action takes care to preserve the data collected through the iterations in the array list `oListResults` and the dictionary `oDicSearches`. It checks if it reaches the last iteration after each search is done. Upon reaching the last iteration, it analyzes the data to decide which term yielded the most results. A more detailed description of the workings of the code can be seen as follows.

First, we create an instance of the `GoogleSearch` class, and the `Class_Initialize` subroutine automatically checks if the browser is not already open. If not open, `Class_Initialize` opens it with the `SystemUtil.Run` command and waits until it is open at the web address defined in `Environment("OPEN_URL")`.

The `Title` property always returns the value of the **Descriptive Programming** (**DP**) value required to identify the Google browser and page.

The `Browser_`, `Page_`, and `Results` properties always return a reference to the Google browser, page, and `WebElement` respectively, which hold the text with the search results.

After the browser is open, we retrieve the search term from the local DataTable parameter `Query` and call the `GoogleSearch DoSearch` method with the search term string as parameter. The `DoSearch` method returns a value with the number of results, which are given by the internal method `GetNumResults`.

In the Action, we store the number itself and add to the dictionary, an entry with this number as the key and the search term as the value.

When the last iteration is reached, an analysis of the results is

automatically done by invoking the `Sort` method of `oListResults ArrayList`, getting the last item (the greatest), and then retrieving the search term associated with this number from the dictionary; it reports the result.

At last, we dispose off all the objects used, and then the `Class_Terminate` subroutine automatically checks if the browser is open. If open, then the `Class_Terminate` subroutine closes the browser.