Desktop Development » Miscellaneous » Miscellaneous Controls

License: The Code Project Open License (CPOL)

C#2.0.NET2.0, Win2K, WinXP, Win2003, VistaVS2005, Dev

The most complete C# Webbrowser wrapper control

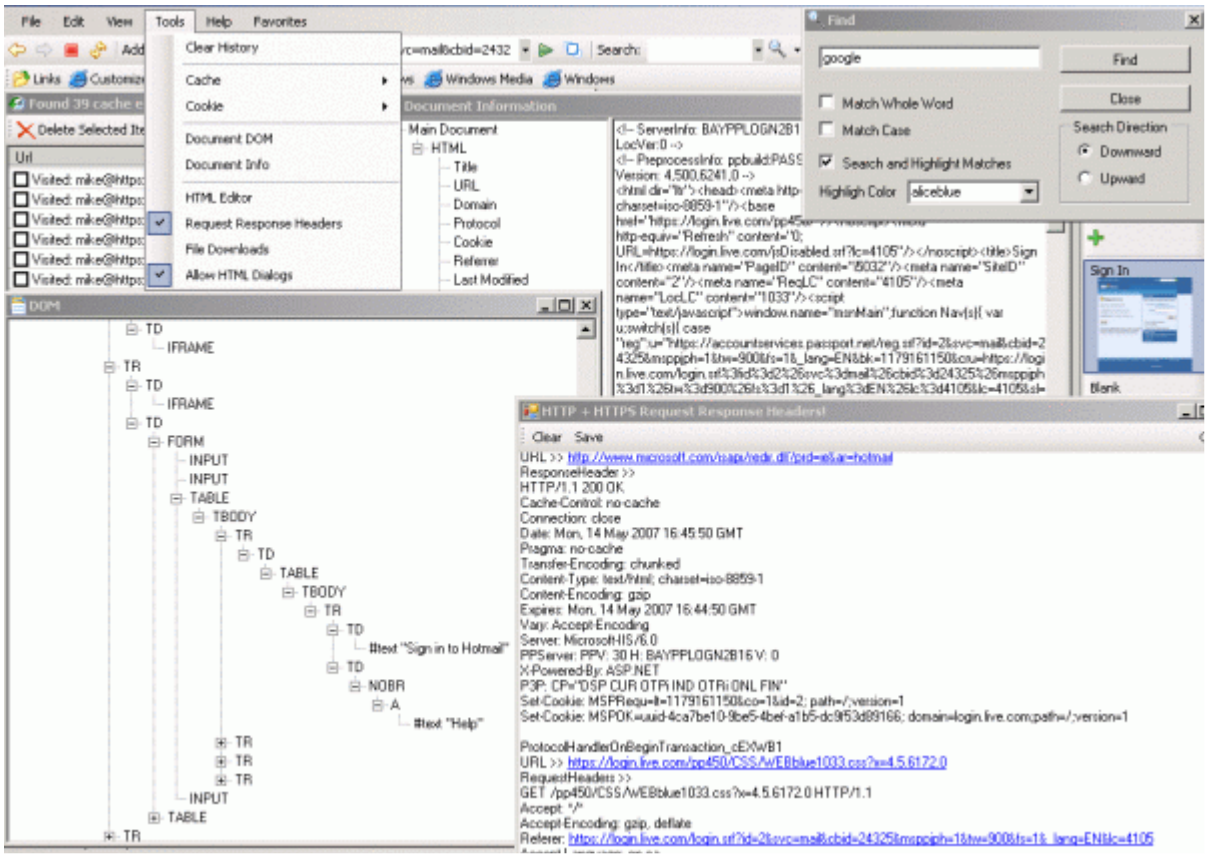
By **mehrcpp**

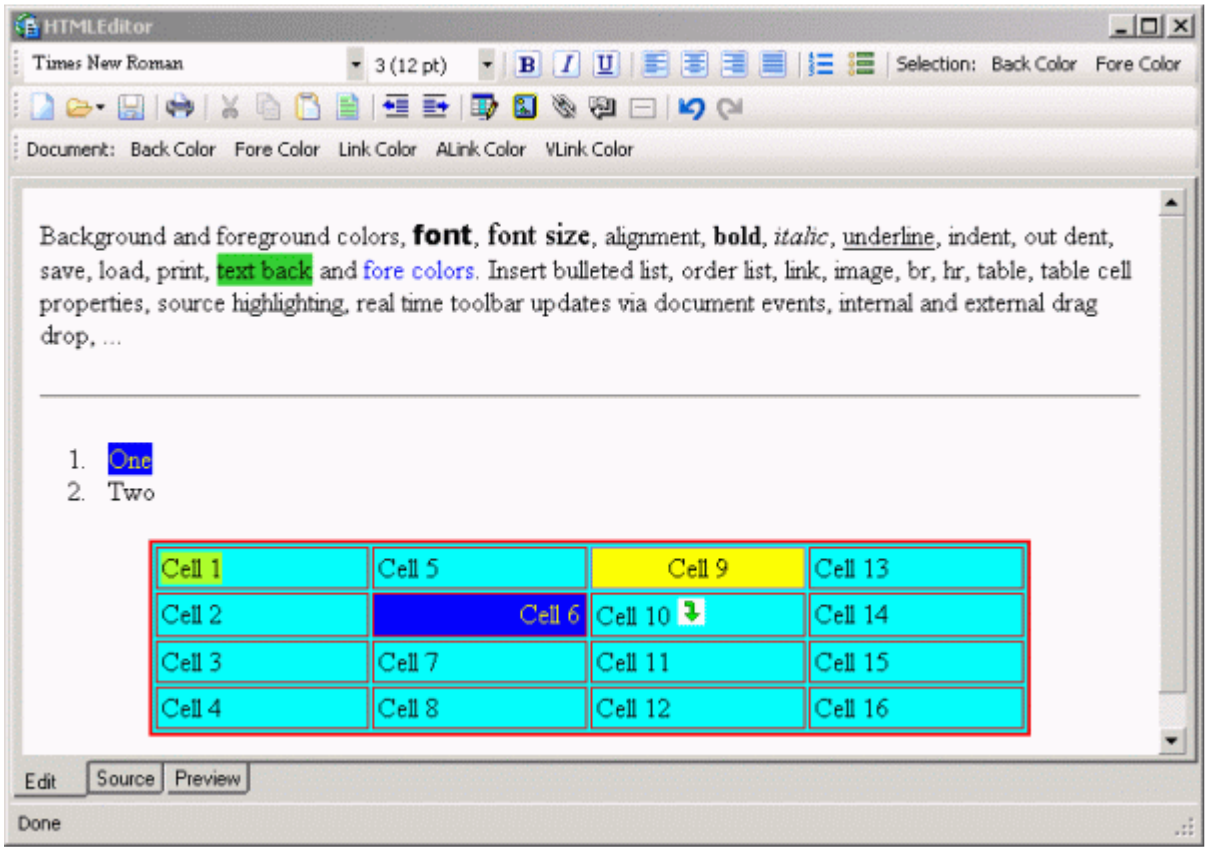
Posted: **25 May 2007**
Updated: **28 May 2007**
Views: **365,762**
Bookmarked: **300 times**

A C# (.NET 2.0) control which creates, hosts, and offers advanced customization such as dragdrop, file downloads, HTTP/S header viewing, and much more.

91 votes for this article.
Popularity: 9.25 Rating: **4.72** out of 5
1 2 3 4 5

Download control, demo source and binaries - 1,110.8 Kb





Introduction

csEXWB is a C# .NET 2.0 control that creates, hosts and sinks the events of the original Webbrowser control (Not .NET or any other wrapper). Advanced customization and total control over the Webbrowser control are achieved via implementation of a number of interfaces, along with the addition of many methods, properties, events and a COM library. The control has no dependencies on MSHTML interop. All of the interfaces, enums and structs are defined within the project. Here are some of the features of this control:

- Can easily be extended by adding other interfaces and methods, unlike .NET or any other Webbrowser wrapper
- All the basics expected from a Webbrowser wrapper are implemented. Setting/getting UI, and DL control flags. Handling of context menus, keyboard, registry, security manager, script errors, popups, authentication, security problems, font size, message boxes
- Take over file downloads
- Stop dialogs launched using `showModelessDialog()` and `showModalDialog()` JavaScript methods
- Monitor all HTTP and HTTPS request and response headers with the opportunity to add your own headers. This includes images, sounds, scripts, etc.
- Built in `HTMLDocument` and `HTMLWindow` events
- Frameset aware. All the relevant routines check for frames
- Full drag and drop support
- A comprehensive demo that demonstrates how to use most of the functionality offered by this control beyond the basics. An HTTP and HTTPS request and response header viewer which can be used for web application development. A complete DOM viewer. Multi tab simulation. Cookies and Cache viewer and remover, per site or all, A simple file download viewer. A demo of how to retrieve information from document object. An HTML editor with edit, source, and preview panes. Many editing functionalities, internal and external drag drop, font, color, inserting items such as table, reading and setting properties such as table cell, etc.
- Refresh begin and end events
- Create and resolve Internet shortcuts using `IShellLink` interface
- Load HTML content specifying `BaseURL`
- Added functionality to use the control as an MSHTML editor
- MSHTML as a UI-less HTML parser helper class

Control

This is a basic class derived from `Control` class. It overrides a number of `Control` methods in order to create, host and dispose the Webbrowser control. In addition, it handles resize events and forwards keyboard, focus, refresh and visibility events to the hosted Webbrowser control.

csExWBDLMan COM library

`csExWBDLMan` is written in VC++ 6.0 using ATL 3.0. It is compiled with minimum dependencies, i.e. no MFC, `std::`, `CString`. The library is used to allow the client to employ the `IDownloadManager` implementation of `csExWBDLMan` library. In addition, the library implements the PassthroughAPP package by Igor Tandetnik, which enables the client to intercept all HTTP and HTTPS request and responses. The last functionality offered by the library is the ability to set any of the available windows hooks with the option to cancel a call. This functionality is used to stop dialogs launched using the `showModelessDialog()` and `showModalDialog()` methods. The library can be used in any other project that needs to set a global windows hook or view HTTP and HTTPS headers, as long as a valid window handle is supplied.

Before opening the solution or running the demo, you need to register this library

- Copy *csExWBDLMan.dll* located in the csExWB\COM_Component_Source_Binaries\ReleaseMinDependency sub folder to your system directory
- Register *csExWBDLMan.dll* using *regsvr32.exe*
- Open the solution

Example: Assuming the system dir path is 'C:\windows\system32\'

```
regsvr32.exe C:\windows\system32\csExWBDLMan.dll
```

Interfaces

Each interface mentioned below plays a role in either the hosting or customization process. I decided not to go through a detailed explanation of how to implement and use the interfaces. One can easily find all of the information regarding implementation and functionalities offered by these interfaces from MSDN, or right here on this site.

- `IOleClientSite` - required as part of Webbrowser control hosting interfaces
- `IOleInPlaceSite` - required as part of Webbrowser control hosting interfaces
- `IDocHostShowUI` - to handle `::ShowMessage`
- `IDocHostUIHandler` - to handle `::ShowContextMenu`, `::TranslateAccelerator`, `::GetOptionKeyPath`, `::GetDropTarget` and `::GetHostInfo`
- `DWebBrowserEvents2` - to handle Webbrowser events
- `IDropTarget` - to handle full drag drop internally
- `IServiceProvider` - to handle `::QueryService`
- `IHttpSecurity` - to handle HTTP-related security problems
- `IWindowForBindingUI` - returns a handle via `::GetWindow` that is used by MSHTML to display information in client UI
- `INewWindowManager` - WinX sp2 and up, to handle popups
- `IAuthenticate` - to handle basic and NTLM authentication
- `IOleCommandTarget` - to intercept script errors via the `::Exec` method
- `IinternetSecurityManager` - to handle `::ProcessUrlAction`
- `IHTMLEventCallBack` - A callback mechanism to pass `HTMLDocument` and `HTMLWindow` events to the control and, in turn, to the client
- `IProtectFocus` - IE7 + Vista - Queries for permission to grab the focus when loading the page or when a script attempts to focus an element
- `IHTMLOMWindowServices` - XP sp2 - Enables applications that are hosting the WebBrowser Control to intercept the window object calls for manipulating the host window coordinates from Dynamic HTML (DHTML) script

Properties

- `WBDOCDOWNLOADCTLFLAG` - `get set` - `DOC_DOWNLOAD_CONTROL_FLAGS`

- `DownloadImages` - `get set` - shortcut for `DOCDOWNLOADCTLFLAG.DLIMAGES`
- `DownloadSounds` - `get set` - shortcut for `DOCDOWNLOADCTLFLAG.BGSOUNDS`
- `DownloadActiveX` - `get set` - shortcut for `DOCDOWNLOADCTLFLAG.NO_DLACTIVEXCTLS`
- `DownloadJava` - `get set` - shortcut for `DOCDOWNLOADCTLFLAG.NO_JAVA`
- `DownloadFrames` - `get set` - shortcut for `DOCDOWNLOADCTLFLAG.NO_FRAMEDOWNLOAD`
- `DownloadScripts` - `get set` - shortcut for `DOCDOWNLOADCTLFLAG.NO_SCRIPTS`
- `WBDOCHOSTUIFLAG` - `get set` - `DOC_HOST_UI_FLAGS`
- `Border3DEnabled` - `get set` - Webbrowser 3D border
- `ScrollBarsEnabled` - `get set` - Webbrowser Scrollbars
- `WBDOCHOSTUIDBLCLK` - `get set` - `DOC_HOST_UI_DBLCLK`
- `TextSize` - `get set` - Webbrowser zoom property
- `CanGoBack` - `get` - can navigate backward
- `CanGoForward` - `get` - can navigate forward
- `WebbrowserObject` - `get` - Document object of Webbrowser control
- `SendSourceOnDocumentCompleteWBEx` - `get set` - fires `DocumentCompleteEX` event rather than `DocumentComplete`. `DocumentCompleteEX` event has one additional parameter that contains the source of the `pDisp` document object at the time `DocmentComplete` was called
- `IEServerHwnd` - `get` - Internet `Explorer_Server` HWND
- `ShellEmbedingHwnd` - `get` - `ShellEmbedding` HWND
- `ShellDocObjectHwnd` - `get` - `ShellDocObject` HWND
- `RegisterAsDropTarget` - `get set` - Webbrowser default drag drop
- `RegisterForInternalDragDrop` - `get set` - Instructs the control to take over drag drop. Uses `WBDragxxx` and `WBDropxxx` events to notify the client
- `RegisterAsBrowser` - `get set` - Registers Webbrowser as a top-level browser for target name resolution
- `Silent` - `get set` - whether the Webbrowser control can show dialog boxes
- `LocationName` - `get` - name of the resource that Webbrowser control is currently displaying
- `LocationUrl` - `get set` - URL of the resource that Webbrowser control is currently displaying
- `Busy` - `get` - indicating whether the Webbrowser control is engaged in a navigation or downloading operation
- `OffLine` - `get set` - currently operating in offline mode
- `ReadyState` - `get` - retrieves `Readystate` of the Webbrowser control
- `ThumbImage` - `get` - Contains a thumb image of the Webbrowser or `null`
- `ObjectForScripting` property. Allows JavaScript functions in an HTML page to call methods and properties of a an instance class passed to this property. Same as `ObjectForScripting` of C# Webbrowser wrapper control. An example of how to use this property has been provided in `WinExternal` class of `frmMain`
- `DocumentTitle` - `Set`s or retrieves the title of the document
- `DocumentSource` - `Set`s or retrieves the HTML source of the document
- `UseInternalDownloadManager` - `Set` to `true`, default, to allow the control to take over file downloads. `FileDownloadExxxx` events are fired instead of `FileDownload` event. This functionality is achieved via COM library which implements `IDownloadManager` interface. The download routines account for redirect and `Content_Disposition` header
- `FileDownloadDirectory` Default file download directory. `Set` to users *MyDocuments* folder by default. Used only if `UseInternalDownloadManager` property is set to `true`

Methods

Basic

These methods are self explanatory

- `public void Navigate(string URL)`
- `public void Navigate(string URL, BrowserNavConstants Flags)`
- `public void Navigate(string URL, BrowserNavConstants Flags, string TargetFrameName)`
- `public void Navigate(string URL, BrowserNavConstants Flags, string PostData)`
- `public void Navigate(string URL, string PostData)`
- `public void Navigate(string URL, byte[] PostData)`
- `public void Navigate2(string URL)`
- `public void Stop()`

- `public bool GoBack()`
- `public bool GoForward()`
- `public void GoHome()`
- `public void GoSearch()`
- `public void Refresh2(RefreshConstants Level)`
- `public bool SelectAll()`
- `public bool Clear()`
- `public bool ClearSelection()`
- `public bool Copy()`
- `public bool Paste()`
- `public bool Cut()`
- `public bool Undo()`
- `public bool Redo()`
- `public bool Delete()`
- `public bool PasteSpecial()`
- `public bool Spell()`
- `public bool NewWindow()`
- `public bool Print()`
- `public bool Print2()`
- `public bool Properties()`
- `public bool PrintPreview()`
- `public bool PrintPreview2()`
- `public bool PageSetup()`
- `public void SaveAs()`
- `public bool Find()`
- `public bool IEOptions()`
- `public bool ViewSource()`
- `public void OrganizeFavorites()`
- `public void PrivacySettings()`
- `public void LanguageDialog()`
- `public void ProgramAccessAndDefaults()`
- `public void AddToFavorites()`
- `public void ImportExport()`
- `public void SendLinkByEmail()`
- `public void SendPageByEmail()`
- `public void SendShortcutToDesktop()`

Extended

Note: In a frameset, the `bTopLevel` parameter determines whether to use the top level document or to attempt to find and use the active document

- `public bool LoadUrlIntoBrowser(String url)` - Loads a URL into browser using `IPersistMoniker` interface
- `public bool LoadHtmlIntoBrowser(string html, string sBaseUrl)` - Loads HTML content into browser using `LoadHTMLMoniker` class and `IPersistMoniker` interface, allowing client to set the `baseurl`
- `public bool LoadHtmlIntoBrowser(string html)` - Loads HTML content into browser using `IPersistStreamInit` interface, `baseurl`, is set to `about:blank` by MSHTML
- `public void ShowCertificateDialog()` - If available, displays certificate for current website
- `public Image DrawThumb(int W, int H, System.Drawing.Imaging.PixelFormat pixFormat)` - Uses IE_Server HWND to draw a thumb image of the Webbrowser control. Faster than other methods with one draw back. It only works if the Webbrowser control is in front of Zorder
- `public Image DrawThumb2(int W, int H, System.Drawing.Imaging.PixelFormat pixFormat)` - Uses `IViewObject` interface obtained from `IHTMLDocument2` to draw a thumb image of the Webbrowser control
- `public void SaveBrowserImage(string sFileName, System.Drawing.Imaging.PixelFormat pixFormat, System.Drawing.Imaging.ImageFormat format)` - Saves Webbrowser image
- `public bool HasFocus()`

- `public void SetFocus()`
- `public void SetFocusBody()`
- `public bool FindInPage(string sFind, bool DownWard, bool MatchWholeWord, bool MatchCase, bool ScrollIntoView)` - Finds a match. Returns `true` if found, else `false`. In a frameset, it attempts to find and use the active document.
- `public int FindAndHightAllInPage(string sFind, bool MatchWholeWord, bool MatchCase, int cbackColor, int cForeColor)` - Finds and highlights all matches. Returns number of matches found. In a frameset, it attempts to find and use the active document.
- `public int FindAndHightAllInPage(string sFind, bool MatchWholeWord, bool MatchCase, string cbackColor, string cForeColor)`
- `public int FindAndHightAllInPage(string sFind, bool MatchWholeWord, bool MatchCase, Color cbackColor, Color cForeColor)`
- `public bool IsCommandEnabled(string sCmdId)` - Wrapper for `IHTMLDocument2::queryCommandEnabled`
- `public bool SetDesignMode(string sMode)`
- `public string GetDesignMode()`
- `public IHTMLElement GetActiveElement()` - Returns the active element or `null`. Accounts for frames
- `public IHTMLDocument2 GetActiveDocument()` - Returns the active document or `null`. Accounts for frames
- `public string GetTitle(bool bTopLevel)` - Wrapper for `IHtmlDocument2::title`
- `public string GetTitle(IWebBrowser2 thisBrowser)`
- `public string GetText(bool bTopLevel)` - Wrapper for `IHTMLDocument3::outerText`
- `public string GetText(IWebBrowser2 thisBrowser)`
- `public string GetSource(bool bTopLevel)` - Wrapper for `IHTMLDocument3::outerHTML`
- `public string GetSource(IWebBrowser2 thisBrowser)`
- `public IHTMLElementCollection GetImages(bool bTopLevel)` - Wrapper for `IHTMLDocument2::images`
- `public IHTMLElementCollection GetAnchors(bool bTopLevel)` - Wrapper for `IHTMLDocument2::anchors`
- `public string GetSelectedText(bool bTopLevel, bool ReturnAsHTML)` - Returns selection as plain text or HTML.
- `public IHTMLElement ElementFromPoint(bool bTopLevel, int X, int Y)` - Wrapper for `IHTMLDocument2::elementFromPoint`
- `public bool ExecCommand(bool bTopLevel, string CmdId)` - Wrapper for `IHTMLDocument2::execCommand`
- `public bool QueryCommandState(bool bTopLevel, string CmdId)` - Wrapper for `IHTMLDocument2::queryCommandState`
- `public bool OleCommandExec(bool bTopLevel, MSHTML_COMMAND_IDS CmdID)` - Wrapper for `IHTMLDocument2::IOleCommandTarget::Exec`
- `public object QueryCommandValue(string CmdID)` - Wrapper for `IHTMLDocument2::queryCommandValue`
- `public bool QueryCommandState(bool bTopLevel, string CmdId)` - Wrapper for `IHTMLDocument2::queryCommandState`
- `public IHTMLElement GetElementByID(bool bTopLevel, string idval)` - Wrapper for `IHTMLDocument3::getElementById`
- `public IHTMLElementCollection GetElementsByTagName(bool bTopLevel, String tagname)` - Wrapper for `IHTMLDocument3::getElementsByTagName`
- `public IHTMLElementCollection GetElementsByName(bool bTopLevel, string elemname)` - Wrapper for `IHTMLDocument3::getElementsByName`
- `public object execScript(bool bTopLevel, string ScriptName, string ScriptLanguage)` - Wrapper for `IHTMLWindow2::execScript`
- `public bool OleCommandExec(bool bTopLevel, MSHTML_COMMAND_IDS CmdID, object pvaIn)` - Wrapper for `IHTMLWindow2::execScript`, which accepts a parameter
- `public object InvokeScript(string ScriptName, object[] Data)` - Invokes a script within the HTML page
- `public object InvokeScript(IWebBrowser2 wb, string ScriptName, object[] Data)` - Invokes a script within the HTML page
- `public bool IsFrameset()`
- `public int FramesCount()`

- `public List<IWebBrowser2> GetFrames()` - Returns a List populated with the `IWebbrowser` interfaces of the frames
- `public bool CreateInternetShortCut(string LocalFileName, string URL, string Description, string IconFileName, int IconIndex)` - Attempts to create an Internet shortcut
- `public string ResolveInternetShortCut(string InternetShortCutPath)` - Attempts to resolve an Internet shortcut
- `public bool ClearHistory()` - Clears IE history
- `public void ActivateHTMLEvents(HTMLEventType EventType, int[] HTMLEventDispIds)` - Activates either `HTMLdocument` or `HTMLWindow` events
- `public void DeactivateHTMLEvents(HTMLEventType EventType)` - Deactivates previously activated `HTMLDocument` or `HTMLwindow` events
- `public int DownloadFile(string Url)` Attempts to download a file asynch. `FileDownloadExXXX` events are used for notifications. Return value is a unique ID for this download that can be used to stop the download
- `public void StopFileDownload(int dlUID)` Stops a file download that was started by calling the `DownloadFile` method
- `public void StartHTTPAPP()` Call to start receiving HTTP request and response headers via `ProtocolHandlerOnResponse` and `ProtocolHandlerOnBeginTransaction` events
- `public void StopHTTPAPP()` Call to stop `ProtocolHandlerOnResponse` and `ProtocolHandlerOnBeginTransaction` events for HTTP protocol
- `public void StopHTTPSAPP()` Call to stop `ProtocolHandlerOnResponse` and `ProtocolHandlerOnBeginTransaction` events for HTTPS protocol
- `public void StartHTTPSAPP()` Call to start receiving HTTPS request and response headers via `ProtocolHandlerOnResponse` and `ProtocolHandlerOnBeginTransaction` events
- `void SetAllowHTMLDialogs()` - default `true` - `set` allow or disallow flag for HTML dialogs launched using `showModelessDialog()` and `showModalDialog()` methods using `CBT` Window Hook
- `bool GetAllowHTMLDialogs()` - `get` allow or disallow flag for HTML dialogs launched using `showModelessDialog()` and `showModalDialog()` methods using `CBT` Window Hook
- `public bool AutomationTask_PerformClickButton(string btnname)` - Performs a click on a `Button` element with given name
- `public bool AutomationTask_PerformClickLink(string linkname)` - Performs a click on a `Link` element with given name
- `public bool AutomationTask_PerformEnterData(string inputname, string strValue)` - Enters `strValue` into an input element with given name
- `public bool AutomationTask_PerformEnterDataTextArea(string inputname, string strValue)` - Enters strValue into a textarea element with given name
- `public bool AutomationTask_PerformSubmitForm(string formname)` - Submits a form with given name
- `public bool AutomationTask_PerformSelectList(string selectname, string listitemvalue)` - Selects a list item element with given `listitemvalue`
- `AutomationTask_PerformSelectRadio(string radioname)` - Selects a radio or checkbox element with given `radioname`

Events

DWebBrowserEvents2

`DocumentComplete` has been modified to include an extra parameter, `IsTopLevel`, which indicates whether or not we have the top-level document.

- `DocumentComplete`
- `BeforeNavigate2`
- `ClientToHostWindow`
- `CommandStateChange`
- `DownloadBegin`
- `DownloadComplete`
- `FileDownload`
- `NavigateComplete2`
- `NavigateError`

- NewWindow2
- NewWindow3
- PrintTemplateInstantiation
- PrintTemplateTeardown
- PrivacyImpactedStateChange
- ProgressChange
- PropertyChange
- SetSecureLockIcon
- StatusTextChange
- TitleChange
- WindowClosing
- WindowSetHeight
- WindowSetLeft
- WindowSetResizable
- WindowSetTop
- WindowSetWidth
- UpdatePageStatus

Drag drop

To use internal drag drop functionality, RegisterForInternalDragDrop must be set to true (default)

- WBDragEnter
- WBDragLeave
- WBDragOver
- WBDragDrop

Extended

FileDownloadExXXX events are activated by setting UseInternalDownloadManager property to true (default)

- WBKeyDown - Rather than just firing when accelerator keys are used, I have decided to intercept key down and up, which I find to be more useful
- WBKeyUp
- WBContextMenu - Handle context menus
- WBGetOptionKeyPath - Allows the client to set up their own registry settings for Webbrowser control to use
- WBDocHostShowUIShowMessage - Allows interception of messageboxes
- DocumentCompleteEX - Setting SendSourceOnDocumentCompleteWBEx property will cause this event to fire instead of DocumentComplete, passing an extra parameter that contains the source of the document
- WBAuthenticate - Fires for basic and NTLM authentications. For NTLM, client needs to pass credentials as Username = Domain\username
- WBSecurityProblem - Fires when WinInet encounters a security problem
- WBEvaluteNewWindow - XPsp2, replaces NewWindowx events
- RefreshBegin
- RefreshEnd
- ScriptError - Fires for script errors. Contains all details in regard to script error
- ProcessUrlAction - Policy based URL processing
- HTMLEvent - Only event handler for HTMLDocument and HTMLwindow events
- FileDownloadExStart - If UseInternalDownloadManager true, notifies client of a request to start a file download. Stop download at any point using a unique ID, save file in the background. Overrides Webbrowser default file download mechanism
- FileDownloadExEnd - If UseInternalDownloadManager true, notifies client of end of a file download
- FileDownloadExProgress - If UseInternalDownloadManager true, notifies client of status of a file download
- FileDownloadExAuthenticate - If UseInternalDownloadManager true, notifies client of a request from server for authentication
- FileDownloadExError - If UseInternalDownloadManager true, notifies client of an error during a file download

- `ProtocolHandlerOnBeginTransaction` Enables the client to view all the HTTP and HTTPS request headers of Webbrowser control
- `ProtocolHandlerOnResponse` Enables the client to view all HTTP and HTTPS the response headers of Webbrowser control
- `AllowFocusChange` - IE7 Vista - notify client when focus is being changed via implementation of `IProtectFocus` interface
- `HTMLOMWindowServices_moveTo` - `IHTMLOMWindowServices` implementation - Moves the screen position of the upper-left corner of the application window to the specified coordinates
- `HTMLOMWindowServices_moveBy` - `IHTMLOMWindowServices` implementation - Moves the screen position of the application window by the specified offset values
- `HTMLOMWindowServices_resizeTo` - `IHTMLOMWindowServices` implementation - Changes the current size of the application window by the specified offset values.
- `HTMLOMWindowServices_resizeBy` - `IHTMLOMWindowServices` implementation - Sets the size of the application window to the specified values

Demo application

I have attempted to make the demo as comprehensive as possible. Here are some of its main features:

- Multi Tab, Thumb navigation and synchronization of GUI elements
- Document information
- DOM information
- A multi tab HTML editor with source and preview tabs. Offering many functionalities, background and foreground colors, font, font size, alignment, bold, italic, underline, indent, out dent, save, load, print, text back and fore colors. Insert bulleted list, order list, link, image, br, hr, table, table cell properties, source highlighting, a tree populated with most common tags for insertion in source view, full drag drop
- Full favorites implementation using dynamic menus
- Cookies and Cache viewer and remover, per site or all
- Demonstration of just about all `DWebBrowserEvents2`, including popups.
- Usage of a static class to share forms and controls within the application
- Find and highlight in page
- Authentication
- Clear cache
- Loading images from an embedded image strip
- Search engine query

Final note

Although reasonable care has been taken to ensure the correctness of this implementation, expect bugs. This code should never be used in any application without proper verification and testing. When reporting bugs, sending suggestions or requests, or asking a question, please include information regarding IE and OS versions and service packs. Include as much detail as possible. This will be very helpful, not only for me to find and fix bugs or implement requested features, but for others who may be in a similar position.

License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so.

History

05 June 2007 - version (1.0.0.3)

- Added `AutomationTask_PerformEnterDataTextArea` method - to automate text entry into textarea element
- Added `AutomationTask_PerformSubmitForm` method - to automate form submit
- Added an overload for `InvokeScript` which accepts an `IWebBrowser2` interface
- Fixed return value of `IDocHostShowUI.ShowMessage` method

25 May 2007 - version (1.0.0.2)

I would like to thank everyone for sending their suggestions, bug reports, and feature requests.

- **Properties**
  - `ObjectForScripting` property. Allows JavaScript functions in an HTML page to call methods and properties of a an instance class passed to this property. Same as `ObjectForScripting` of C# Webbrowser wrapper control. An example of how to use this property has been provided in `WinExternal` class of `frmMain`
  - `UseInternalDownloadManager` - Set to `true`, default, to allow the control to take over file downloads. `FileDownloadExxxx` events are fired instead of `FileDownload` event
  - `FileDownloadDirectory` Default file download directory. Set to users *MyDocuments* folder by default. Used only if `UseInternalDownloadManager` property is set to `true`
  - `DocumentTitle`
  - `DocumentSource`
  - `Border3DEnabled` - Webbrowser 3D border
  - `ScrollBarsEnabled` - Webbrowser Scrollbars
  - `DownloadImages` - shortcut for `DOCDOWNLOADCTLFLAG.DLIMAGES`
  - `DownloadSounds` - shortcut for `DOCDOWNLOADCTLFLAG.BGSOUNDS`
  - `DownloadActiveX` - shortcut for `DOCDOWNLOADCTLFLAG.NO_DLACTIVEXCTLS`
  - `DownloadJava` - shortcut for `DOCDOWNLOADCTLFLAG.NO_JAVA`
  - `DownloadFrames` - shortcut for `DOCDOWNLOADCTLFLAG.NO_FRAMEDOWNLOAD`
  - `DownloadScripts` - shortcut for `DOCDOWNLOADCTLFLAG.NO_SCRIPTS`

- **Events**
  - `FileDownloadExStart` - If `UseInternalDownloadManager` `true`, notifies client of a request to start a file download. Stop download at any point using a unique id, save file in the background, ... Overrides webbrowser default file download mechanism
  - `FileDownloadExEnd` - If `UseInternalDownloadManager` `true`, notifies client of end of a file download
  - `FileDownloadExProgress` - If `UseInternalDownloadManager` `true`, notifies client of status of a file download
  - `FileDownloadExAuthenticate` - If `UseInternalDownloadManager` `true`, notifies client of a request from server for authentication
  - `FileDownloadExError` - If `UseInternalDownloadManager` `true`, notifies client of an error during a file download
  - `ProtocolHandlerOnBeginTransaction` - Enables the client to view all the HTTP and HTTPS request headers of webbrowser control
  - `ProtocolHandlerOnResponse` Enables the client to view all HTTP and HTTPS the response headers of webbrowser control
  - `AllowFocusChange` To notify client when focus is being changed
  - `HTMLOMWindowServices_moveTo` - `IHTMLOMWindowServices` implementation - Moves the screen position of the upper-left corner of the application window to the specified coordinates
  - `HTMLOMWindowServices_moveBy` - `IHTMLOMWindowServices` implementation - Moves the screen position of the application window by the specified offset values
  - `HTMLOMWindowServices_resizeTo` - `IHTMLOMWindowServices` implementation - Changes the current size of the application window by the specified offset values.
  - `HTMLOMWindowServices_resizeBy` - `IHTMLOMWindowServices` implementation - `Set`s the size of the application window to the specified values

- **Interfaces**
  - `IProtectFocus` interface. IE7 and Vista only. Queries for permission to grab the focus when loading the page or when a script attempts to focus an element
  - `IHTMLDocument2.write` method definition, `psarray` param changed from `object[]` to `object`
  - `IHTMLDocument2.writeln` method definition, `psarray` param changed from `object[]` to `object`
  - `IProtectFocus` - IE7 + Vista - Queries for permission to grab the focus when loading the page or when a script attempts to focus an element
  - `IHTMLOMWindowServices` - XP sp2 - Enables applications that are hosting the WebBrowser Control to intercept the window object calls for manipulating the host window coordinates from Dynamic HTML (DHTML) script
  - `IHTMLElementEvents2`

  - IHTMLDataTransfer and cHTMLElementEvenets2 class to handle HTMLElementEvents2 of an HTMLElement. Used in frmHTMLEditor to demonstrate handling of internal drag drop of elements
  - IDownloadManager via a COM assembly. If activated by setting UseInternalDownloadManager property to true, default, a client application can be notified via FileDownloadExXXX events of a file download, replacing the default MSHTML download mechanism

- **Misc**
  - Return values to methods such as GoBack, GoForward,... to indicate failure or success
  - An assembly file to mark the control as COM visible
  - Overload OleCommandExec method. OleCommandExec(bool bTopLevel, MSHTML_COMMAND_IDS CmdID, object pvaIn) which accepts a parameter
  - Edit, Source, and Preview functionalities to frmHTMLEditor via a tab control
  - A CustomComboBox and ToolStripCustomComboBox to handle display of colors and fonts
  - A simple RichTextBox control with HTML syntax highlighting for viewing the source of the HTML document used in frmHTMLEditor
  - frmTableCellProp, to set/get a table cell properties in frmHTMLEditor
  - A tree containing most common HTML tags to be used in conjunction with the RichTextBox source editor in frmHTMLEditor
  - A custom SplitContainer based control, used in frmHTMLEditor
  - frmFileDownload, to manage file downloads visual status (simple)
  - Removed frmThumbs. Moved thumb handling to main form via a toolstrip

- **Methods**
  - SetAllowHTMLDialogs - default true - set allow or disallow flag for HTML dialogs launched using showModelessDialog() and showModalDialog() methods using CBT Window Hook
  - GetAllowHTMLDialogs - get allow or disallow flag for HTML dialogs launched using showModelessDialog() and showModalDialog() methods using CBT Window Hook
  - DownloadFile - Attempts to download a file asynch. FileDownloadExXXX events are used for notifications. Return value is a unique id for this download that can be used to stop the download
  - StopFileDownload - Stops a file download that was started by calling DownloadFile method
  - StartHTTPAPP - Call to start receiving HTTP request and response headers via ProtocolHandlerOnResponse and ProtocolHandlerOnBeginTransaction events
  - StopHTTPAPP - Call to stop ProtocolHandlerOnResponse and ProtocolHandlerOnBeginTransaction events for HTTP protocol
  - AutomationTask_PerformClickButton - Performs a click on a Button element with given name
  - AutomationTask_PerformClickLink - Performs a click on a Link element with given name
  - AutomationTask_PerformEnterData - Enters a given string into an input element with given name
  - AutomationTask_PerformSelectList - Selects a list item element with given list item value
  - AutomationTask_PerformSelectRadio - Selects a radio or checkbox element with given radio or checkbox name
  - GetElementsByName - Wrapper for IHTMLDocument3::getElementsByName

19 Feb 2007 - version (1.0.0.1)

- **BUG FIXES**
  - **Methods**
    - IDocHostShowUI.ShowMessage - no messages are displayed by default
    - DWebBrowserEvents2.WindowClosing - was not firing

- **ADDITIONS**
  - **Methods**
    - DISPATCH_WindowClosing - Replaced DWebBrowserEvents2.WindowClosing
    - NavToBlank
    - LoadHtmlIntoBrowser (Overloaded)

- **BUG FIXES**
  - **Methods**
    - `GetSource(IWebBrowser2 thisBrowser)`, added support for UTF8 and UNICODE pages
    - `Navigate(string URL, BrowserNavConstants Flags)`, using `BrowserNavConstants.navOpenInNewWindow` was loading URL into the current control. Now it opens a new IE window
    - `GetSource(bool bTopLevel)`, changed return value from `innerHTML` to `outerHTML`
  - **Events**
    - `NavigationError` event
  - **Interfaces**
    - `IHTMLDocument2.write` method definition, `psarray` param changed from `object[]` to `object`
    - `IHTMLDocument2.writeln` method definition, `psarray` param changed from `object[]` to `object`

19 Feb 2007 - version (1.0.0.1)

- **BUG FIXES**
  - **Methods**
    - `IDocHostShowUI.ShowMessage` - no messages are displayed by default.
    - `DWebBrowserEvents2.WindowClosing` - was not firing.
- **ADDITIONS**
  - **Methods**
    - `DISPATCH_WindowClosing` - Replaced `DWebBrowserEvents2.WindowClosing`
    - `NavToBlank`
    - `LoadHtmlIntoBrowser` (Overloaded)

06 Feb 2007

- Fixed Drag Drop - Now the client only receives `WBDragDrop` event which contains `DataObject` instance.
- Added new HTML editing functions to the `HTMLEditor` form. Add/delete table row and col, ...

17 Jan 2007

- Initial version (1.0.0.0) posted.

License

This article, along with any associated source code and files, is licensed under The Code Project Open License (CPOL)

About the Author

mehrcpp

Location: Canada

Member

Discussions and Feedback

489 messages have been posted for this article. Visit **http://www.codeproject.com/KB/miscctrl/csEXWB.aspx** to post and view comments on this article, or click **here** to get a print view with messages.

PermaLink | Privacy | Terms of Use
Last Updated: 28 May 2007
Editor: Genevieve Sovereign

Copyright 2007 by mehrcpp
Everything else Copyright © CodeProject, 1999-2010
Web22 | Advertise on the Code Project