Lab 11.1: Using the Web Browser Control

Objectives

After completing this lab, you will be able to:

- Create an MFC application that is a control container.
- Use the Developer Studio Gallery to include the Web Browser control into your project.
- Programmatically create a Web Browser control and set properties and invoke methods of the Web Browser control.
- Handle events of the Web Browser control in your application.

Prerequisites

Before attempting this lab, you should have completed Chapter 11 through the section titled "The Internet Explorer Object and Web Browser Control."

Lab Setup

This demonstration shows what you will accomplish during the lab.

Estimated time to complete this lab: **40 minutes**.

Exercises

The following exercise provides practice working with the concepts and techniques covered in this chapter.

Exercise 1: Using the Web Browser Control

In this exercise, you will create a simple a control container application and use the Developer Studio Gallery to add the Web Browser control to the project. You also will add code to create an instance of the Web Browser control and to invoke a method and set properties of the control.

Exercise 2: Handling Web Browser Control Events

In this exercise, you will modify the application created in Exercise 1 of this lab to handle a Web Browser control event.

Before you start this lab, you should have installed Internet Explorer and have an account with an Internet service provider or access to the Internet via a corporate firewall.

The completed code for these exercises is in \Labs\C11\Lab01\Xxx, where Xxx is the exercise number.

Exercise 1: Using the Web Browser Control

In this exercise, you will create a simple a control container application and use the Microsoft Developer Studio Gallery to add the Web Browser control to the project. You also will add code to create an instance of the Web Browser control and to invoke a method and set properties of the control.

➢ **Create a Control Container Application**

1. Start Microsoft Developer Studio and then from the File menu, click New.
2. In the New dialog box, click the Projects tab, and then select MFC AppWizard (EXE).
3. Type the directory path under which you wish to create the project in the Location field, type **Browse** in the Project Name field, accept the default option Create new workspace, then click OK.
4. In Step 1 of MFC AppWizard, select Single Document for the type of application to create, then click Next.
5. In Step 2 of MFC AppWizard, accept the default settings and click Next.

6. In Step 3 of MFC AppWizard, accept the default of None for compound document support and under other support, make sure that the ActiveX Controls check box is checked. You can accept defaults for the remaining steps, so click Finish.
7. To have MFC AppWizard finish creating the application, in the New Project Information box, click OK.
8. In the Developer Studio ClassView pane, expand the Browse classes to view the list of classes supplied by MFC AppWizard.

➢ **Add the Web Browser Control to Your Project**

1. From the Project menu, choose Add To Project, and then choose Components and Controls. In the Gallery dialog box, double-click the Registered ActiveX Controls folder.
2. Select the Microsoft Web Browser Control component, choose Insert and click OK.
3. In the Confirm Classes dialog box, click OK.
4. In The Gallery dialog box, click Close. Note that the **CWebBrowser** class has been added to the list of classes in the Browse application.
5. In the Developer Studio ClassView pane, expand the **CWebBrowser** class to view properties and methods of the class.

➢ **Create an Instance of the Web Browser Control**

1. Before you create an instance of the Web Browser control, you will need to include the header file for the **CWebBrowser** class in several of your application's implementation files.
   a. Use Developer Studio to open the file Browse. Cpp. After `#include "stdafx.h"`, add this line:

      ```
      #include "webbrowser.h"
      ```

   b. Use Developer Studio to open the file BrowseView.Cpp. After `#include "stdafx.h"`, add this line:

      ```
      #include "webbrowser.h"
      ```

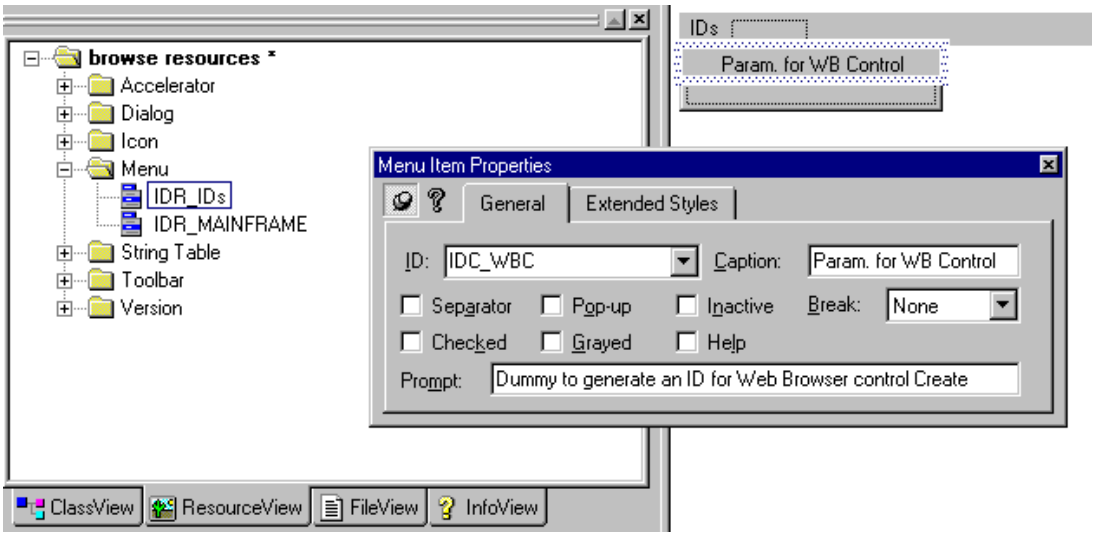
2. You also need to add a member variable for a pointer to a Web Browser control to the class in which you will create the control. In this case, you will embed the control in the view class.
   a. In the Developer Studio ClassView pane, right-click **CBrowseView**, then click Add Member Variable.
   b. In the Add Member Variable dialog box, type **CWebBrowser*** in the Variable Type field.
   c. In the Add Member Variable dialog box, type **m_pBrowse** in the Variable Declaration field.
   d. Accept the default of Public access and click OK.
   e. Use Developer Studio to open the file BrowseView.Cpp, and inside the constructor, **CBrowseView::CBrowseView**, add this line:

      ```
      m_pBrowse = NULL;
      ```

3. Create a control ID to use as a parameter for the Web Browser control **Create** function.

   The **Create** function for the Web Browser control requires a control identifier as a parameter. One way to create a control ID is to insert a new menu resource into your project and then add a menu item for the control ID to be used for the parameter to the **Create** function. This new resource is not visible to the user; it is just an easy way to create IDs.
   a. In the ResourceView pane of your Project Workspace, right-click the Menu folder and from the context menu, select Insert menu.
   b. In the Menu editor pane, double-click the empty top-level menu item provided and in the Menu Item Properties dialog box, set the Caption property to **IDs**.
   c. In the Menu editor pane, double-click the empty menu item under IDs. In the Menu Item Properties dialog box, set the ID property to **IDC_WBC**. This is the control ID you will provide as a parameter to the Web Browser control's **Create** function in the next step.

      You can add caption and prompt information to help you remember what this resource is for.



4. To embed the control in the view, add a message handler for WM_CREATE to the view class, and place the code to create an instance of the Web Browser control in the message handler.
   a. In the Developer Studio ClassView pane, right-click **CBrowseView**, and then click Add Windows Message Handler.
   b. From the New Windows messages/events list, select WM_CREATE, then click the Add and Edit button.
   c. Inside the **CBrowseView::OnCreate** function, before the return statement, add the following code.

```
// Define the area where the control will reside.
CRect rect;
GetClientRect(&rect);

// Create the control.
// IDC_WBC is a unique identifier for the control and was defined
// in Step 3 using a dummy menu item.
m_pBrowse = new CWebBrowser;
ASSERT(m_pBrowse);
if (!m_pBrowse->Create(NULL,NULL,WS_VISIBLE,rect,this,IDC_WBC))
{
    TRACE("failed to create browser\n");
    delete m_pBrowse;
    m_pBrowse = NULL;
    return 0;
}
```

You will not be able to see any results from the code you have added until you have completed the next two steps of invoking a method and setting properties of the control.

➢ **Invoke a Method of the Web Browser Control**

Following the lines added to the **CBrowseView::OnCreate** function in the previous procedure, add the code to invoke the Web Browser control's **Navigate** method.

```
// Initialize the first URL.
COleVariant noArg;
m_pBrowse->Navigate("www.microsoft.com", &noArg, &noArg, &noArg,
                    &noArg);
```

Note The empty **COleVariant** arguments can be set to further customize the Navigate method. For more information on **COleVariant** arguments, refer to the article "Automation" in the Visual C++ Programmer's Guide, in the Microsoft Visual C++ online documentation.

➢ **Set Properties of the Web Browser Control**

To set properties of the control, add a message handler for WM_SIZE to the view class, and place the code to set the height and width properties of the Web Browser control in the message handler.

1. In the Developer Studio class view pane, right-click **CBrowseView**, and then click Add Windows Message Handler.
2. From the New Windows messages/events list, select WM_SIZE, then click the Add and Edit button.
3. Inside the **CBrowseView::OnSize** function, after the call to the base class's **OnSize** function, add the following code.

```
if (m_pBrowse)
{
    m_pBrowse->SetWidth(cx);
    m_pBrowse->SetHeight(cy);
}
```

This code causes the Web Browser control to resize when a user resizes the application's mainframe window.

➢ **Build and test your application**

If you have an active connection to the Internet, you should see the default page for the Microsoft Corporation's Web site displayed in the Web Browser control in your application's view class. The completed code for this exercise is in \Labs\C11\Lab01\Ex01.

Exercise 2: Handling Web Browser Control Events

In this exercise, you will modify the application from Exercise 1 of this lab so that it handles a Web Browser control event.

Continue with the files you created in Exercise 1 or, if you do not have a starting point for this exercise, the code that forms the basis for this exercise is in \Labs\C11\Lab01\Ex01.

There are three main steps to handle events:

1. Determine the function prototype and ID of the event.
2. Add the event function handler to the wrapper class.
3. Link the event to the handler.

➢ **Determine the function prototype and ID of the event**

1. From the Developer Studio Tools menu, choose OLE/COM Object Viewer.
2. In the left-hand pane of the OLE/COM Object Viewer dialog box, double-click the Controls folder. Scroll down the list and right-click the Microsoft Web Browser Control.
3. From the menu that appears, select View Type Information.
4. To view Web Browser control events, in the ITypeLib Viewer dialog box that appears, double-click dispInterface DWebBrowserEvents. Now double-click the Methods folder. This displays the event methods for the Web Browser control.

If you do not see the dispInterfaceDWebBrowserEvents node in the left-hand pane of the ITypeLib Viewer dialog box, on the View menu, see whether "Group by type kind" is checked. If so, click Group by type kind to uncheck it and the dispInterfaceDWebBrowserEvents node will appear.

---

5. Under the Methods folder, click StatusTextChange. The information you require about the ID for the event and the prototype for the event handler is displayed in the right-hand pane of the ITypeLib Viewer dialog box.

```
[id(0x00000066)
void StatusTextChange([in] BSTR Text);
```

➢ **Add the event function handler to the control's wrapper class**

1. Before you create the **OnStatusTextChange** event handler, create a public member function, **Status**, in the **CMainFrame** class. This helper function, which enables the view class to access the status bar object, will be called in the **OnStatusTextChange** event handler.
   a. In the Developer Studio ClassView pane, right-click **CMainFrame**, then click Add Member Function.
   b. In the Add Member Function dialog box, type **void** in the Function Type field, type **Status(LPCTSTR text)** in the Function Declaration field, then click OK.
   c. Add this line inside the **Status** function:

```
m_wndStatusBar.SetWindowText(text);
```

2. You can now add the **OnStatusTextChange** event handler function. In the Developer Studio ClassView pane, right-click **CBrowseView**, then click Add Member Function.
3. In the Add Member Function dialog box, type **void** in the Function Type field, type **OnStatusTextChange(LPCTSTR text)** in the Function Declaration field, then click OK.
4. Add these lines inside the **OnStatusTextChange** function:

```
// Only write to status line if there is a non-zero string.
// (lstrlen works on either ANSI or UNICODE strings.)
if (lstrlen(text))
    ((CMainFrame*)AfxGetMainWnd())->Status(text);
```

AfxGetMainWnd returns a **CFrameWnd** pointer. You must cast the return value to a **CMainFrame** pointer because **Status** is a member function of the **CMainFrame** class.

---

5. Because the **Status** function called in Step 4 of this procedure is a member of the **CMainFrame** class, add a line near the top of the BrowseView.Cpp file to include the header file for the **CMainFrame** class:

```
#include "mainfrm.h"
```

➢ **Link the event to the handler**

In our application, the view class handles the OnStatusTextChange event, so the changes outlined below are made in each case to the appropriate view class file(s).

1. Add a DECLARE_EVENTSINK_MAP() macro call to the header file for the class that handles the event.
   a. Use the Developer Studio editor to open BrowseView.H.
   b. After the call to `DECLARE_MESSAGE_MAP()`, add this line:

```
DECLARE_EVENTSINK_MAP()
```

2. Add an event sink map to the source file for the class that handles the event. Recall that the event ID displayed in the OLE/COM Object Viewer for the StatusTextChange event was 0x66.
   a. If you do not have BrowseView.Cpp open, use the Developer Studio editor to open it.
   b. After the line `END_MESSAGE_MAP()`, add these lines to add an event sink map to the view class:

```
BEGIN_EVENTSINK_MAP(CBrowseView, CView)
    ON_EVENT(CBrowseView, 0, 0x66, OnStatusTextChange, VTS_BSTR)
END_EVENTSINK_MAP()
```

➢ **Build and test your application**

Notice that the text in the status bar changes as the Web Browser control navigation takes place. The completed code for this exercise is in \Labs\C11\Lab01\Ex02.