

Creating a Web Browser

By Karl Moore

Introduction

Fancy creating your own mini Web browser? How about knocking together a more personalised version of Internet Explorer?

In this mini guide, we'll be finding out how you can use the special WebBrowser control that ships with Visual Basic 5 and 6 to instantly add surfing capabilities to your application.

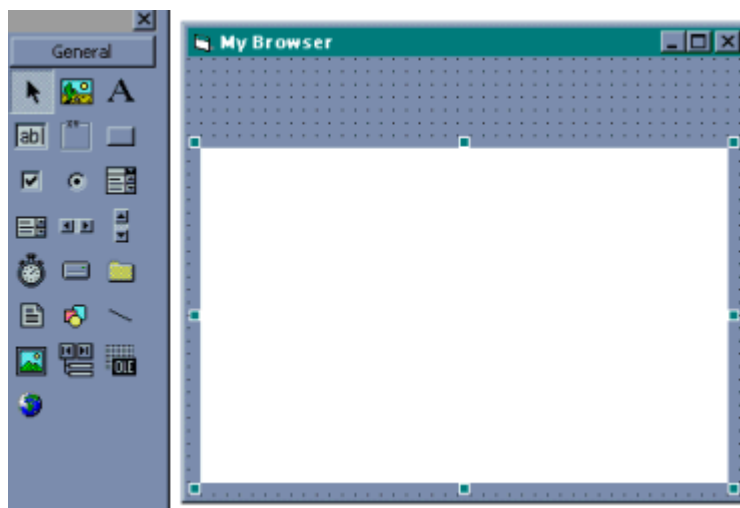
Read it, print it, use it! ;-)

Adding the Control

Before you can surf the Web within your application, you need to add the Web Browser control to your project. To do this:

- Start or open your Project
- Click 'Project', 'Components'
- Check 'Microsoft Internet Controls'
- Hit OK

To add a miniature browser to your application, drag the WebBrowser control – represented by a globe icon – out onto your form.



The white space that appears will become your browsing window. It is, in essence, the main part of the Internet Explorer screen, without the title bar or navigation buttons.

Basic Commands

The main commands for controlling your WebBrowser control are very simple. Let's cover them now:

Visiting a Site

To visit a site, call the `Navigate` method, passing a Web address. Once a site has loaded into the `WebBrowser` control, the surfer can use it just like Internet Explorer. Let's look at two quick samples for opening a site:

```
WebBrowser1.Navigate (txtWebAddress.Text)  
WebBrowser1.Navigate ("http://www.vbworld.com/")
```

Navigating a Site

Once you're at a site, you may want to move around, going backwards and forwards. You can do that easily via the `GoBack` and `GoForward` methods, like this:

```
WebBrowser1.GoBack  
WebBrowser1.GoForward
```

Refreshing and Stopping

If a site is taking too long to load, the user might want to stop it, then refresh. Or perhaps the surfer just needs to refresh and view the latest information. The `Stop` and `Refresh` methods of the `WebBrowser` control work just like Internet Explorer and can be run like this:

```
WebBrowser1.Stop  
WebBrowser1.Refresh
```

Searching for Home

To visit the default home page, you can run the simple `GoHome` method. And if you want to display the default search page, try out the `GoSearch` method. Here's an example:

```
WebBrowser1.GoHome  
WebBrowser1.GoSearch
```

Handling Errors

It's worth noting that a lot of `WebBrowser` commands are prone to silly menial errors, so feel free to use 'On Error Resume Next' statements liberally.

Properties and Events

There are two properties and one main event that will help you build Internet-aware applications with the `WebBrowser` control:

Location Name

When you're at a site, you can retrieve the title – as would typically appear in the Internet Explorer title bar caption – by retrieving the `LocationName` property. Here's an example:

```
MsgBox WebBrowser1.LocationName
```

Location URL

You can retrieve the current Web address by grabbing the `LocationURL` property, as would typically appear in the Internet Explorer Address bar. Here's an example:

```
MsgBox WebBrowser1.LocationURL
```

Document Complete

The DocumentComplete event of the WebBrowser control fires up when any Web page has finished loading. This is the ideal event under which to write code to update possible title and address bars. Here's an example:

```
Private Sub WebBrowser1_DocumentComplete(ByVal pDisp As _  
Object, URL As Variant)  
    Me.Caption = WebBrowser1.LocationName  
    txtWebAddress.Text = WebBrowser1.LocationURL  
End Sub
```

Progress Bars

If you fancy adding a Progress Bar to indicate how fast your Web page commands are moving along, you're in luck – the WebBrowser control has an event designed specially for this purpose.

To use, you first need to add a Progress Bar to your application. This control ships with Visual Basic and can be added to your project by clicking 'Projects', 'Components' and selecting 'Microsoft Windows Common Controls 6.0'.

Here's a piece of sample code that uses the WebBrowser ProgressChange event to alter a simple Progress Bar:

```
Private Sub WebBrowser1_ProgressChange(ByVal _  
Progress As Long, ByVal ProgressMax As Long)  
  
On Error Resume Next  
ProgressBar1.Max = ProgressMax  
ProgressBar1.Value = Progress  
ProgressBar1.Refresh  
  
End Sub
```

After inserting this code, try visiting a Web site or clicking a link – then monitor the progress visually.

Restricting Web sites

If you're creating a custom browser, you may wish to restrict certain Web sites. You do this by intercepting the request to visit a Web site with the BeforeNavigate2 event.

Here's a chunk of sample code that doesn't allow the user to visit playboy.com:

```
Private Sub WebBrowser1_BeforeNavigate2(ByVal pDisp As Object, _  
URL As Variant, Flags As Variant, TargetFrameName As Variant, _  
PostData As Variant, Headers As Variant, Cancel As Boolean)  
  
    If InStr(1, URL, "playboy.com") Then  
        Cancel = True  
        MsgBox "Sorry, that site is restricted!"  
    End If  
  
End Sub
```

This works by checking the passed URL (Web Address). If it contains the string 'playboy.com', the action is cancelled and a message box displayed. Otherwise, if everything is A-OK, access to the site is allowed.

It's worth noting that this code will work equally as well in either the BeforeNavigate or BeforeNavigate2 events. Users of Visual Basic 6 will only have BeforeNavigate2, whilst Visual Basic 5 folk may have both. There is very little different between the two methods. For more information, consult the help file.

Working with Combo Boxes

As most users are acquainted with typing their Web addresses into Combo Boxes, you may want to use the same.

However Internet Explorer allows you to either press the 'Go' button to visit the typed site, or merely hit return. You can detect if return has been pressed like this:

```
Private Sub Combo1_KeyPress(KeyAscii As Integer)

    If KeyAscii = vbKeyReturn Then
        WebBrowser1.Navigate (Combo1.Text)
    End If

End Sub
```

It's also possible to turn a regular Text Box into an Internet Explorer 5 style Web address list with the help of a very sneaky API call. [Click here](#) to learn more.

Conclusion

In this brief printable guide, we covered all the main features of the WebBrowser control.

We discovered how to visit sites, navigate backwards and forwards between Web pages, use Progress Bars, restrict naughty Net stops, plus enhance a basic Combo Box.

As you've found out, the WebBrowser control is an easy and simple way to add Internet capabilities to any application. I hope this article is helpful in wiring up your programs.

And on that note, I'll wish you good luck – and good-bye!