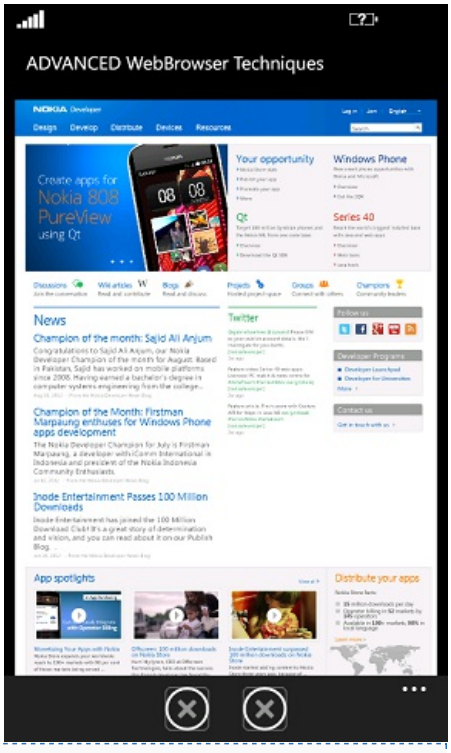
WebBrowser Control Techniques in Windows Phone

This wiki article shows how to use the Windows Phone WebBrowser control. It shows basic usage, and also advanced techniques for integrating the control deeply into C# code.

26 Aug 2012

Introduction



The `WebBrowser` control can be used to display web based content (HTML, CSS, JS) in your Windows Phone application. It can be used as the base of simple web apps, though to a fully fledged web browser. Qt developers will recognise the `WebBrowser` as the Qt "equivalent" of QWebView.

You may use it for instance to:

- Show loaded web content from the network (weather widget or facebook like for instance)
- Show static web content that is embedded to your application package
- Show dynamically generated content where some parts of the content are coming from your C# code and some are loaded from the network
- Create hybrid apps where some parts of your application UI are standard Silverlight stuff and some parts are web based. In some cases users does not even know which parts are web-based and which are not

Basic usage

If you just want to show a web page on your applications, it's really simple. Just add a `WebBrowser` component to your **.xaml** file and call it's Navigate method on the **.xaml.cs** code.

```
<phone:WebBrowser HorizontalAlignment="Stretch" Name="webBrowserControl"
VerticalAlignment="Stretch"/>
```

```
// Constructor
public MainPage()
    {
        webBrowserControl.Navigate(new Uri("http://www.developer.nokia.com/"));
    }
```

If you want to enable JavaScript (usually you want) in your web control you must set property IsScriptEnabled to true. By default, JavaSciprt is `disabled`.You may also want to enable HTML5 location services using IsGeoLocationEnabled property. Please note: to use geo location property, one has to have `Location` capability in the app as well.

```
webBrowserControl.IsScriptEnabled = true;
webBrowserControl.IsGeolocationEnabled = true;
```

Advanced usage

With `WebBrowser` you can build a more fully featured web browser, and even mix C# and JavaScript code. Let's start with navigation stack.

Navigation

If you wish to control how user can navigate in the web control, there are few events that you may want to listen:

1. Navigating. This event is fired when browser starts to navigate to new url. This event has NavigatingEventArgs as parameters and by setting `Cancel` property to true you can prevent user's navigation to certain urls if you wish.

2. Navigated. This event occurs after the web site has been found and device successfully loads the web content.
3. LoadCompleted This event is fired after all content for the site has been loaded (images, external scripts, external style sheet files and so on.).

One example where we could use the `Navigated` event is a case that we want to have a browser's back functionality in the device's back button. In the code we need to do following things

- Create a data structure for the visited URLs as member data. Stack could be useful in this case.

```
public partial class MainPage : PhoneApplicationPage
    {
        Stack<Uri> _navigationStack = new Stack<Uri>();
         ...
```

- Register to listen to `Navigated` event in the constructor.

```
webBrowserControl.Navigated += new
EventHandler<NavigationEventArgs>(WebControlNavigatesTo);
```

- Create the event hander for the `Navigated` event. When event occurs, store the URL to the stack

```
void WebControlNavigatesTo(Object sender, NavigationEventArgs navArgs)
        {
            _navigationStack.Push(navArgs.Uri);
            System.Diagnostics.Debug.WriteLine("WebControlNavigatesTo:" +
navArgs.Uri.AbsoluteUri);
        }
```

- We create a handler for the device's back button
- If the user presses the back button, we pop url from the stack and ask web control to navigate previous url.
- If we are at the first page and user presses the back button, we want to have default functionality (quit the app).

```
       protected override void OnBackKeyPress(CancelEventArgs e) {
            base.OnBackKeyPress(e);

            if (_navigationStack.Count() >= 2)
                {
                _navigationStack.Pop();
                webBrowserControl.Navigate(_navigationStack.Pop());
                e.Cancel = true;  // This prevents the default functionality of the
back button.
                }
        }
```

After these changes, the device's back button works same way as standard browser back button.

Calling C# code from JavaScript

It is also possible to call C# code from JavaScript code. With this technique you can create seamlessly integrated web views to your application. You can for instance:

- Request data or functionalities from the C# side (location, accelerometer data, play sounds etc.)
- Log from web code to `System.Diagnostics`
- Change to different views
- Pass parameters from JavaScript.

In web code, you must call `window.external.notify()` JavaScript function. You can also pass parameters to it:

```
window.external.notify("Uulalaa!");
```

To handle it on C# side you must listen to ScriptNotify event.

```
webBrowserControl.ScriptNotify += new EventHandler <NotifyEventArgs>(JavaScriptNotify);
```

And create the handler (in this case I'm printing the parameter to console and show a dialog).

```
       void JavaScriptNotify(Object sender, NotifyEventArgs notifyArgs)
        {
            System.Diagnostics.Debug.WriteLine("JavaScriptNotify: " + notifyArgs.Value);
            MessageBox.Show("You clicked the button in the \"web-ui\"");
        }
```

Calling (evaulate) JavaScript code from C#

It's also possible to call (or evaluate) JavaScript code from C#. There's a method called InvokeScript that can be used. If you don't need to pass any parameters, you can use it only with JavaScript function name:

```
webBrowserControl.InvokeScript("JSFunctionNameHere");
```

or if you want to pass parameters, parameters are stored to String array:

```
webBrowserControl.InvokeScript("JSFunctionNameHere", new string[] { "parameter1" });
```

You could also call JavaScript `eval()` function and give JavaScript code as parameter.

```
webBrowserControl.InvokeScript("eval", new string[] {
"javascriptcode.you.want.to.eval.goes.here :)" });
```

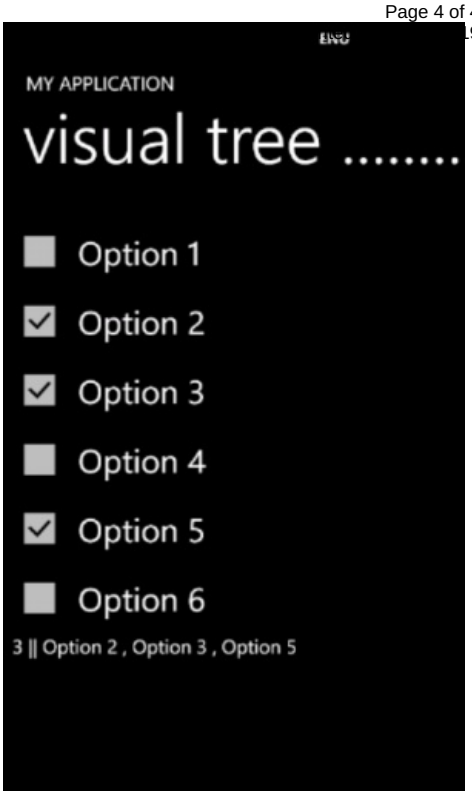
Sample application

I have created a sample application and a html code that demonstrates all the things (and few things more) we have discussed above. Sample application loads this html from the network.

Download the sample code from here: File:AdvancedWebBrowserControl.zip

Instructions:

- Start up the app
- If you press the button on the screen, the code on "C#-side" (In class MainPage) will be called. (JavaScript code calls C# code.) It shows a MessageBox dialog. If you look at the code, you will find out that passing parameters is also possible.
- You can also find two application bar buttons from the screen.
- If you press the button on the left, JavaScript function will be called from the C# side. (C# code calls JavaScript) The code adds some text dynamically to `WebBrowser`'s DOM tree.
- If you press the button on the right, application navigates to Nokia Developer portal and you may test the back button's functionality.



Summary

Thank you for reading this article. Remember to see the sample application and remember also to view the web (html) code. If you have any comments or questions, please use the commenting box below.