Hi Folks  
  
I am having trouble trying to get VBA to click on an image on a web page.   
The HTML java code is as follows: (sendsmsform is a function)  
  
<!--<input type="button" name="submitsms" value="send your message"   
onClick="sendsmsform();">-->  
    <a href="#" onClick="sendsmsform();"><img src="/messaging/send\_message.jpg"   
alt="send message" border="0" /></a>  
  
My VBA code is as follows:  
  
Private Sub SMS\_STORE\_FIGURES\_TO\_MANAGERS\_2()  
   Dim i As Long  
   Dim IE As Object  
   Dim objElement As Object  
   Dim objCollection As Object  
  
   ' Create InternetExplorer Object  
   Set IE = CreateObject("InternetExplorer.Application")  
  
   ' You can uncoment Next line To see form results  
   IE.Visible = True  
  
   ' Send the form data To URL As POST binary request  
   IE.Navigate "http://intranet/cgi-bin/sms.pl?trig=template"  
  
   ' Statusbar  
   Application.StatusBar = "messaging is loading. Please wait..."  
  
   ' Wait while IE loading...  
     'IE.Navigate2 URL  
     Do While IE.readyState <> 4  
        DoEvents  
     Loop  
          
     Do While IE.Busy = True  
        DoEvents  
     Loop  
  
  
   Application.StatusBar = "Search form submission. Please wait..."  
  
   Set objCollection = IE.Document.getElementsByTagName("input")  
  
   i = 0  
   While i < objCollection.Length  
       If objCollection(i).Name = "Number" Then  
  
           ' Set text for search  
           objCollection(i).Value = "07971411105"  
             
       End If  
       i = i + 1  
   Wend  
  
   Set objCollection = IE.Document.getElementsByTagName("textarea")  
  
   i = 0  
   While i < objCollection.Length  
       If objCollection(i).Name = "Message" Then  
  
           ' Set text for search  
           objCollection(i).Value = "07971411105"  
             
       End If  
       i = i + 1  
   Wend  
     
   Set objCollection = IE.Document.getElementsByTagName("textarea")  
  
   i = 0  
   While i < objCollection.Length  
       If objCollection(i).Name = "Message" Then  
  
           ' Set text for search  
           objCollection(i).Value = "Neil is SOOOO COOOOL"  
             
       End If  
       i = i + 1  
   Wend  
  
   Set objCollection = IE.Document.getElementsByTagName("input")  
  
   i = 0  
   While i < objCollection.Length  
       If objCollection(i).Name = "readit" Then  
  
           ' Set text for search  
           Set objElement = objCollection(i)  
           objElement.Click  
       End If  
       i = i + 1  
   Wend  
         
       Set objCollection = IE.Document.getElementsByTagName("img")  
  
           ' Set text for search  
           Set objElement = objCollection  
  
           objElement.onclick  
  
     
   ' Wait while IE re-loading...  
   Do While IE.Busy  
       Application.Wait DateAdd("s", 1, Now)  
   Loop  
  
   ' Show IE  
   IE.Visible = True  
  
   ' Clean up  
   Set IE = Nothing  
   Set objElement = Nothing  
   Set objCollection = Nothing  
  
   Application.StatusBar = ""  
End Sub  
  
If you need the Full Code for the web page I can add this although its a   
company internal page so you will not be able to view it.

http://www.officekb.com/Uwe/Images/plus.gifSignature

Nelly

[[http://www.officekb.com/Uwe/Images/16/Reply.gif](http://www.officekb.com/Uwe/ForumPost.aspx?article=excel-prog:157216:C72E61A0-A7B8-466C-A0B5-3440EA6160D1@microsoft.com)Reply to this Message](http://www.officekb.com/Uwe/ForumPost.aspx?article=excel-prog:157216:C72E61A0-A7B8-466C-A0B5-3440EA6160D1%40microsoft.com)

[http://www.officekb.com/Uwe/Images/Parent.gif](http://www.officekb.com/Uwe/Forum.aspx/excel-prog/157216/Stuck-on-VBA-Web-Automation#C72E61A0A7B8466CA0B53440EA6160D1microsoftcom)http://www.officekb.com/Uwe/Images/MsgTree.gifhttp://www.officekb.com/Uwe/Images/minus.gif**Joel** - 06 Sep 2009 14:55 GMT

I think the answer "MAY" be simple.  Yo need to select the button  
  
objElement.select  
objElement.onclick  
  
You may have multiple tag items named "img".  
  
You may be able to get the button name with this line  
set obj = IE.Document.all.item("submitsms")  
  
this debug code may help  
  
Private Sub Dump()  
   Dim i As Long  
   Dim IE As Object  
   Dim objElement As Object  
   Dim objCollection As Object  
  
   ' Create InternetExplorer Object  
   Set IE = CreateObject("InternetExplorer.Application")  
  
   ' You can uncoment Next line To see form results  
   IE.Visible = True  
  
   ' Send the form data To URL As POST binary request  
   IE.Navigate "http://intranet/cgi-bin/sms.pl?trig=template"  
  
   ' Statusbar  
   Application.StatusBar = "messaging is loading. Please wait..."  
  
   ' Wait while IE loading...  
     'IE.Navigate2 URL  
     Do While IE.readyState <> 4  
        DoEvents  
     Loop  
          
     Do While IE.Busy = True  
        DoEvents  
     Loop  
  
        
     RowCount = 1    
     for each itm in IE.Document.all  
        Range("A" & RowCount) = itm.tagname  
        Range("B" & RowCount) = itm.classname  
        Range("C" & RowCount) = itm.id       'comment out if errors  
        Range("D" & RowCount) = itm.name     'comment out if errors  
        Range("E" & RowCount) = left(itm.innertext,1024)  
        RowCount = Rowcount + 1  
     next itm  
  
end Sub

*> Hi Folks  
>   
[quoted text clipped - 117 lines]  
> If you need the Full Code for the web page I can add this although its a   
> company internal page so you will not be able to view it.*

[[http://www.officekb.com/Uwe/Images/16/Reply.gif](http://www.officekb.com/Uwe/ForumPost.aspx?article=excel-prog:157216:36888A88-DDF5-4348-A0B2-A42382361407@microsoft.com)Reply to this Message](http://www.officekb.com/Uwe/ForumPost.aspx?article=excel-prog:157216:36888A88-DDF5-4348-A0B2-A42382361407%40microsoft.com)

[http://www.officekb.com/Uwe/Images/Parent.gif](http://www.officekb.com/Uwe/Forum.aspx/excel-prog/157216/Stuck-on-VBA-Web-Automation#36888A88DDF54348A0B2A42382361407microsoftcom)http://www.officekb.com/Uwe/Images/MsgTree.gifhttp://www.officekb.com/Uwe/Images/minus.gif**nelly** - 06 Sep 2009 15:04 GMT

Hi Joel  
  
set obj = IE.Document.all.item("submitsms") - obj = Nothing  
  
A runtime error occurs on objElement.select  
  
Regards

http://www.officekb.com/Uwe/Images/plus.gifSignature

Nelly

*> I think the answer "MAY" be simple.  Yo need to select the button  
>   
[quoted text clipped - 170 lines]  
> > If you need the Full Code for the web page I can add this although its a   
> > company internal page so you will not be able to view it.*

[[http://www.officekb.com/Uwe/Images/16/Reply.gif](http://www.officekb.com/Uwe/ForumPost.aspx?article=excel-prog:157216:FA996883-AF3D-4B42-A9C7-D4B4D158221B@microsoft.com)Reply to this Message](http://www.officekb.com/Uwe/ForumPost.aspx?article=excel-prog:157216:FA996883-AF3D-4B42-A9C7-D4B4D158221B%40microsoft.com)

[http://www.officekb.com/Uwe/Images/Parent.gif](http://www.officekb.com/Uwe/Forum.aspx/excel-prog/157216/Stuck-on-VBA-Web-Automation#FA996883AF3D4B42A9C7D4B4D158221Bmicrosoftcom)http://www.officekb.com/Uwe/Images/MsgTree.gifhttp://www.officekb.com/Uwe/Images/minus.gif**Joel** - 06 Sep 2009 17:52 GMT

The all method I jsut learned about and haven';t fully learned how to use.    
It seem to find object listed as ID= but not on tags.  If yo are getting an   
error on the select then you are not on an object the has an ONClick property.  
  
You need to run the dump program I posted and post the rows from the excel   
sheet near the object you are trying to find.  
  
I sometime have to use code like this to stop .  then set a wat item to   
"ITM" and look through the property until you find an item with an ONCLICK   
property.  It is sometimes one ro two objects away from the item that yo are   
looking for.  they also may be children of the object you are looking for.  I   
sometime get very frustrated trying to get code like this working, but I   
always find a way.  
  
Private Sub DSopp()  
   Dim i As Long  
   Dim IE As Object  
   Dim objElement As Object  
   Dim objCollection As Object  
  
   ' Create InternetExplorer Object  
   Set IE = CreateObject("InternetExplorer.Application")  
  
   ' You can uncoment Next line To see form results  
   IE.Visible = True  
  
   ' Send the form data To URL As POST binary request  
   IE.Navigate "http://intranet/cgi-bin/sms.pl?trig=template"  
  
   ' Statusbar  
   Application.StatusBar = "messaging is loading. Please wait..."  
  
   ' Wait while IE loading...  
     'IE.Navigate2 URL  
     Do While IE.readyState <> 4  
        DoEvents  
     Loop  
          
     Do While IE.Busy = True  
        DoEvents  
     Loop  
  
        
     RowCount = 1  
     For Each itm In IE.Document.all  
        If RowCount = 135 Then Stop  
          
        RowCount = RowCount + 1  
     Next itm  
  
End Sub

*> Hi Joel  
>   
[quoted text clipped - 178 lines]  
> > > If you need the Full Code for the web page I can add this although its a   
> > > company internal page so you will not be able to view it.*

[[http://www.officekb.com/Uwe/Images/16/Reply.gif](http://www.officekb.com/Uwe/ForumPost.aspx?article=excel-prog:157216:1FB2614A-E311-457F-8FFA-DD52B12AA934@microsoft.com)Reply to this Message](http://www.officekb.com/Uwe/ForumPost.aspx?article=excel-prog:157216:1FB2614A-E311-457F-8FFA-DD52B12AA934%40microsoft.com)

[http://www.officekb.com/Uwe/Images/Parent.gif](http://www.officekb.com/Uwe/Forum.aspx/excel-prog/157216/Stuck-on-VBA-Web-Automation#1FB2614AE311457F8FFADD52B12AA934microsoftcom)http://www.officekb.com/Uwe/Images/MsgTree.gifhttp://www.officekb.com/Uwe/Images/minus.gif**nelly** - 06 Sep 2009 22:29 GMT

Hi Jeol  
  
Many thanks for your ideas. Through trial and error from the the info you   
gave I got this to work by the following.  
  
   Set obj = IE.Document.all.Item("smsform")  
  
   obj.submit  
  
Thanks Again

------------------------------------------neu

[Code For Excel And Outlook - Learn VBA](http://www.codeforexcelandoutlook.com/)

Automation and VBA code for Microsoft® Excel and Outlook

You are here: [Home](http://www.codeforexcelandoutlook.com/) / [Excel VBA](http://www.codeforexcelandoutlook.com/excel-vba/) / Automate Internet Explorer

Here I will demonstrate some example code for automating Internet Explorer through VBA. You can open web pages, click buttons, import data, etc. The possibilities are endless.  
  
These examples use [late binding](http://www.codeforexcelandoutlook.com/excel-vba/early-late-binding/) with the following two object libraries: **Microsoft Internet Controls (shdocvw.dll)** and **Microsoft HTML Object Library (MSHTML.TLB)**.

There are also some samples for getting data from the web using the [XMLHTTP](http://www.codeforexcelandoutlook.com/blog/tag/xmlhttp/) object through different web APIs on my blog.

## Open a webpage and display (simple)

Sub GoToWebSite()  
Dim appIE As Object ' InternetExplorer  
Dim sURL As String  
Application.ScreenUpdating = False  
  
Set appIE = CreateObject("InternetExplorer.Application")  
  
sURL = "http://www.codeforexcelandoutlook.com"  
  
With appIE  
    .Navigate sURL  
    .Visible = True  
End With  
  
Application.ScreenUpdating = True  
  
Set appIE = Nothing  
End Sub

## Open a webpage, fill in form fields and click buttons

Sub GoToWebSiteAndPlayAround()  
  
Dim appIE As Object ' InternetExplorer.Application  
Dim sURL As String  
Dim UserN As Variant, PW As Variant  
Dim Element As Object ' HTMLButtonElement  
Dim btnInput As Object ' MSHTML.HTMLInputElement  
Dim ElementCol As Object ' MSHTML.IHTMLElementCollection  
Dim Link As Object ' MSHTML.HTMLAnchorElement  
Dim strCountBody As String  
Dim lStartPos As Long  
Dim lEndPos As Long  
Dim TextIWant As String  
  
Application.ScreenUpdating = False  
  
Set appIE = CreateObject("InternetExplorer.Application")  
  
sURL = "http://www.codeforexcelandoutlook.com"  
  
With appIE  
    .Navigate sURL  
    ' uncomment the line below if you want to watch the code execute, or for debugging  
   '.Visible = True  
End With  
  
' loop until the page finishes loading  
Do While appIE.Busy  
Loop  
  
' enter username and password in textboxes  
Set UserN = appIE.Document.getElementsByName("username")  
If Not UserN Is Nothing Then  
    UserN(0).Value = "login name"  
End If  
  
Set PW = appIE.Document.getElementsByName("password")  
' password  
If Not PW Is Nothing Then  
    PW(0).Value = "my Password"  
End If  
  
' click 'Submit' button  
Set ElementCol = appIE.Document.getElementsByTagName("INPUT")  
  
For Each btnInput In ElementCol  
    If btnInput.Value = "Submit" Then  
        btnInput.Click  
        Exit For  
    End If  
Next btnInput  
  
' loop until the page finishes loading  
Do While appIE.Busy  
Loop  
  
' click a button on the next page  
Set ElementCol = appIE.Document.getElementsByTagName("INPUT")  
  
For Each btnInput In ElementCol  
    If btnInput.Value = "Link Page #1" Then  
        btnInput.Click  
        Exit For  
    End If  
Next btnInput  
  
' loop until the page finishes loading  
Do While appIE.Busy  
Loop  
  
' click a text link on the page after that  
Set ElementCol = appIE.Document.getElementsByTagName("a")  
  
For Each Link In ElementCol  
    If Link.innerHTML = "<B>Clickable Text Link Name</B>" Then  
        Link.Click  
        Exit For  
    End If  
Next Link  
  
' loop until the page finishes loading  
Do While appIE.Busy  
  Do Events  
Loop  
  
' grab some text from the body  
strCountBody = appIE.Document.body.innerText  
lStartPos = InStr(1, strCountBody, "Text to find")  
lEndPos = lStartPos + 12  
TextIWant = Mid$(strCountBody, lStartPos, lEndPos - lStartPos)  
  
' grab the whole screen & paste into Excel  
appIE.ExecWB OLECMDID\_SELECTALL, OLECMDEXECOPT\_DONTPROMPTUSER  
appIE.ExecWB OLECMDID\_COPY, OLECMDEXECOPT\_DODEFAULT  
  
Workbooks.Add  
ActiveSheet.Paste  
  
Application.ScreenUpdating = True  
appIE.Quit  
End Sub

I'll admit this isn't the best example; there's no login form on my homepage. But let's examine these samples closely anyway. The first one simply starts an late-bound instance of Internet Explorer, assigns a url (string) to a variable and then passes it to the Internet Explorer application variable. Then it makes the window visible, destroys the variable and exits.

The second example shows in much more detail how to manipulate Internet Explorer.

Use **Do While appIE.Busy: DoEvents: Loop** to keep Excel waiting until the page loads completely.

Textboxes (username, password, etc) that you fill in are usually named elements on a webpage. The code uses

**appIE.Document.getElementsByName("username")**

to find the textbox named "username" and put your chosen text there. To find the name of the text box, click on View>Source in Internet Explorer when viewing the page. You will be using View>Source extensively to get the names of various elements on each page you want to access.

To click buttons on a site, you can cycle through each Input Element of the Elements Collection until you find the button named "Submit":

**appIE.Document.getElementsByTagName("INPUT")**

Again, you would need to View » Source to see the tag name is "INPUT" (the type of element) and the name of the button is "Submit". The Click Method actually does the clicking of the button to load the next page or input whatever you placed in the text boxes.

To click on a link on a page, you would loop through the tags that start with 'a' (links) until you find the one with the exact name you are looking for. You need to include everything between the <a> and </a> tags; so in the code above, the webmaster put 'bold' tags in the link (). That would need to be included because technically it is part of the text of the link (but bad web design if you ask me).

**appIE.Document.getElementsByTagName("a")**

Now to grab the body text (the real reason you are here, I'm sure), assign the innerText property to a String variable:

**appIE.Document.body.innerText**

In the sample above, I use the **Instr** property to look for some text, then the **Mid** function to extract it from the webpage. At that point you could simply paste it into a workbook (see [Latitude and Longitude functions](http://www.codeforexcelandoutlook.com/excel-vba/latitude-longitude-functions/) for the GetDistance() UDF) but I went on and just copied and pasted the entire displayed webpage into a new workbook.

Hopefully these samples will help you automate Internet Explorer in your own projects.

## More Examples

For even more examples, check out:

* [An exploration of IE browser methods, part III](http://www.codeforexcelandoutlook.com/blog/2010/08/an-exploration-of-ie-browser-methods-part-iii/)
* [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/)
* [An exploration of IE browser methods, part I](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/)
* [Search Engine Browsing](http://www.codeforexcelandoutlook.com/excel-vba/search-engine-browsing/)
* [Screen Scraping 101 with VBA](http://www.codeforexcelandoutlook.com/blog/2010/08/screen-scraping-101-with-vba/)

# An exploration of IE browser methods, part I

June 1, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [3 Comments](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/#comments)



It's been a while since I posted anything related to [Internet Explorer](http://www.codeforexcelandoutlook.com/blog/category/internet-explorer/), but here is a series of VBA functions that let you automate Internet Explorer's native methods.

I've received many questions about the code I posted at [Automate Internet Explorer](http://www.codeforexcelandoutlook.com/excel-vba/automate-internet-explorer/), so it's about time it was updated and refreshed with better coding techniques. This will be the first in a short series of articles about automating Internet Explorer.

We're all familiar (I hope) with the late-bound method for instantiating a copy of IE. Here is a function that returns the InternetExplorer object:

Function GetIE() As Object  
  On Error Resume Next  
  Set GetIE = CreateObject("InternetExplorer.Application")  
End Function

Call the function like this:

Dim ie As Object  
Set ie = GetIE

You will notice that it is relatively slow compared to the [XMLHTTP Object](http://www.codeforexcelandoutlook.com/blog/tag/xmlhttp/) that we've seen previously. The only time I recommend using it is when you need to actually interact with a website in order to retrieve information. Those are the bad websites that need to be shut down! Seriously, people, set up an [API](http://www.codeforexcelandoutlook.com/blog/tag/api/).

The MSDN entry for the [InternetExplorer Object](http://msdn.microsoft.com/en-us/library/aa752084%28VS.85%29.aspx) gives you all the methods, events and properties you'll need to make IE dance for you.

The following is a series of functions that encapsulate most of the methods and properties. For most/all of these functions, you will need to pass in an InternetExplorer Object. I didn't do a type check but if you want to add one, check that TypeName(obj) = "IWebBrowser2". That's what the IE Object was referred to on my machine (YMMV).

If you want an example of class-driven events, see [VBA web services](http://www.tushar-mehta.com/publish_train/xl_vba_cases/vba_web_pages_services/index.htm#_Toc173749359).

## AddressBar Property

Check if the Address Bar is visible using this function. If you pass True or False as the second parameter, you can change the property value.

Function ShowAddressBar(inet As Object, Optional addrBar As Variant) As Boolean  
  
  If Not IsMissing(ShowAddressBar) Then  
    If CBool(ShowAddressBar) Then  
      inet.AddressBar = ShowAddressBar  
    End If  
  End If  
  
  ShowAddressBar = inet.AddressBar  
  
End Function

## Busy Property

Check if your IE object is busy loading a page by passing it to this function.

Function IsIEBusy(inet As Object) As Boolean  
  IsIEBusy = inet.Busy  
End Function

## FullScreen Property

This function lets you get or set the FullScreen Property of IE. Personally I hate IE in full-screen.

Function IEFullScreen(inet As Object, \_  
    Optional displayFullScreen As Variant) As Boolean  
  
  If Not IsMissing(displayFullScreen) Then  
    If CBool(displayFullScreen) Then  
      inet.FullScreen = displayFullScreen  
    End If  
  End If  
  
  IEFullScreen = inet.FullScreen  
  
End Function

## LocationName Property

This property returns the title bar caption from the IE window.

Function GetIELocation(inet As Object) As String  
  GetIELocation = inet.locationName  
End Function

## LocationURL Property

If you want the current URL, this is the function to use.

Function GetIEURL(inet As Object) As String  
  GetIEURL = inet.LocationURL  
End Function

## MenuBar Property

Set or get the MenuBar Property that determines whether the IE menu bar is visible.

Function DisplayIEMenuBar(inet As Object, \_  
    Optional displayMenu As Variant) As Boolean  
  
  If Not IsMissing(displayMenu) Then  
    If CBool(displayMenu) Then  
      inet.MenuBar = displayMenu  
    End If  
  End If  
  
  DisplayIEMenuBar = inet.MenuBar  
  
End Function

## ReadyState Property

The ReadyState Property works similar to the Busy Property. It gives you the state of the IE object. I've also provided the Enum so you can see all the possible return values.

Function GetIEReadyState(inet As Object) As Integer  
  GetIEReadyState = inet.READYSTATE  
End Function  
  
'Enum READYSTATE  
'    READYSTATE\_UNINITIALIZED = 0  
'    READYSTATE\_LOADING = 1  
'    READYSTATE\_LOADED = 2  
'    READYSTATE\_INTERACTIVE = 3  
'    READYSTATE\_COMPLETE = 4  
'End Enum

Most of the time you'll be checking for the number 4 (complete) using a **Do While GetIEReadyState(ie) <> 4** type of statement.

## Resizable Property

If you want your IE window to stay a certain size, you'll want to set the Resizable Property to **False**. Here's one way to do it.

Function IEResizable(inet As Object, Optional resize As Variant) As Boolean  
  
  If Not IsMissing(resize) Then  
    If CBool(resize) Then  
      inet.Resizable = resize  
    End If  
  End If  
  
  IEResizable = inet.Resizable  
  
End Function

If you pass in True or False as the second parameter, you can adjust the value. Either way, the function returns the current value.

## Silent Property

This property would be useful if you are expecting to navigate to a site with popups or dialog boxes you don't want to see. Especially those script debugging boxes that seem to appear with IE.

Function IESilent(inet As Object, Optional keepSilent As Variant) As Boolean  
  
  If Not IsMissing(keepSilent) Then  
    If CBool(keepSilent) Then  
      inet.Silent = keepSilent  
    End If  
  End If  
  
  IESilent = inet.Silent  
  
End Function

## StatusBar Property

Hide or show the status bar for the current IE window with this function.

Function IEStatusBar(inet As Object, Optional statusBar As Variant) As Boolean  
  
  If Not IsMissing(statusBar) Then  
    If CBool(statusBar) Then  
      inet.StatusBar = statusBar  
    End If  
  End If  
  
  IEStatusBar = inet.statusBar  
  
End Function

## StatusText Property

Set or get the status bar text with this function. You'll want to make sure the [status bar](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/#statusbar) is visible first.

Function IEStatusBarText(inet As Object, \_  
    Optional statusBarText As String) As String  
  
  If Len(statusBarText) > 0 Then  
    inet.StatusText = statusBarText  
  End If  
  
  IEStatusBarText = inet.statusText  
  
End Function

## TheaterMode Property

This function will put IE into Theater mode. That's where all the toolbars disappear and IE is maximized on the desktop.

Function IETheaterMode(inet As Object, \_  
    Optional isTheaterMode As Variant) As Boolean  
  
  If Not IsMissing(isTheaterMode) Then  
    If CBool(isTheaterMode) Then  
      inet.TheaterMode = isTheaterMode  
    End If  
  End If  
  
  IETheaterMode = inet.TheaterMode  
  
End Function

## Visible Property

Make the IE window visible with this function. When you instantiate IE, the window remains invisible. According to MSDN,

When the Windows Internet Explorer application is first created, it is hidden. It becomes visible after the Navigate method or the GoSearch method is used.

In my experience, it remains invisible until you set the Visible Property to **True**.

Function IEVisible(inet As Object, Optional isVisible As Variant) As Boolean  
  
  If Not IsMissing(isVisible) Then  
    If CBool(isVisible) Then  
      inet.Visible = isVisible  
    End If  
  End If  
  
  IEVisible = inet.Visible  
  
End Function

## Navigation Functions

These functions, respectively, allow you to tell IE to go **back** one page (according to History), go **forward** one page (according to History), or go to the Home page (as defined in Internet Options).

Function GoBack(inet As Object)  
  inet.GoBack  
End Function  
  
Function GoForward(inet As Object)  
  inet.GoForward  
End Function  
  
Function GoHome(inet As Object)  
  inet.GoHome  
End Function

## Navigate Method

Here is one of the more important methods — the one that lets you browse to a particular web page, and even submit POST data. Visit the [Navigate Method](http://msdn.microsoft.com/en-us/library/aa752093%28v=VS.85%29.aspx) for more details about each parameter. Usually I only use the URL parameter.

Function Navigate(inet As Object, url As String, flags As Variant, targetFrameName As Variant, \_  
postData As Variant, headers As Variant)  
inet.Navigate url, flags, targetFrameName, postData, headers  
End Function

## Helper Functions

To close the IE instance, use the Quit Method.

Function IEQuit(inet As Object)  
  inet.Quit  
End Function

Refresh the current page using the Refresh Method.

Function IERefresh(inet As Object)  
  inet.Refresh  
End Function

To stop a page from loading, use the Stop Method.

Function IEStop(inet As Object)  
  inet.Stop  
End Function

Next time we'll go through the [HTMLDocument Object](http://msdn.microsoft.com/en-us/library/ms531073%28v=VS.85%29.aspx) and, as a practical example, use the HTML Object Model to do something useful (extracting the UV Index from the U.S. EPA's [SunWise](http://www.epa.gov/sunwise/uvindex.html) website for a given zip code).

You are here: [Home](http://www.codeforexcelandoutlook.com/) / [Internet Explorer](http://www.codeforexcelandoutlook.com/blog/category/internet-explorer/) / An exploration of IE browser methods, part II

# An exploration of IE browser methods, part II

June 8, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [13 Comments](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/#comments)

In a [previous post](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/) I promised to go through a real example of how we can [automate Internet Explorer](http://www.codeforexcelandoutlook.com/excel-vba/automate-internet-explorer/) to retrieve data from a website. We'll also enumerate some of the members of the IE Document Object as well as see how we can incorporate XMLHTTP into the mix.

The EPA's SunWise Program publishes a [UV Index](http://www.epa.gov/sunwise/uvindex.html), which is the amount of UV exposure you can expect for the day. Their website has some neat graphs of how the UV Index works, and what each value means. I'm obliged to remind you that you should cover up in the sun, wear at least SPF 15 sunblock, and remember that just because it isn't sunny out doesn't mean you're not getting exposed to UV rays.

You can [lookup the UV Index](http://www.epa.gov/sunwise/uvindex.html#lookup) directly on their site. But we don't want to do that. We want to extract the UV index programmatically from the EPA's website.

The first thing we do is look at the source of the page (View > Source in Internet Explorer, Ctrl+U in FireFox) and see that the UV Index lookup is wrapped in HTML **form** tags. Wonderful! Here is the code.

<form action="http://oaspub.epa.gov/enviro/uv\_search" method="post" id="uviform" onsubmit="return validateInput();">  
  <table border="0" cellpadding="1" cellspacing="0" summary="The UV Index query form." style="width:auto; margin-right:60px;">  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label">ZIP Code:</label></td>  
      <td><input name="zipcode" type="text" id="zipcode" size="5" maxlength="5" /></td>  
    </tr>  
    <tr>  
      <td>&nbsp;</td>  
      <td style="padding:0.7em 0;">or</td>  
    </tr>  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label2">City:</label></td>  
      <td><input id="city\_name" type="text" name="city\_name" size="30" /></td>  
    </tr>  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label3">State:</label></td>  
      <td><input name="state\_code" type="text" id="state\_code" title="Postal abbreviation" size="2" maxlength="2" /></td>  
    </tr>  
    <tr>  
      <td>&nbsp;</td>  
      <td style="padding-top:20px;"><input type="submit" id="submit" style="font-weight:bold;" value="Get UV Index" />  
        <input type="reset" id="reset" style="font-size:0.9em;" value="Reset" /></td>  
    </tr>  
  </table>  
</form>

This markup tells me:

* The data you enter into this form is submitted to a URL for processing.
* The base URL for my query is "http://oaspub.epa.gov/enviro/uv\_search".
* The form takes three inputs, named **zipcode**, **city\_name** and **state\_code**.

How did I figure that out?

* The ["action" attribute](http://www.w3schools.com/html/html_forms.asp) of the form tag gives you the base URL for any query using that form's data. So any IE navigation or XMLHTTP query I want must begin with "http://oaspub.epa.gov/enviro/uv\_search".
* There are three input fields for this particular form. Those are the fields that will be sent to the URL in the action attribute of the form tag. The name attribute for each one gives you the parameters you'll need to add to the query.

Any HTTP GET request must be formatted like this:

**base URL?first\_parameter=value&second\_parameter=value&third\_parameter=value**

In other words, to start the parameter list, add a question mark to the end of the base URL. Each subsequent parameter is prepended with an ampersand ('**&**') and added to the end of the parameter list. Here's a great example: [HTTP Client Methods – GET and POST](http://www.webdevelopersnotes.com/basics/http_client_methods_get_post.php3)

Since we can submit the form using only zip code, the URL for the query would look like this:

**http://oaspub.epa.gov/enviro/uv\_search?zipcode=11103**

(It would actually be ***http://oaspub.epa.gov/enviro/uv\_search?zipcode=11103&city\_name=&state\_code=*** but we can leave off the other parameters since they are blank anyway)

If you enter either URL into your web browser, and view the source of the resulting page, the tag we're looking for looks like this:

<img src="//www.epa.gov/enviro/gif/C\_UV08.gif" alt="UVI 8" border=0>

This is the graphic that displays the UV Index. The index number itself is in the alt attribute of the IMG tag. So now we'll need to move to the HTML Object Model to parse the HTML from this page and extract the UV Index number.

## The easy way

We'll use a simple trick: use the XMLHTTP object to retrieve the result of the query, instead of instantiating Internet Explorer. This will make our code run much faster, since we figured out how to get our result without having to actually visit the site.

**No early bound references required!**

Function GetUVIndex(zipCode As String) As String  
  
Dim xml As Object ' MSXML2.XMLHTTP  
Dim html As Object ' MSHTML.HTMLDocument  
Dim imgLinks As Object ' MSHTML.IHTMLElementCollection  
Dim imgLink As Object ' MSHTML.IHTMLElement  
Dim result As String  
  
  Set xml = CreateObject("MSXML2.XMLHTTP.6.0")  
  
  With xml  
    .Open "GET", "http://oaspub.epa.gov/enviro/uv\_search?zipcode=" & zipCode, False  
    .Send  
  End With  
  
  result = xml.responseText  
  
  ' create HTML document  
 Set html = CreateObject("htmlfile") ' New MSHTML.HTMLDocument  
  
  html.body.innerHTML = result  
  
  Set imgLinks = html.getElementsByTagName("img")  
  
  For Each imgLink In imgLinks  
    If Left$(imgLink.getAttribute("alt"), 3) = "UVI" Then  
      GetUVIndex = Right$(imgLink.getAttribute("alt"), \_  
          Len(imgLink.getAttribute("alt")) - InStrRev(imgLink.getAttribute("alt"), " "))  
      Exit Function  
    End If  
  Next imgLink  
  
End Function

You are here: [Home](http://www.codeforexcelandoutlook.com/) / [Internet Explorer](http://www.codeforexcelandoutlook.com/blog/category/internet-explorer/) / An exploration of IE browser methods, part II

# An exploration of IE browser methods, part II

June 8, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [13 Comments](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/#comments)

In a [previous post](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/) I promised to go through a real example of how we can [automate Internet Explorer](http://www.codeforexcelandoutlook.com/excel-vba/automate-internet-explorer/) to retrieve data from a website. We'll also enumerate some of the members of the IE Document Object as well as see how we can incorporate XMLHTTP into the mix.

The EPA's SunWise Program publishes a [UV Index](http://www.epa.gov/sunwise/uvindex.html), which is the amount of UV exposure you can expect for the day. Their website has some neat graphs of how the UV Index works, and what each value means. I'm obliged to remind you that you should cover up in the sun, wear at least SPF 15 sunblock, and remember that just because it isn't sunny out doesn't mean you're not getting exposed to UV rays.

You can [lookup the UV Index](http://www.epa.gov/sunwise/uvindex.html#lookup) directly on their site. But we don't want to do that. We want to extract the UV index programmatically from the EPA's website.

The first thing we do is look at the source of the page (View > Source in Internet Explorer, Ctrl+U in FireFox) and see that the UV Index lookup is wrapped in HTML **form** tags. Wonderful! Here is the code.

<form action="http://oaspub.epa.gov/enviro/uv\_search" method="post" id="uviform" onsubmit="return validateInput();">  
  <table border="0" cellpadding="1" cellspacing="0" summary="The UV Index query form." style="width:auto; margin-right:60px;">  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label">ZIP Code:</label></td>  
      <td><input name="zipcode" type="text" id="zipcode" size="5" maxlength="5" /></td>  
    </tr>  
    <tr>  
      <td>&nbsp;</td>  
      <td style="padding:0.7em 0;">or</td>  
    </tr>  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label2">City:</label></td>  
      <td><input id="city\_name" type="text" name="city\_name" size="30" /></td>  
    </tr>  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label3">State:</label></td>  
      <td><input name="state\_code" type="text" id="state\_code" title="Postal abbreviation" size="2" maxlength="2" /></td>  
    </tr>  
    <tr>  
      <td>&nbsp;</td>  
      <td style="padding-top:20px;"><input type="submit" id="submit" style="font-weight:bold;" value="Get UV Index" />  
        <input type="reset" id="reset" style="font-size:0.9em;" value="Reset" /></td>  
    </tr>  
  </table>  
</form>

This markup tells me:

* The data you enter into this form is submitted to a URL for processing.
* The base URL for my query is "http://oaspub.epa.gov/enviro/uv\_search".
* The form takes three inputs, named **zipcode**, **city\_name** and **state\_code**.

How did I figure that out?

* The ["action" attribute](http://www.w3schools.com/html/html_forms.asp) of the form tag gives you the base URL for any query using that form's data. So any IE navigation or XMLHTTP query I want must begin with "http://oaspub.epa.gov/enviro/uv\_search".
* There are three input fields for this particular form. Those are the fields that will be sent to the URL in the action attribute of the form tag. The name attribute for each one gives you the parameters you'll need to add to the query.

Any HTTP GET request must be formatted like this:

**base URL?first\_parameter=value&second\_parameter=value&third\_parameter=value**

In other words, to start the parameter list, add a question mark to the end of the base URL. Each subsequent parameter is prepended with an ampersand ('**&**') and added to the end of the parameter list. Here's a great example: [HTTP Client Methods – GET and POST](http://www.webdevelopersnotes.com/basics/http_client_methods_get_post.php3)

Since we can submit the form using only zip code, the URL for the query would look like this:

**http://oaspub.epa.gov/enviro/uv\_search?zipcode=11103**

(It would actually be ***http://oaspub.epa.gov/enviro/uv\_search?zipcode=11103&city\_name=&state\_code=*** but we can leave off the other parameters since they are blank anyway)

If you enter either URL into your web browser, and view the source of the resulting page, the tag we're looking for looks like this:

<img src="//www.epa.gov/enviro/gif/C\_UV08.gif" alt="UVI 8" border=0>

This is the graphic that displays the UV Index. The index number itself is in the alt attribute of the IMG tag. So now we'll need to move to the HTML Object Model to parse the HTML from this page and extract the UV Index number.

## The easy way

We'll use a simple trick: use the XMLHTTP object to retrieve the result of the query, instead of instantiating Internet Explorer. This will make our code run much faster, since we figured out how to get our result without having to actually visit the site.

**No early bound references required!**

Function GetUVIndex(zipCode As String) As String  
  
Dim xml As Object ' MSXML2.XMLHTTP  
Dim html As Object ' MSHTML.HTMLDocument  
Dim imgLinks As Object ' MSHTML.IHTMLElementCollection  
Dim imgLink As Object ' MSHTML.IHTMLElement  
Dim result As String  
  
  Set xml = CreateObject("MSXML2.XMLHTTP.6.0")  
  
  With xml  
    .Open "GET", "http://oaspub.epa.gov/enviro/uv\_search?zipcode=" & zipCode, False  
    .Send  
  End With  
  
  result = xml.responseText  
  
  ' create HTML document  
 Set html = CreateObject("htmlfile") ' New MSHTML.HTMLDocument  
  
  html.body.innerHTML = result  
  
  Set imgLinks = html.getElementsByTagName("img")  
  
  For Each imgLink In imgLinks  
    If Left$(imgLink.getAttribute("alt"), 3) = "UVI" Then  
      GetUVIndex = Right$(imgLink.getAttribute("alt"), \_  
          Len(imgLink.getAttribute("alt")) - InStrRev(imgLink.getAttribute("alt"), " "))  
      Exit Function  
    End If  
  Next imgLink  
  
End Function

### Sample usage

Sub TestGetUVIndex()  
  
Debug.Print GetUVIndex("11103")  
  
End Sub

We use XMLHTTP to run the query, but instead of using the MSXML Object Model, we use the HTML Object Model because the result is HTML, **not** XML.

After grabbing the result page, we put it into a HTMLDocument Object. The **getElementsByTagName** Method returns a collection of all the elements with the given tag name (for example: **input**, **a**, **img**). The **getAttribute** Method will return the given attribute name for each tag element. Then it's a simple matter of looking for the correct tag information.

We're looking for the IMG tag that contains the UV Index in its alt attribute, so we use **html.getElementsByTagName("img")** and then loop through the resulting collection.

## The hard way

Let's explore how we can do this using only IE automation (no XMLHTTP).

I call this "the hard way" because you will see how much more is required of this method.

You'll need to study the source code of the first webpage much more closely, to figure out the field names you want to fill out along with the appropriate methods for each. We didn't need to do that when we pointed XMLHTTP directly at the endpoint URL.

Here's the function and then we'll discuss further.

Function GetUVIndexWithIE(zipCode As String) As String  
  
Dim ie As Object ' InternetExplorer  
Dim html As Object ' MSHTML.HTMLDocument  
Dim imgLinks As Object ' MSHTML.IHTMLElementCollection  
Dim imgLink As Object ' MSHTML.IHTMLElement  
Dim inputCollection As Object ' MSHTML.IHTMLElementCollection  
Dim inputElement As Object ' MSHTML.IHTMLElement  
Dim result As String  
  
  Set ie = CreateObject("InternetExplorer.Application")  
  
  ' go to target page  
 ie.Navigate "http://www.epa.gov/sunwise/uvindex.html"  
  'ie.Visible = True  
  
  ' loop until page is loaded  
 Do Until ie.READYSTATE = 4  ' READYSTATE\_COMPLETE  
   DoEvents  
  Loop  
  
  ' enter zip code  
 ' there's only one page element with id="zipcode", so we  
 ' can enter its value directly  
 ie.document.getElementById("zipcode").value = zipCode  
  
  ' click submit button  
 ' there are multiple input elements named "submit", so we  
 ' need to loop  
 Set inputCollection = ie.document.getElementsByTagName("input")  
  
  For Each inputElement In inputCollection  
    If inputElement.getAttribute("value") = "Get UV Index" Then  
      inputElement.Click  
      Exit For  
    End If  
  Next inputElement  
  
  ' loop until page is loaded  
 Do Until ie.READYSTATE = 4  ' READYSTATE\_COMPLETE  
   DoEvents  
  Loop  
  
  ' grab screenshot  
 result = ie.document.body.innerHTML  
  
  ' close IE  
 ie.Quit  
  
  ' create HTML document  
 Set html = CreateObject("htmlfile") ' New MSHTML.HTMLDocument  
  
  html.body.innerHTML = result  
  
  ' loop through IMG tags  
 Set imgLinks = html.getElementsByTagName("img")  
  
  For Each imgLink In imgLinks  
    If Left$(imgLink.getAttribute("alt"), 3) = "UVI" Then  
      GetUVIndexWithIE = Right$(imgLink.getAttribute("alt"), \_  
                Len(imgLink.getAttribute("alt")) - InStrRev(imgLink.getAttribute("alt"), " "))  
      Exit Function  
    End If  
  Next imgLink  
  
End Function

You are here: [Home](http://www.codeforexcelandoutlook.com/) / [Internet Explorer](http://www.codeforexcelandoutlook.com/blog/category/internet-explorer/) / An exploration of IE browser methods, part II

# An exploration of IE browser methods, part II

June 8, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [13 Comments](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/#comments)

In a [previous post](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/) I promised to go through a real example of how we can [automate Internet Explorer](http://www.codeforexcelandoutlook.com/excel-vba/automate-internet-explorer/) to retrieve data from a website. We'll also enumerate some of the members of the IE Document Object as well as see how we can incorporate XMLHTTP into the mix.

The EPA's SunWise Program publishes a [UV Index](http://www.epa.gov/sunwise/uvindex.html), which is the amount of UV exposure you can expect for the day. Their website has some neat graphs of how the UV Index works, and what each value means. I'm obliged to remind you that you should cover up in the sun, wear at least SPF 15 sunblock, and remember that just because it isn't sunny out doesn't mean you're not getting exposed to UV rays.

You can [lookup the UV Index](http://www.epa.gov/sunwise/uvindex.html#lookup) directly on their site. But we don't want to do that. We want to extract the UV index programmatically from the EPA's website.

The first thing we do is look at the source of the page (View > Source in Internet Explorer, Ctrl+U in FireFox) and see that the UV Index lookup is wrapped in HTML **form** tags. Wonderful! Here is the code.

<form action="http://oaspub.epa.gov/enviro/uv\_search" method="post" id="uviform" onsubmit="return validateInput();">  
  <table border="0" cellpadding="1" cellspacing="0" summary="The UV Index query form." style="width:auto; margin-right:60px;">  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label">ZIP Code:</label></td>  
      <td><input name="zipcode" type="text" id="zipcode" size="5" maxlength="5" /></td>  
    </tr>  
    <tr>  
      <td>&nbsp;</td>  
      <td style="padding:0.7em 0;">or</td>  
    </tr>  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label2">City:</label></td>  
      <td><input id="city\_name" type="text" name="city\_name" size="30" /></td>  
    </tr>  
    <tr>  
      <td style="text-align:right; padding-right:0.5em; font-weight:bold;"><label for="label3">State:</label></td>  
      <td><input name="state\_code" type="text" id="state\_code" title="Postal abbreviation" size="2" maxlength="2" /></td>  
    </tr>  
    <tr>  
      <td>&nbsp;</td>  
      <td style="padding-top:20px;"><input type="submit" id="submit" style="font-weight:bold;" value="Get UV Index" />  
        <input type="reset" id="reset" style="font-size:0.9em;" value="Reset" /></td>  
    </tr>  
  </table>  
</form>

This markup tells me:

* The data you enter into this form is submitted to a URL for processing.
* The base URL for my query is "http://oaspub.epa.gov/enviro/uv\_search".
* The form takes three inputs, named **zipcode**, **city\_name** and **state\_code**.

How did I figure that out?

* The ["action" attribute](http://www.w3schools.com/html/html_forms.asp) of the form tag gives you the base URL for any query using that form's data. So any IE navigation or XMLHTTP query I want must begin with "http://oaspub.epa.gov/enviro/uv\_search".
* There are three input fields for this particular form. Those are the fields that will be sent to the URL in the action attribute of the form tag. The name attribute for each one gives you the parameters you'll need to add to the query.

Any HTTP GET request must be formatted like this:

**base URL?first\_parameter=value&second\_parameter=value&third\_parameter=value**

In other words, to start the parameter list, add a question mark to the end of the base URL. Each subsequent parameter is prepended with an ampersand ('**&**') and added to the end of the parameter list. Here's a great example: [HTTP Client Methods – GET and POST](http://www.webdevelopersnotes.com/basics/http_client_methods_get_post.php3)

Since we can submit the form using only zip code, the URL for the query would look like this:

**http://oaspub.epa.gov/enviro/uv\_search?zipcode=11103**

(It would actually be ***http://oaspub.epa.gov/enviro/uv\_search?zipcode=11103&city\_name=&state\_code=*** but we can leave off the other parameters since they are blank anyway)

If you enter either URL into your web browser, and view the source of the resulting page, the tag we're looking for looks like this:

<img src="//www.epa.gov/enviro/gif/C\_UV08.gif" alt="UVI 8" border=0>

This is the graphic that displays the UV Index. The index number itself is in the alt attribute of the IMG tag. So now we'll need to move to the HTML Object Model to parse the HTML from this page and extract the UV Index number.

## The easy way

We'll use a simple trick: use the XMLHTTP object to retrieve the result of the query, instead of instantiating Internet Explorer. This will make our code run much faster, since we figured out how to get our result without having to actually visit the site.

**No early bound references required!**

Function GetUVIndex(zipCode As String) As String  
  
Dim xml As Object ' MSXML2.XMLHTTP  
Dim html As Object ' MSHTML.HTMLDocument  
Dim imgLinks As Object ' MSHTML.IHTMLElementCollection  
Dim imgLink As Object ' MSHTML.IHTMLElement  
Dim result As String  
  
  Set xml = CreateObject("MSXML2.XMLHTTP.6.0")  
  
  With xml  
    .Open "GET", "http://oaspub.epa.gov/enviro/uv\_search?zipcode=" & zipCode, False  
    .Send  
  End With  
  
  result = xml.responseText  
  
  ' create HTML document  
 Set html = CreateObject("htmlfile") ' New MSHTML.HTMLDocument  
  
  html.body.innerHTML = result  
  
  Set imgLinks = html.getElementsByTagName("img")  
  
  For Each imgLink In imgLinks  
    If Left$(imgLink.getAttribute("alt"), 3) = "UVI" Then  
      GetUVIndex = Right$(imgLink.getAttribute("alt"), \_  
          Len(imgLink.getAttribute("alt")) - InStrRev(imgLink.getAttribute("alt"), " "))  
      Exit Function  
    End If  
  Next imgLink  
  
End Function

### Sample usage

Sub TestGetUVIndex()  
  
Debug.Print GetUVIndex("11103")  
  
End Sub

We use XMLHTTP to run the query, but instead of using the MSXML Object Model, we use the HTML Object Model because the result is HTML, **not** XML.

After grabbing the result page, we put it into a HTMLDocument Object. The **getElementsByTagName** Method returns a collection of all the elements with the given tag name (for example: **input**, **a**, **img**). The **getAttribute** Method will return the given attribute name for each tag element. Then it's a simple matter of looking for the correct tag information.

We're looking for the IMG tag that contains the UV Index in its alt attribute, so we use **html.getElementsByTagName("img")** and then loop through the resulting collection.

## The hard way

Let's explore how we can do this using only IE automation (no XMLHTTP).

I call this "the hard way" because you will see how much more is required of this method.

You'll need to study the source code of the first webpage much more closely, to figure out the field names you want to fill out along with the appropriate methods for each. We didn't need to do that when we pointed XMLHTTP directly at the endpoint URL.

Here's the function and then we'll discuss further.

Function GetUVIndexWithIE(zipCode As String) As String  
  
Dim ie As Object ' InternetExplorer  
Dim html As Object ' MSHTML.HTMLDocument  
Dim imgLinks As Object ' MSHTML.IHTMLElementCollection  
Dim imgLink As Object ' MSHTML.IHTMLElement  
Dim inputCollection As Object ' MSHTML.IHTMLElementCollection  
Dim inputElement As Object ' MSHTML.IHTMLElement  
Dim result As String  
  
  Set ie = CreateObject("InternetExplorer.Application")  
  
  ' go to target page  
 ie.Navigate "http://www.epa.gov/sunwise/uvindex.html"  
  'ie.Visible = True  
  
  ' loop until page is loaded  
 Do Until ie.READYSTATE = 4  ' READYSTATE\_COMPLETE  
   DoEvents  
  Loop  
  
  ' enter zip code  
 ' there's only one page element with id="zipcode", so we  
 ' can enter its value directly  
 ie.document.getElementById("zipcode").value = zipCode  
  
  ' click submit button  
 ' there are multiple input elements named "submit", so we  
 ' need to loop  
 Set inputCollection = ie.document.getElementsByTagName("input")  
  
  For Each inputElement In inputCollection  
    If inputElement.getAttribute("value") = "Get UV Index" Then  
      inputElement.Click  
      Exit For  
    End If  
  Next inputElement  
  
  ' loop until page is loaded  
 Do Until ie.READYSTATE = 4  ' READYSTATE\_COMPLETE  
   DoEvents  
  Loop  
  
  ' grab screenshot  
 result = ie.document.body.innerHTML  
  
  ' close IE  
 ie.Quit  
  
  ' create HTML document  
 Set html = CreateObject("htmlfile") ' New MSHTML.HTMLDocument  
  
  html.body.innerHTML = result  
  
  ' loop through IMG tags  
 Set imgLinks = html.getElementsByTagName("img")  
  
  For Each imgLink In imgLinks  
    If Left$(imgLink.getAttribute("alt"), 3) = "UVI" Then  
      GetUVIndexWithIE = Right$(imgLink.getAttribute("alt"), \_  
                Len(imgLink.getAttribute("alt")) - InStrRev(imgLink.getAttribute("alt"), " "))  
      Exit Function  
    End If  
  Next imgLink  
  
End Function

The first thing we do is instantiate Internet Explorer, which itself can take a few seconds. After navigating to the first page, we need to loop to wait until the page is loaded. This will usually happen quickly, but if the page never loads, or loads very slowly, we could be here a while. In the declarations section as well as inline, I provide the early-bound versions in case you want to set a reference to Microsoft Internet Controls and [take advantage of early-binding](http://www.codeforexcelandoutlook.com/blog/2008/11/take-advantage-of-intellisense-when-writing-late-bound-code/), or if you just want to look up each object on MSDN to see its members.

Now we'll need to locate the input element for zip code. By studying the underlying HTML source for the page, I learned that the zip code field is the only one named "zipcode":

<input name="zipcode" type="text" id="zipcode" size="5" maxlength="5" />

That means it is the only field with the name attribute "zipcode". I can therefore input the zip code directly into the field like this:

ie.document.getElementById("zipcode").value = zipCode

Take note: this is how you input anything on a webpage using IE automation where the field has a unique name. Unfortunately, this is the only place where we're able to do this — we'll need to loop to find the Submit button because there are multiple Submit buttons on this page.

The Submit button is an input element; that is to say, the tag name is "input":

<input type="submit" id="submit" style="font-weight:bold;" value="Get UV Index" />

So we set an object reference to the collection of input elements using this syntax:

ie.document.getElementsByTagName("input")

Late bound, you would declare the object reference As Object; early bound it is a MSHTML.IHTMLElementCollection Object.

As we saw earlier, the **getElementsByTagName** Method returns a collection of all the elements with the given tag name. So if I needed all the links (<a> tag), I would use **ie.document.getElementsByTagName("a")** and if I wanted all the text boxes (<textarea> tag), I would write **ie.document.getElementsByTagName("textarea")**. For a list of HTML tags you can grab, see [HTML Reference](http://www.w3schools.com/TAGS/default.asp).

The Submit button's **value** attribute DOES have a unique name, so we'll check for that in our loop:

' use Click Method on appropriate Submit button  
For Each inputElement In inputCollection  
  If inputElement.getAttribute("value") = "Get UV Index" Then  
    inputElement.Click  
    Exit For  
  End If  
Next inputElement

We loop again to make sure the next page is fully loaded, then "screen scrape" it into a String variable. The next step is very important. Once we're done with IE, we call the Quit Method to make sure it doesn't hang around after the function ends.

After that, the code is identical to the function that uses XMLHTTP. The resulting page is put into an HTMLDocument Object and parsed for the UV Index.

## The Results

How do the two functions stack up?

In my tests, [GetUVIndexWithIE](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/#getuvindexwithie) took anywhere from 1.5 to 2.5 seconds, where [GetUVIndex](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/#GetUVIndex) took less than 1/10 of a second. Try this test procedure and report your findings.

Sub TestGetUVIndex()  
  
Dim starttime As Single  
Dim endtime As Single  
  
  starttime = Timer  
  Debug.Print GetUVIndexWithIE("11103")  
  endtime = Timer  
  
  Debug.Print "GetUVIndexWithIE took " & endtime - starttime & " seconds."  
  
  starttime = Timer  
  Debug.Print GetUVIndex("11103")  
  endtime = Timer  
  
  Debug.Print "GetUVIndex took " & endtime - starttime & " seconds."  
  
End Sub

## HTML Document Object Members

The Document Object is one of the more important Objects in the HTML Object Model because it represents the document being displayed by the browser at any given time. This is the typical object you'll be working with when automating IE, filling in forms and clicking web buttons. Here are some of its members.

All of these functions assume that you have already navigated to a URL.

### Document Object Properties

The Charset Property returns something like "utf-8" for Unicode support.

Function GetCharSet(ie As Object) As String  
  GetCharSet = ie.document.Charset  
End Function

The last modified date of the current page.

Function GetLastModified(ie As Object) As Date  
  GetLastModified = ie.document.LastModified  
End Function

ReadyState Property

Function GetReadyState(ie As Object) As String  
  GetReadyState = ie.document.READYSTATE  
End Function

Current document URL

Function GetURL(ie As Object) As String  
  GetURL = ie.document.URL  
End Function

### Document Object Collections

The Document Object has several important collections you can iterate through pretty easily.

All Elements

Function GetAllElements(ie As Object) As Object  
  Set GetAllElements = ie.document.all  
End Function

Anchors Collection

From MSDN:

Retrieves a collection of all a objects that have a name and/or id property. Objects in this collection are in HTML source order.

Just like getElementByID, this function returns all the A tags (links) that have a name or ID attribute. The important thing to note is the "source order" which is the order in which the links appear in the document.

Function GetAllAnchors(ie As Object) As Object  
  Set GetAllAnchors = ie.document.anchors  
End Function

Images Collection

From MSDN:

Retrieves a collection, in source order, of img objects in the document.

Function GetAllImages(ie As Object) As Object  
  Set GetAllImages = ie.document.images  
End Function

Links Collection

From MSDN:

Retrieves a collection of all a objects that specify the href property and all area objects in the document.

Function GetAllLinks(ie As Object) As Object  
  Set GetAllLinks = ie.document.Links  
End Function

### Document Object Methods

Some of the more popular methods are listed here. Each of these returns an object or collection of objects, which you iterate through in your code to fill in forms, click buttons, follow links, and so on.

Function GetSpecificElement(ie As Object, elementName As String) As Object  
  Set GetSpecificElement = ie.document.getElementById(elementName)  
End Function

Function GetElementsByName(ie As Object, elementName As String) As Object  
  Set GetElementsByName = ie.document.getElementsByName(elementName)  
End Function

Function GetElementsByTagName(ie As Object, tagName As String) As Object  
  Set GetElementsByTagName = ie.document.getElementsByTagName(tagName)  
End Function

### Sample usage

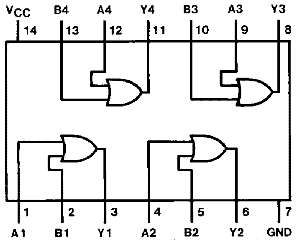
This function returns an instance of IE, and optionally visits a URL. It uses a simple RegEx pattern to validate the URL passed to the function.

Function GetIE(Optional URL As String) As Object  
  
Dim regExp As Object ' VBScript\_RegExp\_55.regExp  
  
  On Error Resume Next  
  Set GetIE = CreateObject("InternetExplorer.Application")  
  On Error GoTo 0  
  
  If Not GetIE Is Nothing Then  
    If Len(URL) > 0 Then  
      ' validate URL  
  
      ' http://www.dailydoseofexcel.com/archives/2005/08/13/pattern-matching/  
     Set regExp = CreateObject("VBScript.RegExp")  
  
      With regExp  
        .MultiLine = False  
        .Global = False  
        .IgnoreCase = True  
        ' http://regexlib.com/Search.aspx?k=url  
       .Pattern = "^https?://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S\*)?$"  
      End With  
  
      ' match URL against pattern  
     If regExp.Test(URL) Then  
        GetIE.Navigate URL  
      End If  
    End If  
  End If  
End Function

Sub TestIEDocumentObject()  
  
Dim ie As Object  
  
  Set ie = GetIE("http://www.google.com/")  
  
  If Not ie Is Nothing Then  
    Debug.Print GetCharSet(ie)  
  End If  
End Sub

# An exploration of IE browser methods, part III

August 17, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [6 Comments](http://www.codeforexcelandoutlook.com/blog/2010/08/an-exploration-of-ie-browser-methods-part-iii/#comments)



In previous articles I've written about how to extract information from webpages, and interact with them to navigate through a website.

This article won't strictly focus on IE, but on several collections available through the MSHTML Object Model.

## Previous articles:

* [An exploration of IE browser methods, part I](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/)
* [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/)
* [Automate Internet Explorer](http://www.codeforexcelandoutlook.com/excel-vba/automate-internet-explorer/)

## The GetMSXML Function

I've started using this function to return the XMLHTTP object. Hopefully I'll remember to use it consistently.

We'll use it because it's much faster than automating IE, as we learned [previously](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/), and we're more concerned with the HTML document once it's created rather than the method of actually creating it.

Function GetMSXML() As Object ' MSXML2.XMLHTTP  
 On Error Resume Next  
  Set GetMSXML = CreateObject("MSXML2.XMLHTTP.6.0")  
End Function

## Creating MSHTML Documents

Another function we'll need creates the HTML document which we'll load the HTML into. We also saw this code in [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/).

Function CreateHTMLDoc() As Object ' MSHTML.HTMLDocument  
 Set CreateHTMLDoc = CreateObject("htmlfile")  
End Function

## HTML Document Object Collections

These are the three collections we'll be working with here: **Anchors**, **Images**, **Links**. We dealt with them in [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/), but here I'll be going into a bit more detail.

Each one accepts a HTML Document Object (which we've already loaded with content from a website) and returns a reference to the relevant collection. The collection can then be iterated to look for a value, compare values, and so on.

### Anchors Collection

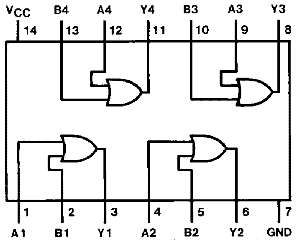
The Anchors Collection "retrieves a collection of all a objects that have a name and/or id property. Objects in this collection are in HTML source order."

Function GetHTMLAnchors(htmlDoc As Object) As Object ' MSHTML.IHTMLElementCollection  
 Set GetHTMLAnchors = htmlDoc.anchors  
End Function

You are here: [Home](http://www.codeforexcelandoutlook.com/) / [Internet Explorer](http://www.codeforexcelandoutlook.com/blog/category/internet-explorer/) / An exploration of IE browser methods, part III

# An exploration of IE browser methods, part III

August 17, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [6 Comments](http://www.codeforexcelandoutlook.com/blog/2010/08/an-exploration-of-ie-browser-methods-part-iii/#comments)



In previous articles I've written about how to extract information from webpages, and interact with them to navigate through a website.

This article won't strictly focus on IE, but on several collections available through the MSHTML Object Model.

## Previous articles:

* [An exploration of IE browser methods, part I](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/)
* [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/)
* [Automate Internet Explorer](http://www.codeforexcelandoutlook.com/excel-vba/automate-internet-explorer/)

## The GetMSXML Function

I've started using this function to return the XMLHTTP object. Hopefully I'll remember to use it consistently.

We'll use it because it's much faster than automating IE, as we learned [previously](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/), and we're more concerned with the HTML document once it's created rather than the method of actually creating it.

Function GetMSXML() As Object ' MSXML2.XMLHTTP  
 On Error Resume Next  
  Set GetMSXML = CreateObject("MSXML2.XMLHTTP.6.0")  
End Function

## Creating MSHTML Documents

Another function we'll need creates the HTML document which we'll load the HTML into. We also saw this code in [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/).

Function CreateHTMLDoc() As Object ' MSHTML.HTMLDocument  
 Set CreateHTMLDoc = CreateObject("htmlfile")  
End Function

## HTML Document Object Collections

These are the three collections we'll be working with here: **Anchors**, **Images**, **Links**. We dealt with them in [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/), but here I'll be going into a bit more detail.

Each one accepts a HTML Document Object (which we've already loaded with content from a website) and returns a reference to the relevant collection. The collection can then be iterated to look for a value, compare values, and so on.

### Anchors Collection

The Anchors Collection "retrieves a collection of all a objects that have a name and/or id property. Objects in this collection are in HTML source order."

Function GetHTMLAnchors(htmlDoc As Object) As Object ' MSHTML.IHTMLElementCollection  
 Set GetHTMLAnchors = htmlDoc.anchors  
End Function

### The Images Collection

The Images Collection "retrieves a collection, in source order, of img objects in the document."

Function GetHTMLImages(htmlDoc As Object) As Object ' MSHTML.IHTMLElementCollection  
 Set GetHTMLImages = htmlDoc.images  
End Function

### The Links Collection

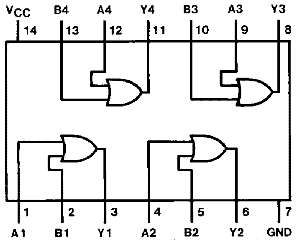
The Links Collection "retrieves a collection of all a objects that specify the href property and all area objects in the document.".

Function GetHTMLLinks(htmlDoc As Object) As Object ' MSHTML.IHTMLElementCollection  
 Set GetHTMLLinks = htmlDoc.links  
End Function

You are here: [Home](http://www.codeforexcelandoutlook.com/) / [Internet Explorer](http://www.codeforexcelandoutlook.com/blog/category/internet-explorer/) / An exploration of IE browser methods, part III

# An exploration of IE browser methods, part III

August 17, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [6 Comments](http://www.codeforexcelandoutlook.com/blog/2010/08/an-exploration-of-ie-browser-methods-part-iii/#comments)



In previous articles I've written about how to extract information from webpages, and interact with them to navigate through a website.

This article won't strictly focus on IE, but on several collections available through the MSHTML Object Model.

## Previous articles:

* [An exploration of IE browser methods, part I](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-i/)
* [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/)
* [Automate Internet Explorer](http://www.codeforexcelandoutlook.com/excel-vba/automate-internet-explorer/)

## The GetMSXML Function

I've started using this function to return the XMLHTTP object. Hopefully I'll remember to use it consistently.

We'll use it because it's much faster than automating IE, as we learned [previously](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/), and we're more concerned with the HTML document once it's created rather than the method of actually creating it.

Function GetMSXML() As Object ' MSXML2.XMLHTTP  
 On Error Resume Next  
  Set GetMSXML = CreateObject("MSXML2.XMLHTTP.6.0")  
End Function

## Creating MSHTML Documents

Another function we'll need creates the HTML document which we'll load the HTML into. We also saw this code in [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/).

Function CreateHTMLDoc() As Object ' MSHTML.HTMLDocument  
 Set CreateHTMLDoc = CreateObject("htmlfile")  
End Function

## HTML Document Object Collections

These are the three collections we'll be working with here: **Anchors**, **Images**, **Links**. We dealt with them in [An exploration of IE browser methods, part II](http://www.codeforexcelandoutlook.com/blog/2010/06/an-exploration-of-ie-browser-methods-part-ii/), but here I'll be going into a bit more detail.

Each one accepts a HTML Document Object (which we've already loaded with content from a website) and returns a reference to the relevant collection. The collection can then be iterated to look for a value, compare values, and so on.

### Anchors Collection

The Anchors Collection "retrieves a collection of all a objects that have a name and/or id property. Objects in this collection are in HTML source order."

Function GetHTMLAnchors(htmlDoc As Object) As Object ' MSHTML.IHTMLElementCollection  
 Set GetHTMLAnchors = htmlDoc.anchors  
End Function

### The Images Collection

The Images Collection "retrieves a collection, in source order, of img objects in the document."

Function GetHTMLImages(htmlDoc As Object) As Object ' MSHTML.IHTMLElementCollection  
 Set GetHTMLImages = htmlDoc.images  
End Function

### The Links Collection

The Links Collection "retrieves a collection of all a objects that specify the href property and all area objects in the document.".

Function GetHTMLLinks(htmlDoc As Object) As Object ' MSHTML.IHTMLElementCollection  
 Set GetHTMLLinks = htmlDoc.links  
End Function

Note that each collection returns objects "in source order", which means that the first link you see on a page will be the first link in the Anchors/Links Collection. Also, all three Collections have **Length** and **Item** Properties, which let you check the amount of items in the collection and return each individual item, respectively.

## Looping through each collection

In the sample procedure below, we create a new HTML document and populate it with the source code from Google's homepage (hey, got a better site to test? Leave a comment!). Then we grab each collection in turn and print various attributes from each member.

Sub TestHTMLCollections()  
  
On Error GoTo ErrorHandler  
  
Dim xml As Object ' MSXML2.XMLHTTP  
Dim htmlDoc As Object ' MSHTML.HTMLDocument  
Dim htmlBody As Object ' MSHTML.HTMLBody  
Dim anchors As Object ' MSHTML.IHTMLElementCollection  
Dim anchor As Object ' MSHTML.IHTMLElement  
Dim images As Object ' MSHTML.IHTMLElementCollection  
Dim image As Object ' MSHTML.IHTMLElement  
Dim links As Object ' MSHTML.IHTMLElementCollection  
Dim link As Object ' MSHTML.IHTMLElement  
  
Set xml = GetMSXML  
Set htmlDoc = CreateHTMLDoc  
Set htmlBody = htmlDoc.body  
  
With xml  
  .Open "GET", "http://www.google.com/", False  
  .send  
End With  
  
  ' put xml response into HTML document  
 htmlBody.innerHTML = xml.responseText  
  
  ' get all anchors  
 Set anchors = GetHTMLAnchors(htmlDoc)  
  
  ' number of anchor tags  
 Debug.Print anchors.Length  
    
  ' print each anchor tag contents  
 For Each anchor In anchors  
    Debug.Print anchor.innerHTML  
  Next anchor  
    
  ' get all images  
 Set images = GetHTMLImages(htmlDoc)  
    
  ' number of images  
 Debug.Print images.Length  
    
  ' print each image URL (img 'src' attribute)  
 For Each image In images  
    Debug.Print image.getAttribute("src")  
  Next image  
    
  ' get links collection  
 Set links = GetHTMLLinks(htmlDoc)  
    
  ' number of links  
 Debug.Print links.Length  
    
  ' print link target  
 For Each link In links  
    Debug.Print link.getAttribute("href")  
  Next link  
  
ProgramExit:  
  Exit Sub  
ErrorHandler:  
  MsgBox Err.number & " - " & Err.description  
  Resume ProgramExit  
End Sub

# Another URL shortener API

January 14, 2011 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [Leave a Comment](http://www.codeforexcelandoutlook.com/blog/2011/01/another-url-shortener-api/#comments)



I've posted code for several URL shorteners already:

* [Create Tiny URLs using VBA](http://www.codeforexcelandoutlook.com/blog/2009/06/create-tiny-urls-using-vba/)
* [bit.ly API v3 update](http://www.codeforexcelandoutlook.com/blog/2010/05/bit-ly-api-v3-update/)

Here's one more, using the API for [omani.ac](http://omani.ac/).

This code has a lot in common with the previous post's code for the [scr.im API](http://www.codeforexcelandoutlook.com/blog/2011/01/another-way-to-hide-your-email/):

* Requires POST request (syntax courtesy of [Excel Geek](http://blog.excelgeek.com/2006/10/advanced-excel-vba-web-queries-parsing.html))
* Uses MSXML6.DLL (late bound)
* API response is cached to limit bandwidth usage

However, in this function we do check for a valid URL before continuing.

Function GetShortURL(longURL As String, \_  
    Optional forceRequery As Boolean = False) As String  
' http://omani.ac/  
Dim xml As Object ' MSXML2.XMLHTTP  
Dim xmlDoc As Object ' MSXML2.DOMDocument  
Dim xmlDocRoot As Object ' MSXML2.IXMLDOMNode  
Dim result As String  
Dim results() As String  
Dim tempFile As String  
  
Const XML\_FILE\_EXTENSION As String = ".xml"  
Const BASE\_URL As String = "http://omani.ac/api/shorten.xml"  
    
  ' check for valid string  
 If Len(longURL) = 0 Then  
    Exit Function  
  End If  
    
  tempFile = environ("temp") & "\" & \_  
    Replace(Replace(longURL, "/", "\_"), ":", "+") & XML\_FILE\_EXTENSION  
  
  ' requery if cache file is missing or forceRequery set to True  
 If Len(Dir(tempFile)) = 0 Or forceRequery Then  
  
    Set xml = CreateObject("MSXML2.XMLHTTP.6.0")  
      
    ' check for valid url  
   With xml  
      .Open "GET", longURL, False  
      .send  
      If .Status <> "200" Then  
        Exit Function  
      End If  
    End With  
  
    With xml  
      .Open "POST", BASE\_URL, False  
      .setRequestHeader "Content-Type", "application/x-www-form-urlencoded"  
      .send "url=" & longURL  
    End With  
  
    result = xml.responseText  
  
    CreateFile tempFile, result  
  
  End If  
  
  ' load XML file into new XML document  
 Set xmlDoc = CreateObject("MSXML2.DOMDocument.6.0")  
  
  With xmlDoc  
    .async = False  
    .validateOnParse = False  
    .Load tempFile  
  End With  
  
  ' check that the XML doc loaded  
 If LoadError(xmlDoc) Then  
    Exit Function  
  End If  
  
  ' get root node  
 Set xmlDocRoot = GetRootNode(xmlDoc)  
    
  ' grab last sub-node's value, even if it's an error  
   GetShortURL = GetNode(xmlDocRoot, xmlDocRoot.childNodes.Length).nodeTypedValue  
  
End Function

You'll need to check for return values that are either empty (invalid URL) or error values. See [omani.ac](http://omani.ac/) for a list of error codes.

## Helper Functions

Paste [these helper functions](http://www.codeforexcelandoutlook.com/vba/msxml-object-library-routines/) into a standard module in the same project as the above function. They are all used by the function above.

You'll also need to grab the [URLEncode function](http://www.freevbcode.com/ShowCode.Asp?ID=5137), courtesy of our friends at Free VB Code.

### Sample Usage

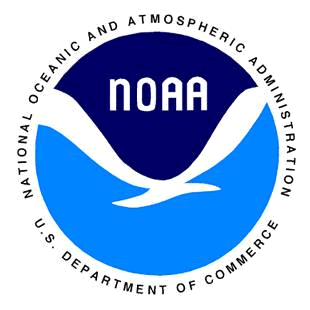
Sub TestURLs()  
  
Debug.Print GetShortURL("http://www.google.com/")  
  
End Sub

They also have a "lengthen" method, for taking omani.ac URLs and converting them back to the original (longer) URL. Extra credit for anyone who wants to tackle that challenge!

You are here: [Home](http://www.codeforexcelandoutlook.com/) / [VBA](http://www.codeforexcelandoutlook.com/blog/category/vba/) / Exploring the NOAA API

# Exploring the NOAA API

September 3, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [Leave a Comment](http://www.codeforexcelandoutlook.com/blog/2010/09/exploring-the-noaa-api/#comments)



The National Weather Service's [National Hurricane Center](http://www.nhc.noaa.gov/index.shtml) tracks storms that occur in the Atlantic and Eastern Pacific oceans. Starting in July 2010, weather advisories were made available via RSS. Let's take a look at one and see what we can do with it.

A list of available RSS feeds from NHC is [here](http://www.nhc.noaa.gov/aboutrss.shtml#rsslist).

The one we'll parse is the Atlantic (English) feed from the "Specific Tropical Cyclone Feeds by Basin" section. Here's an example of what it looks like when this article was written:

<?xml version="1.0" encoding="iso-8859-1"?>  
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">  
  <channel>  
    <pubDate>Tue, 27 Jul 2010 00:02:56 GMT</pubDate>  
    <title>National Hurricane Center (Atlantic)</title>  
    <description>Active tropical cyclones in the Atlantic, Caribbean, and the Gulf of Mexico</description>  
    <link>http://www.nhc.noaa.gov/</link>  
    <copyright>none</copyright>  
    <managingEditor>nhcwebmaster@noaa.gov (nhcwebmaster)</managingEditor>  
    <language>en-us</language>  
    <webMaster>nhcwebmaster@noaa.gov (nhcwebmaster)</webMaster>  
    <image>  
      <url>http://www.nhc.noaa.gov/gifs/xml\_logo\_nhc.gif</url>  
      <link>http://www.nhc.noaa.gov/</link>  
      <description>NOAA logo</description>  
      <title>National Hurricane Center (Atlantic)</title>  
      <width>95</width>  
      <height>45</height>  
    </image>  
    <item>  
      <title>There are no tropical cyclones at this time.</title>  
      <description>No tropical cyclones as of Tue, 27 Jul 2010 00:02:56 GMT</description>  
      <pubDate>Tue, 27 Jul 2010 00:02:56 GMT</pubDate>  
      <link>http://www.nhc.noaa.gov/</link>  
      <guid>http://www.nhc.noaa.gov/</guid>  
      <author>nhcwebmaster@noaa.gov (nhcwebmaster)</author>  
    </item>  
    <item>  
      <title>Atlantic Tropical Weather Outlook</title>  
      <description>  
      <![CDATA[  
      <br/>  
      000<br/>  
      ABNT20 KNHC 262336<br/>  
      TWOAT <br/>  
      TROPICAL WEATHER OUTLOOK<br/>  
      NWS TPC/NATIONAL HURRICANE CENTER MIAMI FL<br/>  
      800 PM EDT MON JUL 26 2010<br/>  
      <br/>  
      FOR THE NORTH ATLANTIC...CARIBBEAN SEA AND THE GULF OF MEXICO...<br/>  
      <br/>  
      TROPICAL CYCLONE FORMATION IS NOT EXPECTED DURING THE NEXT 48 HOURS.<br/>  
      <br/>  
      $$<br/>  
      FORECASTER CANGIALOSI/STEWART<br/>  
      <br/>  
      ]]>  
      </description>  
      <pubDate>Mon, 26 Jul 2010 23:36:56 GMT</pubDate>  
      <link>http://www.nhc.noaa.gov/gtwo\_atl.shtml</link>  
      <guid>http://www.nhc.noaa.gov/gtwo\_atl.shtml?201007262336</guid>  
      <author>nhcwebmaster@noaa.gov (nhcwebmaster)</author>  
    </item>  
  </channel>  
</rss>

Because of the way items are listed in the feed, we'll need to start parsing from the ninth node. Of course, you can grab more information, but I felt that the first eight nodes were useless.

## The NOAAWeather Function

This function returns all weather items available in the Atlantic (English) RSS feed from NHC/NOAA as a String array. As usual, I've preserved the early bound references so you can study the MSXML object model if you wish.

The API response is cached, so you'll need to pass a parameter of **True** (or clear your temp folder) to force a requery. And don't forget to include the [helper functions](http://www.codeforexcelandoutlook.com/vba/msxml-object-library-routines/) in your project.

Function NOAAWeather(Optional forceRequery As Boolean = False) As String()  
' Atlantic, Caribbean, and Gulf of Mexico tropical cyclones in English  
' http://www.nhc.noaa.gov/aboutrss.shtml  
 Dim xml As Object ' MSXML2.XMLHTTP  
 Dim result As String  
  Dim tempFile As String  
  Dim xmlDoc As Object 'MSXML2.DOMDocument  
 Dim xmlDocRoot As Object 'MSXML2.IXMLDOMNode  
 Dim rss As Object 'MSXML2.IXMLDOMNode  
 Dim channel As Object 'MSXML2.IXMLDOMNodeList  
 Dim item As Object 'MSXML2.IXMLDOMNode  
 Dim subNodes As Object 'MSXML2.IXMLDOMNodeList  
 Dim i As Long, j As Long  
  Dim tempString() As String  
  Dim numRows As Long  
  
  Const numCols As Long = 6  
  
  Set xml = CreateObject("MSXML2.XMLHTTP.6.0")  
  
  tempFile = Environ("temp") & Application.PathSeparator & "weather.xml"  
  
  If (Len(Dir(tempFile)) = 0 Or forceRequery) Then  
  
    With xml  
      .Open "GET", "http://www.nhc.noaa.gov/index-at.xml"  
      .send  
    End With  
  
    result = xml.responseText  
  
    ' create XML file from result  
   Call CreateXMLFile(tempFile, result)  
  
  End If  
  
  Set xmlDoc = CreateObject("MSXML2.DOMDocument")  
  
  With xmlDoc  
    .async = False  
    .validateOnParse = False  
    .Load tempFile  
  End With  
  
  ' check that the XML doc loaded  
 If LoadError(xmlDoc) Then  
    Exit Function  
  End If  
  
  ' get root node  
 Set xmlDocRoot = GetRootNode(xmlDoc)  
  Set rss = xmlDocRoot.childNodes(0)  
  Set channel = GetChildNodes(rss)  
  
  numRows = channel.Length - 9  
  
  ' resize array  
 ReDim tempString(1 To numRows, 1 To numCols)  
  
  For i = 9 To channel.Length - 1  
    Set item = channel.item(i)  
    Set subNodes = GetChildNodes(item)  
  
    For j = 1 To subNodes.Length  
      tempString(i - 8, j) = subNodes.item(j - 1).nodeTypedValue  
    Next j  
  Next i  
  
  NOAAWeather = tempString  
  
End Function

As I mentioned earlier, I felt the first nine nodes were useless, so the number of "rows" in the array is nine less than the total. Since there are eleven nodes, the total output will be two "records", with six pieces of information each. Then it's a "simple" matter of parsing the XML to grab the data we want.

## Sample usage

Sub TestNOAAWebsite()  
  
  Dim results() As String  
  Dim i As Long, j As Long  
  
  results = NOAAWeather  
  
  For i = LBound(results) To UBound(results)  
    For j = LBound(results, 2) To UBound(results, 2)  
      Debug.Print results(i, j)  
    Next j  
  Next i  
  
End Sub

## Output

The output looks like this:

There are no tropical cyclones at this time.  
No tropical cyclones as of Mon, 26 Jul 2010 21:55:14 GMT  
Mon, 26 Jul 2010 21:55:14 GMT

http://www.nhc.noaa.gov/

http://www.nhc.noaa.gov/

nhcwebmaster@noaa.gov (nhcwebmaster)  
Atlantic Tropical Weather Outlook

000  
  
ABNT20 KNHC 261747  
  
TWOAT   
  
TROPICAL WEATHER OUTLOOK  
  
NWS TPC/NATIONAL HURRICANE CENTER MIAMI FL  
  
200 PM EDT MON JUL 26 2010  
  
  
  
FOR THE NORTH ATLANTIC…CARIBBEAN SEA AND THE GULF OF MEXICO…  
  
  
  
TROPICAL CYCLONE FORMATION IS NOT EXPECTED DURING THE NEXT 48 HOURS.  
  
  
  
$$  
  
FORECASTER ROBERTS/PASCH

Mon, 26 Jul 2010 17:47:23 GMT

http://www.nhc.noaa.gov/gtwo\_atl.shtml

http://www.nhc.noaa.gov/gtwo\_atl.shtml?201007261747

nhcwebmaster@noaa.gov (nhcwebmaster)

[Code For Excel And Outlook - Learn VBA](http://www.codeforexcelandoutlook.com/)

Automation and VBA code for Microsoft® Excel and Outlook

You are here: [Home](http://www.codeforexcelandoutlook.com/) / [VBA](http://www.codeforexcelandoutlook.com/vba/) / MSXML Object Library Routines

If you were directed to this page, it means that you need to copy and paste one or more of the following procedures into a standard module in the same project as the code you are working on.

I've placed the code here to avoid having to duplicate it on every page that shows how to consume some web API's XML response.

You may not need all of these procedures for the particular API you're consuming. Copy and paste as needed, or copy the whole thing into a standard module and remove what you don't need.

The MSXML routines I use require MSXML6.DLL in the local system32 folder. If you want to use a prior version, just change "6.0" in the GetMSXML function to point to the version you want to use: 2.0, 3.0, and so on.

Function GetMSXML() As Object  '  MSXML2.XMLHTTP60  
On Error Resume Next  
  Set GetMSXML = CreateObject("MSXML2.XMLHTTP.6.0")  
End Function  
  
Function GetNode(parentNode As Object, nodeNumber As Long) As Object  
  On Error Resume Next  
  ' if parentNode is a MSXML2.IXMLDOMNodeList  
Set GetNode = parentNode.item(nodeNumber - 1)  
  
  ' if parentNode is a MSXML2.IXMLDOMNode  
If GetNode Is Nothing Then  
    Set GetNode = parentNode.childNodes(nodeNumber - 1)  
  End If  
End Function  
  
Public Function GetChildNodes(node As Object) As Object  
' returns child nodes for a given MSXML2.IXMLDOMNode  
Set GetChildNodes = node.childNodes  
End Function  
  
Function GetRootNode(xmlDoc As Object) As Object  
' returns root node  
Set GetRootNode = xmlDoc.documentElement  
End Function  
  
Function LoadError(xmlDoc As Object) As Boolean  
' checks if a xml file load error occurred  
LoadError = (xmlDoc.parseError.errorCode <> 0)  
End Function  
  
Function CreateFile(fileName As String, contents As String) As String  
' creates file from string contents  
  
Dim tempFile As String  
Dim nextFileNum As Long  
  
  nextFileNum = FreeFile  
  
  tempFile = fileName  
  
  Open tempFile For Output As #nextFileNum  
  Print #nextFileNum, contents  
  Close #nextFileNum  
  
  CreateFile = tempFile  
End Function  
  
Public Function CreateXMLFile(fileName As String, contents As String) As String  
' creates XML file from string contents  
  
Dim tempFile As String  
Dim nextFileNum As Long  
  
  nextFileNum = FreeFile  
  
  tempFile = fileName  
  
  Open tempFile For Output As #nextFileNum  
  Print #nextFileNum, contents  
  Close #nextFileNum  
  
  CreateXMLFile = tempFile  
End Function  
  
Function ConvertAccent(ByVal inputString As String) As String  
' http://www.vbforums.com/archive/index.php/t-483965.html  
  
Const AccChars As String = \_  
      "²—­–ŠŽšžŸÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖÙÚÛÜÝàáâãäåçèéêëìíîïðñòóôõöùúûüýÿ’"  
Const RegChars As String = \_  
      "2---SZszYAAAAAACEEEEIIIIDNOOOOOUUUUYaaaaaaceeeeiiiidnooooouuuuyy'"  
  
Dim i As Long, j As Long  
Dim tempString As String  
Dim currentCharacter As String  
Dim found As Boolean  
Dim foundPosition As Long  
  
  tempString = inputString  
  
  ' loop through the shorter string  
Select Case True  
    Case Len(AccChars) <= Len(inputString)  
      ' accent character list is shorter (or same)  
    ' loop through accent character string  
    For i = 1 To Len(AccChars)  
  
        ' get next accent character  
      currentCharacter = Mid$(AccChars, i, 1)  
  
        ' replace with corresponding character in "regular" array  
      If InStr(tempString, currentCharacter) > 0 Then  
          tempString = Replace(tempString, currentCharacter, \_  
                               Mid$(RegChars, i, 1))  
        End If  
      Next i  
    Case Len(AccChars) > Len(inputString)  
      ' input string is shorter  
    ' loop through input string  
    For i = 1 To Len(inputString)  
  
        ' grab current character from input string and  
      ' determine if it is a special char  
      currentCharacter = Mid$(inputString, i, 1)  
        found = (InStr(AccChars, currentCharacter) > 0)  
  
        If found Then  
  
          ' find position of special character in special array  
        foundPosition = InStr(AccChars, currentCharacter)  
  
          ' replace with corresponding character in "regular" array  
        tempString = Replace(tempString, currentCharacter, \_  
                               Mid$(RegChars, foundPosition, 1))  
  
        End If  
      Next i  
  End Select  
  
  ConvertAccent = tempString  
End Function  
  
Function FixAngleBrackets(textString As String) As String  
  FixAngleBrackets = Replace(Replace(textString, "&lt;", "<"), "&gt;", ">")  
End Function

You may also need to urlencode your URLs. This is so they can be passed to the API. I use something I found on Free VB Code: [URLEncode for Large Strings](http://www.freevbcode.com/ShowCode.Asp?ID=5137)

# Another way to hide your email

January 12, 2011 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [Leave a Comment](http://www.codeforexcelandoutlook.com/blog/2011/01/another-way-to-hide-your-email/#comments)



Looking to post your email address on Twitter or on your website, but keep it away from the spammers? Here's one way using a neat API I found last week.

One of my favorite WordPress plugins is [Who Sees Ads](http://planetozh.com/blog/my-projects/wordpress-plugin-who-sees-ads-control-adsense-display/), which I use to selectively display ads and other content on this site. I highly recommend it if, for example, you only want search engine visitors to see your newsletter signup form.

Anyway, the plugin author has a website that lets you obfuscate your email address in a unique way. The site is called [Scr.im](http://scr.im/) and here is the rationale:

Leaving your email as plain text in forums, on Twitter or on classified sites makes you an easy spam target: spam robots and email harvesters constantly browse these sites to collect new victim emails.

Don't share your email on public sites. Instead, use our free service that will convert your email address (joe@email.com) into a safe and short URL (for instance http://scr.im/joe). People willing to email you will go to this URL that will reveal your email address, after a simple test that automated scripts and bots cannot pass.

Scr.im has an API that requires a POST request with your email address, as well as a (optional) keyword for a custom alias. So I wrote a VBA function that calls the scr.im website and returns the URL for you to use instead of your email address.

## Scr.im that email!

The following function calls the scr.im API and caches the response. It uses MSXML6.DLL which should be located in the local SYSTEM32 folder. I've decided to upgrade all my old code to use MSXML6.DLL because it is included with Windows Vista, 7 and later versions of XP, so most people using Windows should have it (whereas MSXML2.DLL is missing from my Vista laptop). However you do not need any early bound references for this code to work.

The response is cached because if you call the API with the same email address, you get the same response (i.e. you don't get a new URL). Plus, in keeping with best practice, we don't want to overload the API with useless repeated calls. Besides, if you want to change your custom alias, you can't do it programmatically.

POST request syntax courtesy of [Excel Geek](http://blog.excelgeek.com/2006/10/advanced-excel-vba-web-queries-parsing.html).

Function GetScrim(emailAddress As String, Optional scrimKeyword As String, \_  
                  Optional forceRequery As Boolean = False) As String  
' http://scr.im/api/  
Dim xml As Object ' MSXML2.XMLHTTP  
Dim xmlDoc As Object ' MSXML2.DOMDocument  
Dim xmlDocRoot As Object ' MSXML2.IXMLDOMNode  
Dim url As Object ' MSXML2.IXMLDOMNode  
Dim result As String  
Dim tempFile As String  
  
Const XML\_FILE\_EXTENSION As String = ".xml"  
Const BASE\_URL As String = "http://scr.im/xml/"  
  
  tempFile = environ("temp") & "\" & emailAddress & XML\_FILE\_EXTENSION  
  
  If Len(Dir(tempFile)) = 0 Or forceRequery Then  
  
    Set xml = CreateObject("MSXML2.XMLHTTP.6.0")  
  
    With xml  
      .Open "POST", BASE\_URL, False  
      .setRequestHeader "Content-Type", "application/x-www-form-urlencoded"  
        
      If Len(scrimKeyword) > 0 Then  
        .send "email=" & emailAddress & "&scrim=" & URLEncode(scrimKeyword)  
      Else  
        .send "email=" & emailAddress  
      End If  
    End With  
  
    result = xml.responseText  
  
    ' save result as temp XML document  
   tempFile = CreateXMLFile(tempFile, result)  
  
  End If  
  
  ' load XML file into new XML document  
 Set xmlDoc = CreateObject("MSXML2.DOMDocument.6.0")  
  
  With xmlDoc  
    .async = False  
    .validateOnParse = False  
    .Load tempFile  
  End With  
  
  ' check that the XML doc loaded  
 If LoadError(xmlDoc) Then  
    Exit Function  
  End If  
  
  ' get root node  
 Set xmlDocRoot = GetRootNode(xmlDoc)  
    
  ' get last node, return error code if applicable  
 Set url = GetNode(xmlDocRoot, xmlDocRoot.childNodes.Length)  
    
  GetScrim = url.nodeTypedValue  
End Function

## Helper Functions

[These helper functions](http://www.codeforexcelandoutlook.com/vba/msxml-object-library-routines/) are used by the above function. Paste them into a standard module in the same project. You'll also need the URLEncode function courtesy of [Free VB Code](http://www.freevbcode.com/ShowCode.Asp?ID=5137).

Notice that our function does not [validate email address](http://www.codeforexcelandoutlook.com/blog/2010/02/email-validation-in-vba/), which you might want to incorporate. Also, you'll need to check the output of the function to look for error messages; anything other than a URL is an error message.

### Sample Usage

Here's Joe's URL:

Sub ScrimThis()  
  
  Debug.Print GetScrim("joe@email.com", "joe")  
  
End Sub

# Screen Scraping 101 with VBA

August 10, 2010 by [JP](http://www.codeforexcelandoutlook.com/blog/author/jp/) · [7 Comments](http://www.codeforexcelandoutlook.com/blog/2010/08/screen-scraping-101-with-vba/#comments)



No, not that kind of scraping. You can capture and download virtually any webpage using VBA. Here's one method for doing so.

I've written a simple function that lets you screen scrape almost any webpage you can pass as a parameter. I say "almost" because some pages are rendered in Javascript or Flash (or frames!) and don't really reveal any useful source code. The function will return the raw client-side output as a String with no parsing whatsoever. How you choose to parse the information is up to you.

## The GetWebpage Function

This function should be familiar to anyone who has read any of my articles on [XMLHTTP](http://www.codeforexcelandoutlook.com/blog/tag/xmlhttp/) parsing. It uses the MSXML.XMLHTTP object to open and grab a web page. Simple as that. If you specify a filename as the second parameter, the webpage contents are written to a file.

Function GetWebpage(url As String, Optional fileName As String) As String  
  
Dim xml As Object ' MSXML2.XMLHTTP  
Dim result As String  
  
  Set xml = GetMSXML  
  
  ' grab webpage  
 With xml  
    .Open "GET", url, False  
    .send  
  End With  
  
  GetWebpage = xml.responseText  
  
  ' write to file?  
 If Len(fileName) > 0 Then  
    If Not FileExists(fileName) Then  
      Call CreateFile(fileName, GetWebpage)  
    Else ' file exists  
     If MsgBox("File already exists, overwrite?", vbYesNo) = vbYes Then  
        Call CreateFile(fileName, GetWebpage)  
      End If  
    End If  
  End If  
  
End Function

To create an instance of the XMLHTTP Object, I've started using this function:

Function GetMSXML() As Object  
  On Error Resume Next  
  Set GetMSXML = CreateObject("MSXML2.XMLHTTP.6.0")  
End Function

If you do choose to write the output to a file, you'll need these two functions:

Sub CreateFile(fileName As String, contents As String)  
' create file from string contents  
  
Dim tempFile As String  
Dim nextFileNum As Long  
  
  nextFileNum = FreeFile  
  
  tempFile = fileName  
  
  Open tempFile For Output As #nextFileNum  
  Print #nextFileNum, contents  
  Close #nextFileNum  
  
End Sub  
  
Function FileExists(fileName As String) As Boolean  
  FileExists = (Len(Dir(fileName)) > 0)  
End Function

## Sample usage

Sub TestGetWebpage()  
  
Dim webpage As String  
  
webpage = GetWebpage("http://www.codeforexcelandoutlook.com/blog/")  
  
Debug.Print webpage  
  
End Sub

<http://www.codeforexcelandoutlook.com>

http://msdn.microsoft.com/en-us/library/aa752084%28VS.85%29.aspx

# InternetExplorer Object

Controls an instance of Windows Internet Explorer through automation.

**Members Table**

The following table lists the members exposed by the **InternetExplorer** object.

Events

|  |  |
| --- | --- |
| **Event** | **Description** |
| [BeforeNavigate](http://msdn.microsoft.com/en-us/library/aa768325%28v=VS.85%29.aspx) | Fires before navigation occurs in the given object (on either a window or frameset element). |
| [BeforeNavigate2](http://msdn.microsoft.com/en-us/library/aa768326%28v=VS.85%29.aspx) | Fires before navigation occurs in the given object (on either a window element or a frameset element). |
| [CommandStateChange](http://msdn.microsoft.com/en-us/library/aa768328%28v=VS.85%29.aspx) | Fires when the enabled state of a command changes. |
| [DocumentComplete](http://msdn.microsoft.com/en-us/library/aa768329%28v=VS.85%29.aspx) | Fires when a document is completely loaded and initialized. |
| [DownloadBegin](http://msdn.microsoft.com/en-us/library/aa768330%28v=VS.85%29.aspx) | Fires when a navigation operation begins. |
| [DownloadComplete](http://msdn.microsoft.com/en-us/library/aa768331%28v=VS.85%29.aspx) | Fires when a navigation operation finishes, is halted, or fails. |
| [FileDownload](http://msdn.microsoft.com/en-us/library/bb268220%28v=VS.85%29.aspx) | Fires to indicate that a file download is about to occur. If a file download dialog box can be displayed, this event fires prior to the appearance of the dialog box. |
| [NavigateComplete](http://msdn.microsoft.com/en-us/library/aa768332%28v=VS.85%29.aspx) | Fires after a navigation to a link is completed on either a [window](http://msdn.microsoft.com/en-us/library/ms535873%28v=VS.85%29.aspx) element or a [frameSet](http://msdn.microsoft.com/en-us/library/ms535251%28v=VS.85%29.aspx) element. |
| [NavigateComplete2](http://msdn.microsoft.com/en-us/library/aa768334%28v=VS.85%29.aspx) | Fires after a navigation to a link is completed on a **window** element or a **frameSet** element. |
| [NavigateError](http://msdn.microsoft.com/en-us/library/bb268221%28v=VS.85%29.aspx) | Fires when an error occurs during navigation. |
| [NewProcess](http://msdn.microsoft.com/en-us/library/cc891504%28v=VS.85%29.aspx) | Creates a new process to handle the navigation. |
| [NewWindow](http://msdn.microsoft.com/en-us/library/aa768335%28v=VS.85%29.aspx) | Fires when a new window is to be created. |
| [NewWindow2](http://msdn.microsoft.com/en-us/library/aa768336%28v=VS.85%29.aspx) | Fires when a new window is to be created. |
| [NewWindow3](http://msdn.microsoft.com/en-us/library/aa768337%28v=VS.85%29.aspx) | Raised when a new window is to be created. Extends [NewWindow2](http://msdn.microsoft.com/en-us/library/aa768336%28v=VS.85%29.aspx) with additional information about the new window. |
| [OnFullScreen](http://msdn.microsoft.com/en-us/library/aa768338%28v=VS.85%29.aspx) | Fires when the [FullScreen](http://msdn.microsoft.com/en-us/library/aa752054%28v=VS.85%29.aspx) property is changed. |
| [OnMenuBar](http://msdn.microsoft.com/en-us/library/aa768339%28v=VS.85%29.aspx) | Fires when the [MenuBar](http://msdn.microsoft.com/en-us/library/aa752059%28v=VS.85%29.aspx) property is changed. |
| [OnQuit](http://msdn.microsoft.com/en-us/library/aa768340%28v=VS.85%29.aspx) | Fires before the Internet Explorer application quits. |
| [OnStatusBar](http://msdn.microsoft.com/en-us/library/aa768341%28v=VS.85%29.aspx) | Fires when the [StatusBar](http://msdn.microsoft.com/en-us/library/aa752075%28v=VS.85%29.aspx) property is changed. |
| [OnTheaterMode](http://msdn.microsoft.com/en-us/library/aa768342%28v=VS.85%29.aspx) | Fires when the [TheaterMode](http://msdn.microsoft.com/en-us/library/aa752077%28v=VS.85%29.aspx) property is changed. |
| [OnToolBar](http://msdn.microsoft.com/en-us/library/aa768343%28v=VS.85%29.aspx) | Fires when the [ToolBar](http://msdn.microsoft.com/en-us/library/aa752078%28v=VS.85%29.aspx) property is changed. |
| [OnVisible](http://msdn.microsoft.com/en-us/library/aa768344%28v=VS.85%29.aspx) | Fires when the [Visible](http://msdn.microsoft.com/en-us/library/aa752082%28v=VS.85%29.aspx) property of the object is changed. |
| [PrintTemplateInstantiation](http://msdn.microsoft.com/en-us/library/aa768345%28v=VS.85%29.aspx) | Fires when a print template is instantiated. |
| [PrintTemplateTeardown](http://msdn.microsoft.com/en-us/library/aa768346%28v=VS.85%29.aspx) | Fires when a print template is destroyed. |
| [PrivacyImpactedStateChange](http://msdn.microsoft.com/en-us/library/bb268223%28v=VS.85%29.aspx) | Fired when an event occurs that impacts privacy, or when a user navigates away from a URL that has impacted privacy. |
| [ProgressChange](http://msdn.microsoft.com/en-us/library/aa768347%28v=VS.85%29.aspx) | Fires when the progress of a download operation is updated on the object. |
| [PropertyChange](http://msdn.microsoft.com/en-us/library/aa768348%28v=VS.85%29.aspx) | Fires when the [PutProperty](http://msdn.microsoft.com/en-us/library/aa752095%28v=VS.85%29.aspx) method of the object changes the value of a property. |
| [RedirectXDomainBlocked](http://msdn.microsoft.com/en-us/library/dd565687%28v=VS.85%29.aspx) | Fired when a cross-domain redirect request is blocked. |
| [SetPhishingFilterStatus](http://msdn.microsoft.com/en-us/library/aa741308%28v=VS.85%29.aspx) | Fires to indicate the progress and status of Microsoft Phishing Filter analysis of the current webpage. |
| [SetSecureLockIcon](http://msdn.microsoft.com/en-us/library/bb268224%28v=VS.85%29.aspx) | Fires when there is a change in encryption level. |
| [StatusTextChange](http://msdn.microsoft.com/en-us/library/aa768349%28v=VS.85%29.aspx) | Fires when the status bar text of the object has changed. |
| [ThirdPartyUrlBlocked](http://msdn.microsoft.com/en-us/library/cc848905%28v=VS.85%29.aspx) | Fired when a third-party URL is blocked. |
| [TitleChange](http://msdn.microsoft.com/en-us/library/aa768350%28v=VS.85%29.aspx) | Fires when the title of a document in the object becomes available or changes. |
| [UpdatePageStatus](http://msdn.microsoft.com/en-us/library/aa768351%28v=VS.85%29.aspx) | Not implemented. |
| [WindowActivate](http://msdn.microsoft.com/en-us/library/bb268225%28v=VS.85%29.aspx) | Not implemented. |
| [WindowMove](http://msdn.microsoft.com/en-us/library/bb268226%28v=VS.85%29.aspx) | Not implemented. |
| [WindowResize](http://msdn.microsoft.com/en-us/library/bb268227%28v=VS.85%29.aspx) | Not implemented. |
| [WindowStateChanged](http://msdn.microsoft.com/en-us/library/aa768358%28v=VS.85%29.aspx) | Fires when the visibility state of a content window, such as the browser window or a tab, changes. |

Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| [ClientToWindow](http://msdn.microsoft.com/en-us/library/aa752086%28v=VS.85%29.aspx) | Computes the full size of the browser window when given the specified width and height of the content area. |
| [ExecWB](http://msdn.microsoft.com/en-us/library/aa752087%28v=VS.85%29.aspx) | Executes a command and returns the status of the command execution using the [IOleCommandTarget](http://msdn.microsoft.com/en-us/library/ms683797%28v=VS.85%29.aspx) interface. |
| [GetProperty](http://msdn.microsoft.com/en-us/library/aa752088%28v=VS.85%29.aspx) | Gets the value associated with a user-defined property name. |
| [GoBack](http://msdn.microsoft.com/en-us/library/aa752089%28v=VS.85%29.aspx) | Navigates backward one item in the history list. |
| [GoForward](http://msdn.microsoft.com/en-us/library/aa752090%28v=VS.85%29.aspx) | Navigates forward one item in the history list. |
| [GoHome](http://msdn.microsoft.com/en-us/library/aa752091%28v=VS.85%29.aspx) | Navigates to the current home or start page. |
| [GoSearch](http://msdn.microsoft.com/en-us/library/aa752092%28v=VS.85%29.aspx) | Navigates to the current search page. |
| [Navigate](http://msdn.microsoft.com/en-us/library/aa752093%28v=VS.85%29.aspx) | Navigates to a resource identified by a URL or to a file identified by a full path. |
| [Navigate2](http://msdn.microsoft.com/en-us/library/aa752094%28v=VS.85%29.aspx) | Navigates the browser to a location that might not be expressed as a URL, such as a pointer to an item identifier list (PIDL) for an entity in the Windows Shell namespace. |
| [PutProperty](http://msdn.microsoft.com/en-us/library/aa752095%28v=VS.85%29.aspx) | Associates a user-defined name/value pair with the object. |
| [QueryStatusWB](http://msdn.microsoft.com/en-us/library/aa752096%28v=VS.85%29.aspx) | Queries the object for the status of commands using the **IOleCommandTarget** interface. |
| [Quit](http://msdn.microsoft.com/en-us/library/aa752097%28v=VS.85%29.aspx) | Closes the object. |
| [Refresh](http://msdn.microsoft.com/en-us/library/aa752098%28v=VS.85%29.aspx) | Reloads the file that is currently displayed in the object. |
| [Refresh2](http://msdn.microsoft.com/en-us/library/aa752099%28v=VS.85%29.aspx) | Reloads the file that is currently displayed with the specified refresh level. |
| [ShowBrowserBar](http://msdn.microsoft.com/en-us/library/aa752100%28v=VS.85%29.aspx) | Shows or hides a specified browser bar. |
| [Stop](http://msdn.microsoft.com/en-us/library/aa752101%28v=VS.85%29.aspx) | Cancels a pending navigation or download, and stops dynamic page elements, such as background sounds and animations. |

Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| [AddressBar](http://msdn.microsoft.com/en-us/library/aa752048%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the address bar of the object is visible or hidden. |
| [Application](http://msdn.microsoft.com/en-us/library/aa752049%28v=VS.85%29.aspx) | Gets the automation object for the application that is hosting the [WebBrowser Control](http://msdn.microsoft.com/en-us/library/aa752040%28v=VS.85%29.aspx). |
| [Busy](http://msdn.microsoft.com/en-us/library/aa752050%28v=VS.85%29.aspx) | Gets a value that indicates whether the object is engaged in a navigation or downloading operation. |
| [Container](http://msdn.microsoft.com/en-us/library/aa752051%28v=VS.85%29.aspx) | Gets an object reference to a container. |
| [Document](http://msdn.microsoft.com/en-us/library/aa752052%28v=VS.85%29.aspx) | Gets the automation object of the active document, if any. |
| [FullName](http://msdn.microsoft.com/en-us/library/aa752053%28v=VS.85%29.aspx) | Retrieves the fully qualified path of the Internet Explorer executable. |
| [FullScreen](http://msdn.microsoft.com/en-us/library/aa752054%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether Internet Explorer is in full-screen mode or normal window mode. |
| [Height](http://msdn.microsoft.com/en-us/library/aa752055%28v=VS.85%29.aspx) | Sets or gets the height of the object. |
| [HWND](http://msdn.microsoft.com/en-us/library/bb268219%28v=VS.85%29.aspx) | Gets the handle of the Internet Explorer main window. |
| [Left](http://msdn.microsoft.com/en-us/library/aa752056%28v=VS.85%29.aspx) | Sets or gets the coordinate of the left edge of the object. |
| [LocationName](http://msdn.microsoft.com/en-us/library/aa752057%28v=VS.85%29.aspx) | Retrieves the path or title of the resource that is currently displayed. |
| [LocationURL](http://msdn.microsoft.com/en-us/library/aa752058%28v=VS.85%29.aspx) | Gets the URL of the resource that is currently displayed. |
| [MenuBar](http://msdn.microsoft.com/en-us/library/aa752059%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the Internet Explorer menu bar is visible. |
| [Name](http://msdn.microsoft.com/en-us/library/aa752060%28v=VS.85%29.aspx) | Retrieves the frame name or application name of the object. |
| [Offline](http://msdn.microsoft.com/en-us/library/aa752061%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object is operating in offline mode. |
| [Parent](http://msdn.microsoft.com/en-us/library/aa752063%28v=VS.85%29.aspx) | Gets the parent of the object. |
| [Path](http://msdn.microsoft.com/en-us/library/aa752064%28v=VS.85%29.aspx) | Retrieves the system folder of the Internet Explorer executable. |
| [ReadyState](http://msdn.microsoft.com/en-us/library/aa752066%28v=VS.85%29.aspx) | Gets the ready state of the object. |
| [RegisterAsBrowser](http://msdn.microsoft.com/en-us/library/aa752069%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object is registered as a top-level browser window. |
| [RegisterAsDropTarget](http://msdn.microsoft.com/en-us/library/aa752072%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object is registered as a drop target for navigation. |
| [Resizable](http://msdn.microsoft.com/en-us/library/aa752073%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object can be resized. |
| [Silent](http://msdn.microsoft.com/en-us/library/aa752074%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object can display dialog boxes. |
| [StatusBar](http://msdn.microsoft.com/en-us/library/aa752075%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the status bar for the object is visible. |
| [StatusText](http://msdn.microsoft.com/en-us/library/aa752076%28v=VS.85%29.aspx) | Sets or gets the text in the status bar for the object. |
| [TheaterMode](http://msdn.microsoft.com/en-us/library/aa752077%28v=VS.85%29.aspx) | Sets or gets whether the object is in theater mode. |
| [ToolBar](http://msdn.microsoft.com/en-us/library/aa752078%28v=VS.85%29.aspx) | Sets or gets whether toolbars for the object are visible. |
| [Top](http://msdn.microsoft.com/en-us/library/aa752079%28v=VS.85%29.aspx) | Sets or gets the coordinate of the top edge of the object. |
| [TopLevelContainer](http://msdn.microsoft.com/en-us/library/aa752080%28v=VS.85%29.aspx) | Gets a value that indicates whether the object is a top-level container. |
| [Type](http://msdn.microsoft.com/en-us/library/aa752081%28v=VS.85%29.aspx) | Gets the user type name of the contained document object. |
| [Visible](http://msdn.microsoft.com/en-us/library/aa752082%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object is visible or hidden. |
| [Width](http://msdn.microsoft.com/en-us/library/aa752083%28v=VS.85%29.aspx) | Sets or gets the width of the object. |

**Remarks**

Internet Explorer 8. On Windows Vista, to create an instance of Internet Explorer running at a medium integrity level, pass CLSID\_InternetExplorerMedium (defined in exdisp.idl) to [CoCreateInstance](http://msdn.microsoft.com/en-us/library/ms686615%28v=VS.85%29.aspx). The resulting **InternetExplorerMedium** object supports the same events, methods, and properties as the **InternetExplorer** object.

**Examples**

The following example uses [CreateObject](http://msdn.microsoft.com/en-us/library/7t9k08y5%28VS.71%29.aspx) in Microsoft Visual Basic to launch an instance of Internet Explorer.

Dim IE As SHDocVw.InternetExplorer

Set IE = CreateObject("InternetExplorer.Application")

The following C# example launches an instance of Internet Explorer and navigates to a Web page. The code also demonstrates how to create an event handler to listen for the [BeforeNavigate2](http://msdn.microsoft.com/en-us/library/aa768326%28v=VS.85%29.aspx) event. The project requires a reference to the Microsoft Internet Controls (SHDocVw) type library.

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApplication1

{

class EventHandlers

{

public void OnBeforeNavigate2(object sender, ref object URL,

ref object Flags, ref object Target,

ref object PostData, ref object Headers,

ref bool Cancel)

{

Console.WriteLine("BeforeNavigate2 fired!");

}

}

class Program

{

static void Main(string[] args)

{

EventHandlers e = new EventHandlers();

SHDocVw.InternetExplorer IE = new SHDocVw.InternetExplorer();

object Empty = 0;

object URL = "http://www.live.com";

// override BeforeNavigate2 event

IE.BeforeNavigate2 += new

SHDocVw.DWebBrowserEvents2\_BeforeNavigate2EventHandler(

e.OnBeforeNavigate2);

IE.Visible = true;

IE.Navigate2(ref URL, ref Empty, ref Empty, ref Empty, ref Empty);

System.Threading.Thread.Sleep(5000);

IE.Quit();

}

}

}

# WebBrowser Object

Adds browsing capabilities to your applications. For the Microsoft .NET Framework version of this control, see [WebBrowser Control (Windows Forms)](http://msdn.microsoft.com/en-us/library/2te2y1x6%28v=VS.85%29.aspx).

**Members Table**

The following table lists the members exposed by the **WebBrowser** object.

Events

|  |  |
| --- | --- |
| **Event** | **Description** |
| [BeforeNavigate](http://msdn.microsoft.com/en-us/library/aa768325%28v=VS.85%29.aspx) | Fires before navigation occurs in the given object (on either a window or frameset element). |
| [BeforeNavigate2](http://msdn.microsoft.com/en-us/library/aa768326%28v=VS.85%29.aspx) | Fires before navigation occurs in the given object (on either a window element or a frameset element). |
| [ClientToHostWindow](http://msdn.microsoft.com/en-us/library/aa768327%28v=VS.85%29.aspx) | Fires to request that the client window size is converted to the host window size. |
| [CommandStateChange](http://msdn.microsoft.com/en-us/library/aa768328%28v=VS.85%29.aspx) | Fires when the enabled state of a command changes. |
| [DocumentComplete](http://msdn.microsoft.com/en-us/library/aa768329%28v=VS.85%29.aspx) | Fires when a document is completely loaded and initialized. |
| [DownloadBegin](http://msdn.microsoft.com/en-us/library/aa768330%28v=VS.85%29.aspx) | Fires when a navigation operation begins. |
| [DownloadComplete](http://msdn.microsoft.com/en-us/library/aa768331%28v=VS.85%29.aspx) | Fires when a navigation operation finishes, is halted, or fails. |
| [FileDownload](http://msdn.microsoft.com/en-us/library/bb268220%28v=VS.85%29.aspx) | Fires to indicate that a file download is about to occur. If a file download dialog box can be displayed, this event fires prior to the appearance of the dialog box. |
| [NavigateComplete](http://msdn.microsoft.com/en-us/library/aa768332%28v=VS.85%29.aspx) | Fires after a navigation to a link is completed on either a [window](http://msdn.microsoft.com/en-us/library/ms535873%28v=VS.85%29.aspx) element or a [frameSet](http://msdn.microsoft.com/en-us/library/ms535251%28v=VS.85%29.aspx) element. |
| [NavigateComplete2](http://msdn.microsoft.com/en-us/library/aa768334%28v=VS.85%29.aspx) | Fires after a navigation to a link is completed on a **window** element or a **frameSet** element. |
| [NavigateError](http://msdn.microsoft.com/en-us/library/bb268221%28v=VS.85%29.aspx) | Fires when an error occurs during navigation. |
| [NewProcess](http://msdn.microsoft.com/en-us/library/cc891504%28v=VS.85%29.aspx) | Creates a new process to handle the navigation. |
| [NewWindow](http://msdn.microsoft.com/en-us/library/aa768335%28v=VS.85%29.aspx) | Fires when a new window is to be created. |
| [NewWindow2](http://msdn.microsoft.com/en-us/library/aa768336%28v=VS.85%29.aspx) | Fires when a new window is to be created. |
| [NewWindow3](http://msdn.microsoft.com/en-us/library/aa768337%28v=VS.85%29.aspx) | Raised when a new window is to be created. Extends [NewWindow2](http://msdn.microsoft.com/en-us/library/aa768336%28v=VS.85%29.aspx) with additional information about the new window. |
| [PrintTemplateInstantiation](http://msdn.microsoft.com/en-us/library/aa768345%28v=VS.85%29.aspx) | Fires when a print template is instantiated. |
| [PrintTemplateTeardown](http://msdn.microsoft.com/en-us/library/aa768346%28v=VS.85%29.aspx) | Fires when a print template is destroyed. |
| [PrivacyImpactedStateChange](http://msdn.microsoft.com/en-us/library/bb268223%28v=VS.85%29.aspx) | Fired when an event occurs that impacts privacy, or when a user navigates away from a URL that has impacted privacy. |
| [ProgressChange](http://msdn.microsoft.com/en-us/library/aa768347%28v=VS.85%29.aspx) | Fires when the progress of a download operation is updated on the object. |
| [RedirectXDomainBlocked](http://msdn.microsoft.com/en-us/library/dd565687%28v=VS.85%29.aspx) | Fired when a cross-domain redirect request is blocked. |
| [SetPhishingFilterStatus](http://msdn.microsoft.com/en-us/library/aa741308%28v=VS.85%29.aspx) | Fires to indicate the progress and status of Microsoft Phishing Filter analysis of the current webpage. |
| [SetSecureLockIcon](http://msdn.microsoft.com/en-us/library/bb268224%28v=VS.85%29.aspx) | Fires when there is a change in encryption level. |
| [StatusTextChange](http://msdn.microsoft.com/en-us/library/aa768349%28v=VS.85%29.aspx) | Fires when the status bar text of the object has changed. |
| [ThirdPartyUrlBlocked](http://msdn.microsoft.com/en-us/library/cc848905%28v=VS.85%29.aspx) | Fired when a third-party URL is blocked. |
| [TitleChange](http://msdn.microsoft.com/en-us/library/aa768350%28v=VS.85%29.aspx) | Fires when the title of a document in the object becomes available or changes. |
| [UpdatePageStatus](http://msdn.microsoft.com/en-us/library/aa768351%28v=VS.85%29.aspx) | Not implemented. |
| [WindowClosing](http://msdn.microsoft.com/en-us/library/aa768352%28v=VS.85%29.aspx) | Fires when the window of the object is about to be closed by script. |
| [WindowSetHeight](http://msdn.microsoft.com/en-us/library/aa768353%28v=VS.85%29.aspx) | Fires when the object changes its height. |
| [WindowSetLeft](http://msdn.microsoft.com/en-us/library/aa768354%28v=VS.85%29.aspx) | Fires when the object changes its left position. |
| [WindowSetResizable](http://msdn.microsoft.com/en-us/library/aa768355%28v=VS.85%29.aspx) | Fires to indicate whether the host window should allow resizing of the object. |
| [WindowSetTop](http://msdn.microsoft.com/en-us/library/aa768356%28v=VS.85%29.aspx) | Fires when the object changes its top position. |
| [WindowSetWidth](http://msdn.microsoft.com/en-us/library/aa768357%28v=VS.85%29.aspx) | Fires when the object changes its width. |
| [WindowStateChanged](http://msdn.microsoft.com/en-us/library/aa768358%28v=VS.85%29.aspx) | Fires when the visibility state of a content window, such as the browser window or a tab, changes. |

Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| [ExecWB](http://msdn.microsoft.com/en-us/library/aa752087%28v=VS.85%29.aspx) | Executes a command and returns the status of the command execution using the [IOleCommandTarget](http://msdn.microsoft.com/en-us/library/ms683797%28v=VS.85%29.aspx) interface. |
| [GetProperty](http://msdn.microsoft.com/en-us/library/aa752088%28v=VS.85%29.aspx) | Gets the value associated with a user-defined property name. |
| [GoBack](http://msdn.microsoft.com/en-us/library/aa752089%28v=VS.85%29.aspx) | Navigates backward one item in the history list. |
| [GoForward](http://msdn.microsoft.com/en-us/library/aa752090%28v=VS.85%29.aspx) | Navigates forward one item in the history list. |
| [GoHome](http://msdn.microsoft.com/en-us/library/aa752091%28v=VS.85%29.aspx) | Navigates to the current home or start page. |
| [GoSearch](http://msdn.microsoft.com/en-us/library/aa752092%28v=VS.85%29.aspx) | Navigates to the current search page. |
| [Navigate](http://msdn.microsoft.com/en-us/library/aa752093%28v=VS.85%29.aspx) | Navigates to a resource identified by a URL or to a file identified by a full path. |
| [Navigate2](http://msdn.microsoft.com/en-us/library/aa752094%28v=VS.85%29.aspx) | Navigates the browser to a location that might not be expressed as a URL, such as a pointer to an item identifier list (PIDL) for an entity in the Windows Shell namespace. |
| [PutProperty](http://msdn.microsoft.com/en-us/library/aa752095%28v=VS.85%29.aspx) | Associates a user-defined name/value pair with the object. |
| [Refresh](http://msdn.microsoft.com/en-us/library/aa752098%28v=VS.85%29.aspx) | Reloads the file that is currently displayed in the object. |
| [Refresh2](http://msdn.microsoft.com/en-us/library/aa752099%28v=VS.85%29.aspx) | Reloads the file that is currently displayed with the specified refresh level. |
| [Stop](http://msdn.microsoft.com/en-us/library/aa752101%28v=VS.85%29.aspx) | Cancels a pending navigation or download, and stops dynamic page elements, such as background sounds and animations. |

Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| [Application](http://msdn.microsoft.com/en-us/library/aa752049%28v=VS.85%29.aspx) | Gets the automation object for the application that is hosting the [WebBrowser Control](http://msdn.microsoft.com/en-us/library/aa752040%28v=VS.85%29.aspx). |
| [Busy](http://msdn.microsoft.com/en-us/library/aa752050%28v=VS.85%29.aspx) | Gets a value that indicates whether the object is engaged in a navigation or downloading operation. |
| [Container](http://msdn.microsoft.com/en-us/library/aa752051%28v=VS.85%29.aspx) | Gets an object reference to a container. |
| [Document](http://msdn.microsoft.com/en-us/library/aa752052%28v=VS.85%29.aspx) | Gets the automation object of the active document, if any. |
| [Height](http://msdn.microsoft.com/en-us/library/aa752055%28v=VS.85%29.aspx) | Sets or gets the height of the object. |
| [Left](http://msdn.microsoft.com/en-us/library/aa752056%28v=VS.85%29.aspx) | Sets or gets the coordinate of the left edge of the object. |
| [LocationName](http://msdn.microsoft.com/en-us/library/aa752057%28v=VS.85%29.aspx) | Retrieves the path or title of the resource that is currently displayed. |
| [LocationURL](http://msdn.microsoft.com/en-us/library/aa752058%28v=VS.85%29.aspx) | Gets the URL of the resource that is currently displayed. |
| [Offline](http://msdn.microsoft.com/en-us/library/aa752061%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object is operating in offline mode. |
| [Parent](http://msdn.microsoft.com/en-us/library/aa752063%28v=VS.85%29.aspx) | Gets the parent of the object. |
| [ReadyState](http://msdn.microsoft.com/en-us/library/aa752066%28v=VS.85%29.aspx) | Gets the ready state of the object. |
| [RegisterAsBrowser](http://msdn.microsoft.com/en-us/library/aa752069%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object is registered as a top-level browser window. |
| [RegisterAsDropTarget](http://msdn.microsoft.com/en-us/library/aa752072%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object is registered as a drop target for navigation. |
| [Silent](http://msdn.microsoft.com/en-us/library/aa752074%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object can display dialog boxes. |
| [Top](http://msdn.microsoft.com/en-us/library/aa752079%28v=VS.85%29.aspx) | Sets or gets the coordinate of the top edge of the object. |
| [TopLevelContainer](http://msdn.microsoft.com/en-us/library/aa752080%28v=VS.85%29.aspx) | Gets a value that indicates whether the object is a top-level container. |
| [Type](http://msdn.microsoft.com/en-us/library/aa752081%28v=VS.85%29.aspx) | Gets the user type name of the contained document object. |
| [Visible](http://msdn.microsoft.com/en-us/library/aa752082%28v=VS.85%29.aspx) | Sets or gets a value that indicates whether the object is visible or hidden. |
| [Width](http://msdn.microsoft.com/en-us/library/aa752083%28v=VS.85%29.aspx) | Sets or gets the width of the object. |

**Remarks**

Applications that host this Microsoft ActiveX control can browse sites on the World Wide Web, as well as directories on the local computer and on network servers.