Cor Ligthert is Microsoft MVP Visual Developer

Het gebruiksgemak van de nieuwe WebBrowser Control

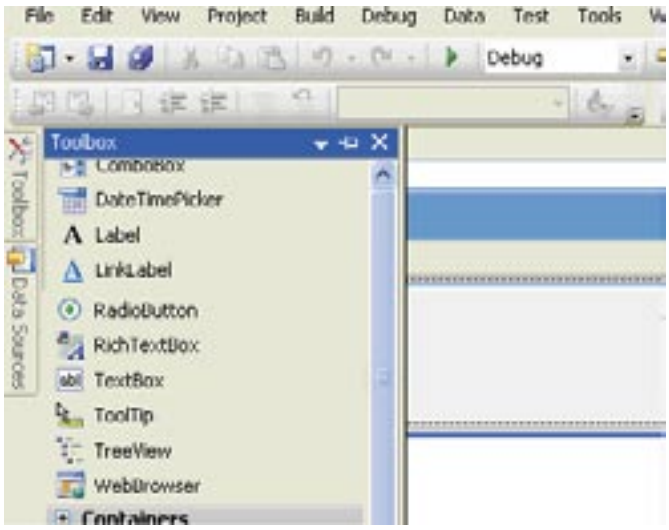
DE WEBBROWSER CONTROL IN DE PRAKTIJK

We zien vaak prachtige browsers opduiken waarmee de mooiste dingen mogelijk zijn. Bijvoorbeeld browsers met tabbladen. Dat een dergelijk tabbed browser met Visual Studio in 10 minuten te maken is weten echter weinigen.

Wat is de WebBrowser Control? Internet Explorer is al jaren ingebed in het Windows besturingssysteem, en kan daarom dus al van de versie IE 4.0 worden gebruikt voor het ontwikkelen van programmatuur. Hiervoor is in de loop der tijd - als interface - van AxWebbrowser, MSHTML en DHTML gebruikgemaakt. De ingebedde Internet Explorer zelf heet SHDOCVW. Nu is er ook een wrapper (interface) standaard in het .NET Framework 2.0 opgenomen onder de naam WebBrowser. Zoals we hier zullen aantonen, is deze webbrowser eenvoudig te gebruiken. De webbrowser in 2.0 heeft een flink aantal aspecten, dat het gebruik ten opzichte van de vorige versies vereenvoudigt. De documentatie op MSDN is op dit moment veelal nog gebaseerd op de AXWebbrowser-class, de MSHTML-class of de DHTML-class en daarom moeilijk te vinden maar begint er nu beter uit te zien.

Wat is het vernieuwende aan de .NET 2.0 webbrowser?

Het belangrijkste is wel dat de webbrowser geen interop meer is naar de Interop.AxWebbrowser. De lastige RESX interop-koppeling, die eigenlijk met de hand niet te zetten was, is niet meer nodig. Ook bij de deployment is deze DLL niet meer nodig, omdat de webbrowser nu een onderdeel is van het Framework. De set-up package wordt dus net zo klein als bij alle .NET-programma's. Het werken met de webbrowser is ook veel prettiger geworden. Geen uit het project verdwenen browser wat vaak kwam door een verkeerde wijziging in de code; iets dat voor sommige ontwikkelaars maar op één manier te herstellen was, namelijk door opnieuw beginnen.



Afbeelding 1. De WebBrowser Control in de toolbox van Visual Studio

Er zijn volgens mij twee belangrijke toevoegingen in de webbrowser:

- DocumentText, het document met de HTML-codes in tekstformaat
- Document.DomDocument, het document in HTML Document Object Model (DOM)-opmaak.

Deze twee toevoegingen vereenvoudigen het ontwikkelen met de webbrowser vergeleken met de vorige klassen, waar eindeloos casten nodig was/is (hoewel MSHTML heel vaak nodig blijft, maar dit zit al langer in .NET). Iets zeggen over de eenvoud van een tabbed website is natuurlijk één, iets bewijzen is twee. Daarom laat ik hieronder zien hoe je een tabbed webbrowser moet maken.

Het maken van een tabbed browser

We gebruiken hiervoor Visual Studio 2005 en openen een nieuw project met Visual Basic of C#. We doen de volgende stappen. Sleep een TabControl op het form en zet de eigenschap 'Dock' hiervan op 'Fill'. Sleep hierop een textbox en zet de eigenschap 'Dock' hiervan op Top; zie afbeelding 1.

Sleep nu ook een webbrowser op het tabblad en verander de eigenschap Dock van Fill in Bottom. Maak de webbrowser passend onder de Tekstbox en kopieer de webbrowser en de textbox, ga dan naar tab2 in de TabControl en plak vervolgens de gekopieerde webbrowser en de textbox op dit tabblad 2. Ga naar het codegedeelte en typ de code van codevoorbeeld 1 hierin. Gebruik bij het maken van de events in C# de solution explorer en in Visual Basic de comboboxen. Uiteindelijk zijn het slechts ongeveer 70 tekens die getypt moeten worden.
Het codevoorbeeld 1 is geschreven in Visual Basic. De code in C# is praktisch dezelfde buiten de (e.KeyCode == Keys.Return) een puntkomma aan het einde van de twee regels en het niet plaatsen van de 'End If'.
Druk op debug en browse bijvoorbeeld in de eerste tekstbox naar www.microsoft.com en in de volgende tab naar www.google.com. Voilà, je hebt nu een tabbed browser.

Meer mogelijkheden met de webbrowser

We kunnen met de webbrowser nog veel meer doen; bijvoorbeeld een webcrawler maken (een programma om eenvoudig wat dan ook van het web te verzamelen). We zien dat dit duizendvoudig gebeurt. Het is namelijk relatief eenvoudig; afhankelijk van wat men wil bereiken.

Een WYSIWYG HTML-editor met de webbrowser

Een derde mogelijkheid is om van de webbrowser een leuke, in goed Nederlands, WYSIWYG HTML-editor te maken. Dit echter met de beperking zoals de makers van de webbrowser dit hebben

```
Public Class Form1
  Private Sub TextBox1_KeyDown(ByVal sender As System.Object, _
  ByVal e As System.Windows.Forms.KeyEventArgs) Handles TextBox1.KeyDown
    If e.KeyCode = Keys.Return Then
      WebBrowser1.Navigate(TextBox1.Text)
    End If
  End Sub

  Private Sub TextBox2_KeyDown(ByVal sender As System.Object, _
  ByVal e As System.Windows.Forms.KeyEventArgs) Handles TextBox2.KeyDown
    If e.KeyCode = Keys.Return Then
      WebBrowser2.Navigate(TextBox2.Text)
    End If
  End Sub
End Class
```

Codevoorbeeld 1.

bedacht. Net als veel andere WYSIWYG HTML-editors zit er geen voorziening in om eenvoudig met tabellen om te gaan. Het nadeel is dus dat plaatjes moeilijk zijn te positioneren. Net als met de tabbed webbrowser gaan we er hier een maken. We gaan echter even verder dan met de tabbed webbrowser en proberen het ook nog een beetje mooi te maken. Dat wat onder andere nodig is om een tabbed webbrowser mooier te maken, komt dan meteen tevoorschijn.

We gaan een editor maken waarmee we:

- Blanco kunnen beginnen, maar die we ook op 'Nieuw' kunnen zetten
- Tekst kunnen lezen vanuit een bestand
- Tekst als HTML kunnen opslaan op schijf
- Kunnen afdrukken als HTML
- Vanuit de tekst kunnen knippen, kopiëren, plakken en verwijderen met buttons
- De fonts kunnen wijzigen met een button
- De tekst kunnen weergeven als HTML in een tekstbox.

De benodigde code is via kopiëren en plakken op te halen van deze website, terwijl het project ook als zipfile hier is op te halen: http://www.vb-tips.com/default.aspx?ID=58514c5a-5f5d-4b6d-a7bd-da738cdc7c2c

Het begin van het maken van de webbrowser HTML-editor

Start een nieuw windowsform-project. Sleep een splitcontainer van de toolbox op het form; het komt meteen in het formaat dock fill. Zet de eigenschap 'orientation' hiervan op 'horizontaal'. Sleep nu een toolstrip vanuit de toolbox op het form; het wordt direct bovenin geplaatst. Sleep vervolgens een webbrowser vanuit

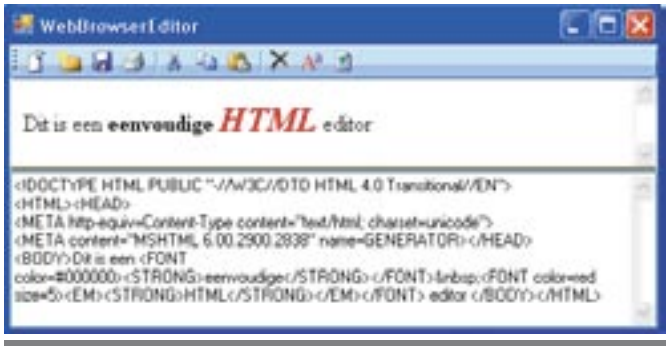
Afbeelding 2. Tabbed browser

```
Public Class Form1
' Om niet teveel te hoeven casten deze extra reference
Private wb As mshtml.IHTMLDocument2
Private Sub Form1_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles MyBase.Load
    WebBrowser1.DocumentText = ""
    ' De zetting van de extra reference
    wb = DirectCast(WebBrowser1.Document.DomDocument,
    mshtml.IHTMLDocument2)
    wb.designMode = "On"
End Sub
```

Codevoorbeeld 2.

de toolbox in panel1; het staat direct in formaat dock fill. Sleep als laatste een textbox vanuit de toolbox in panel2. Zet de dock-eigenschappen van de texbox op fill en de multiline op true. In het project creëren we dan nog een folder 'images'. We selecteren deze map en gaan de images importeren. De benodigde images voor de buttons zijn te vinden in het opgeslagen zipfile in Visual Studio 2005. We vinden hem op: C:\Program Files\Microsoft Visual Studio8\Common7\VS2005ImageLibrary. We selecteren dus de map 'images' en gaan de benodigde images importeren via Add Existing Item. Houd er rekening mee dat dit een naar werkje is. Waarschijnlijk heeft de Microsoft-ontwikkelaar willen laten zien hoe het niet moet. De map die we moeten gebruiken is. C:\Program Files\Microsoft Visual Studio8\Common7\VS2005ImageLibrary\bitmaps\commands\24color. Je hebt dus ongeveer 10 stappen nodig om er te komen. We selecteren achtereenvolgens Copy.bmp, Cut.bmp, Delete.bmp, Font.bmp, HTMLpage.bmp, newfolder.bmp, open.bmp, paste.bmp en print.bmp. Dit kan helaas niet in één keer, Hiermee zijn de voorbereidingen beëindigd. We beginnen met wat code te schrijven zoals in codevoorbeeld 2.

We hebben MSHTML voor een interface nodig. MSHTML is de afschuwelijkste class die waarschijnlijk ooit is gemaakt. Het zit vol met overlappende interfaces en geeft in Visual Basic .NET 2002/2003 - als een import is gezet (voor C#-lezers 'using') - een bijna complete blokkering van de IDE. In de versie 2005 is dit gelukkig niet meer het geval. De MSHTML-class is, buiten dat hij verschrikkelijk is, ook heel mooi. Via diverse interfaces representeert deze class volledig het HTML Document Object Model (DOM) en dus het mooie hulpmiddel bij webcrawling. Een HTML-document kan volledig worden ontleed; dus ook een gegenereerde ASPX, ASP of wat dan ook aan de clientzijde wordt gelezen. Om MSHTML te kunnen gebruiken, moet een reference worden gezet. Voor degenen die niet weten hoe dit gaat. Project -> Add Reference en dan vanuit de .NET-box: Microsoft.mshtml. Zoals we zien creëert onze procedure in codevoorbeeld 2 meteen een leeg document. Dit gebeurt via WebBrowser1.DocumentText = "". (Hier zien we meteen een voordeel van de property 'DocumentText'. Dit ging vroeger veel lastiger.) De code 'Wb.designMode = "On"' zet de webbrowser in editing-mode. In de documentatie op MSDN staat dat dit nog niet is geïmplementeerd. Dus bij problemen is er niet veel ondersteuning te halen. Dit 'On' staan heeft de



Afbeelding 3. De WebrowserEditor

onhebbelijke eigenschap dat als iets wordt overschreven, er altijd gevraagd wordt of er eerst veiliggesteld moet worden. Dit is te voorkomen door de design-mode eerst op 'Off' te zetten als dit niet moet worden gedaan. (Gaat dit dan nog niet dan is er een truc met een timer, een extra visible control en terugspringen, ik toon deze truc hier echter niet). We kunnen nu het programma proberen via Debug. Het ziet er nu nog steeds uit als een normale text editor. Het heeft echter een contextmenu, dus we kunnen al heel wat doen. (Dit contextmenu is te overschrijven. In dit voorbeeld is dat niet gedaan, omdat ik het contextmenu gebruiksvriendelijk vind.)

Toevoegen van de functies van de eerste button

We willen natuurlijk met een schone lei beginnen. We creëren hiervoor een button in onze toolstrip. We gaan naar de designer en klikken op de toolstrip. Een dropdownbox verschijnt, we openen dit en kiezen de button; zie afbeelding 4. We selecteren de button die vooraan is verschenen en openen de eigenschappen hiervan.

Klik nu op images en dan op import. We moeten nu naar de image-map browsen die we in ons project hebben gecreëerd; omdat het pad met Visual Studio nogal verschillend is, is hier geen leidraad gegeven. Als we de image-map hebben gevonden, kiezen we voor de bitmap NewFolder en importeren deze. Om ons programma leesbaar te houden, veranderen we in de eigenschappen de naam van deze button in NewButton. In de ToolTipText zetten we 'Nieuw'. We creëren met een dubbelclick op deze button het click-event dat we nodig hebben. Hierin plakken we de volgende coderegel. Dit is dus dezelfde code als in het load_form event.

```
WebBrowser1.DocumentText = ""
```

Toevoegen van een tweede button

We willen ook een bestaand document kunnen inlezen. Hiervoor doen we weer alle stappen en voegen nu de button NewFolder toe. We geven de button een naam en zetten de ToolTipText op Bestand. Door erop te klikken creëren we een event waarin we de code van codevoorbeeld 3 plakken.

Het gebruik van de execCommand-code

We willen de code echter ook wegschrijven. Dit kan door een SaveFileDialog te gebruiken met de DocumentText, dit kon dus vroeger niet. We gaan nu voor het eerst de execCommand-codes gebruiken. Op MSDN zijn de meeste voorbeelden gebaseerd op C++ en kunnen volgens mij een C#- of Visual Basic-programmeur afschrikken door de wijze zoals het is genoteerd. We creëren een button zoals eerder beschreven en gebruiken dan de onderstaande code.

```
wb.execCommand("SaveAS", False, Nothing)
```

Deze code zal altijd een dialogbox openen. Ook afdrukken gaat op dezelfde wijze, dus voor het programma de button toevoegen met alle eigenschappen en vervolgens codevoorbeeld 4 op de button plaatsen.

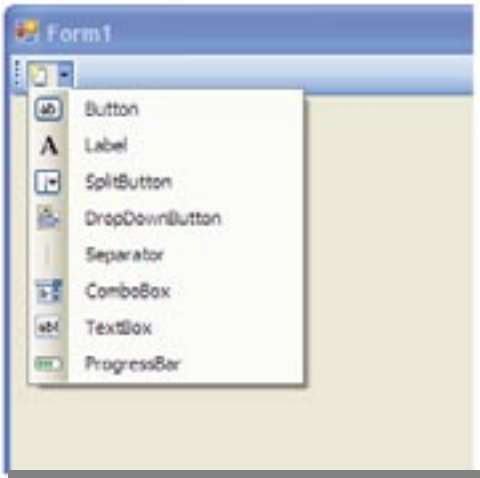
Hier zijn dus twee mogelijkheden, via het execCommand, waarbij netjes de dialogbox wordt getoond, zodat de instellingen nog veranderd kunnen worden, en zonder dat deze box meteen de instellingen van Internet Explorer overneemt.

Het omzetten naar HTML-text

Willen we zien wat we hebben gedaan, dan kunnen we de HTMLpageButton toevoegen.
We doen weer hetzelfde met deze button en de code zoals hieronder is te zien.

```
TextBox1.Text = WebBrowser1.DocumentText
```

De meeste zullen zeggen: logisch. Voor degenen die de oude



Afbeelding 4. De dropdownbox

AxWebbrowser nog kennen, is dit niet zo logisch. Let op, er is in het voorbeeld geen automatische terugfunctie, zodat editing in dit deel - zoals het nu is - geen resultaat heeft in het webbrowsergedeelte.

De edit-buttons

In dit voorbeeld gebruiken we vier buttons om te editen (Knip, Kopieer, Plak, Verwijder). Het kan met het contextmenu, maar met enkele buttons staat het mooier. Het gaat op dezelfde wijze als alle andere buttons. De code per button is zoals in codevoorbeeld 5.

Het queryCommandValue-commando

Dit is allemaal heel eenvoudig. Wat ik niet meer heb kunnen vinden, is de Font-editing van de DHTML-class. Hiervoor heb ik dus wat vingeroefeningen gedaan met de FontDialog. Ook is in de voorbeeldcode het queryCommandValue opgenomen. Hiermee is informatie van het webbrowser-document terug te halen. Een klein probleem in dit voorbeeld is wel dat de Developer waarschijnlijk niet zoals wij van links naar rechts leest, maar van rechts naar links. Waarschijnlijk komt dit omdat niet RGB, maar BGR is gebruikt. In dit geval geef ik alleen de code, waartussen wat begeleidende tekst staat over het waarom. Ik heb op deze manier het font behandeld, maar waarschijnlijk zijn er meer oplossingen. Wat direct duidelijk wordt is dat de webbrowser-control geen onbeperkt aantal mogelijkheden heeft. De fonts kunnen uitsluitend gezet worden als 'extra small', 'small', enzovoort. Deze worden weergegeven met de enums 1, 2, 3. 4, 5, 6, 7. Ik heb hiervoor een omrekening gemaakt vanuit pixels. Het mag duidelijk zijn dat het terugbrengen van meer pixelmogelijkheden naar zeven HTML-mogelijkheden nogal arbitrair is gebeurd.

```
Dim fo As New OpenFileDialog
    If fo.ShowDialog = Windows.Forms.DialogResult.OK Then
      Dim sr As New IO.StreamReader(fo.FileName)
      WebBrowser1.DocumentText = sr.ReadToEnd
End If
```

Codevoorbeeld 3.

```
wb.execCommand("Print", False, Nothing)
  'Bovenstaand is met print dialogbox
  'WebBrowser1.Print()
```

Codevoorbeeld 4.

```
wb.execCommand("Cut", False, Nothing)
wb.execCommand("Copy", False, Nothing)
wb.execCommand("Paste", False, Nothing)
wb.execCommand("Delete", False, Nothing)
```

Codevoorbeeld 5.

```
Dim fb As New FontDialog

'De fontsize in de WB is alleen de 1 tot en met de 7.
Dim fontSizes32() As Integer = {0, 8, 10, 12, 14, 16, 20, 24}

'Onderstaand de methode om iets op te halen uit de geselecteerde tekst
Dim fontSize As Integer = CInt(wb.queryCommandValue("FontSize"))
Dim returnColor As Integer = CInt(wb.queryCommandValue("ForeColor"))

'Hieronder is om de BGR bug te overkomen (is door mij geraporteerd)
Dim Intermidiate() As Byte = BitConverter.GetBytes(returnColor)
Dim IntermidiateByte As Byte = Intermidiate(0)
Intermidiate(0) = Intermidiate(2)
Intermidiate(2) = IntermidiateByte
returnColor = BitConverter.ToInt32(Intermidiate, 0)
'Einde workaround Bug

fb.Color = Color.FromArgb(returnColor)
Dim Bold As Boolean = CBool(wb.queryCommandValue("Bold"))
Dim Italic As Boolean = CBool(wb.queryCommandValue("Italic"))
Dim BoldStyle As New FontStyle
Dim ItalicStyle As New FontStyle
If Bold Then BoldStyle = FontStyle.Bold
If Italic Then ItalicStyle = FontStyle.Italic
   fb.Font = New Font(wb.queryCommandValue("FontName").ToString, _
   fontSizes32(fontSize), FontStyle.Regular Or BoldStyle _
   Or ItalicStyle)
fb.ShowColor = True
fb.ShowDialog()
'Het schoonmaken van het font formaat
wb.execCommand("RemoveFormat", False, Nothing)

'Het plaatsen van opnieuw de font settings.
Dim fz As String
Select Case fb.Font.Size
   Case Is < 9
      fz = "1"
   Case Is < 11
      fz = "2"
   Case Is < 13
      fz = "3"
   Case Is < 15
      fz = "4"
   Case Is < 17
      fz = "5"
   Case Is < 21
      fz = "6"
   Case Else
      fz = "7"
End Select
wb.execCommand("FontSize", False, fz)
wb.execCommand("ForeColor", False, ColorTranslator.ToHtml(fb.Color))
'Hieronder weer het zetten via het execCommand
If fb.Font.Italic = True Then
   wb.execCommand("Italic", False, Nothing)
End If
If fb.Font.Bold = True Then
   wb.execCommand("Bold", False, Nothing)
End If
   wb.execCommand("FontName", False, fb.Font.Name)
```

Codevoorbeeld 6.

Meer frames in een document

Waar velen zich nogal eens in vergissen is dat een webpagina niet altijd een document is, maar vaak uit verschillende documenten bestaat; frames. De webbrowser kan meer documenten binnenhalen en die vervolgens bewerken. De documenten dienen via de verschillende events in een collectie te worden gezet. Vervelend is dat er nog wel eens pop-ups of headerloze documenten worden gemaakt. Gelukkig zal dit afnemen door de veiligheidswijzigingen in IE 6.0.

De koppeling met IE

De webbrowser gebruikt SHDOCVW. Dit betekent dat alle instellingen voor Internet Explorer ook van kracht zijn voor de webbrowser. Het is natuurlijk mogelijk om deze settings via het register aan te passen. In mijn ogen is dit laatste een slecht en zelfs gevaarlijk gebruik. De setting geldt dan ook voor alle Internet Explorer-acties die tijdens het gebruik van de webbrowswer worden gestart (hopend dat de programmeur ze terugzet) en kunnen de oorzaak zijn van veiligheidsrisico's. Dit lijkt mij uit den boze, vooral bij commerciele software die met de webbrowser wordt gemaakt.

Conclusie

De webbrowser is een leuk speeltje, en kan een heel goede invulling geven op die plaatsen waar geen moeilijke webpagina's gemaakt moeten worden. Het invoegen van images is moeilijk; mogelijk maar zonder tabellen geen gezicht. Het belangrijkste is om een webpagina zonder poespas binnen organisaties in beeld te brengen. Ook kan hierbij een volledige scanning op de te bezoeken websites plaatsvinden. Uitsluitend dat wat vanuit het management functioneel wordt bevonden, kan zodoende worden gebruikt. Als de selectie in een centrale database is opgeslagen, dan wordt het nog eenvoudiger te beheren, waarbij we zelfs aan persoons- of afdelingsgebonden afscheidingen kunnen denken. Natuurlijk kennen we allemaal de toepassing van een webbrowser in Visual Studio, dit soort toepassingen zijn natuurlijk niet alleen aan Microsoft voorbehouden. De webbrowser maakt het eenvoudig. Velen proberen de webbrowser te gebruiken om een soort webservice te creëren. Een simpel advies: niet doen. Een webservice is eenvoudiger te creëren via de .NET-tools. Ook het automatiseren van requests en responses vanuit bestaande, niet te beïnvloeden webpagina's is veel beter te doen met HTTPWebRequest en de HTTPWebResponse. Een voorbeeld van een alternatieve truc om zaken op te halen: http://www.vb-tips.com/default.aspx?ID=541adf13-d9c0-435c-893f-56dbb63fdf1c

De hoofdpagina op MSDN over de WebBrowser Control met veel maar nog beknopte informatie. De mogelijkheden zijn eigenlijk eindeloos. De beschrijving van de webbrowser-class was op het moment van schrijven van dit artikel moeilijk te vinden, omdat alle links nog naar de oude klassen verwezen.
Zorg ervoor dat je de eindeloze informatie op internet over de AxWebBrowser en de WebBrowser weet te scheiden. Bijna alles wat met de AxWebBrowser mogelijk is, moet ook mogelijk zijn met de webbrowser. Vaak zal echter veel casten nodig zijn plus dat het gebruik van extra DLL's moet worden vermeden.

Cor Ligthert (MVP) is waarschijnlijk een van de in Nederland langst actieve personen in de ICT. Dit laatste in zowel de technische als de organisatorische aspecten. Cor is als het even kan wars van nonsens benaderingen. De tijd heeft hem geleerd dat wat vandaag alleen voor de sier wordt toegevoegd morgen belachelijk is. Samen met zijn ' fellow' MVP Ken Tucker onderhoudt Cor een website. www.vb-tips.com die is gericht op het gebruik van Visual Basic (ADO.NET, ASP.NET).

Referentie

msdn2.microsoft.com/en-us/library/2te2y1x6(VS.80).aspx
msdn2.microsoft.com/en-us/library/system.windows.forms.webbrowser(VS.80).aspx
msdn2.microsoft.com/en-us/library/system.windows.forms.webbrowser_members(VS.80).aspx