**Mobile Web Development**

**(Part 7)**

**Topic 1: IEMobile 6 features**

**Topic 2: Making use of a Browser Control inside a Smart Client Application**



[**http://msdn.microsoft.com/rampup**](http://msdn.microsoft.com/rampup)

**Introduction**

When you are about to start developing a new application that targets Windows Mobile devices, there is one fundamental question to be answered first. Must the application run locally on the device? The answer to this question determines if a pure web based application is suitable for you. If the device does not have a guaranteed network connection, developing a web based application does not make much sense. In order for the application to run, the device must be connected. On the other hand, if there is a reliable network connection available, creating a web based application does make sense. The advantage of web based applications is that they are not limited to run on Windows Mobile devices, but you can target a broader range of mobile devices. Having said that, the latest Windows Mobile Devices have great support for web based applications, thanks to the fact that they ship with a brand new browser, IEMobile, that is based on some of the latest Internet Explorer functionality. With the latest Windows Mobile devices, the end user’s browsing experience will be the same as using a desktop browser. The only difference is the smaller screen size. The latest Windows Mobile Devices can make use of ASP.NET 3.5 Web Sites and have Ajax support to allow limited roundtrips with less data transfer between the server and the device. This is great news, especially if your network connection is a relatively slow connection like a GPRS connection.

This article provides information on creating web based applications for Windows Mobile Devices, setting up the web application in such a way Ajax support is enabled and it also contains information on how to make use of browser controls inside a Smart Device application to get the best out of both worlds.

**Web Based Applications and Windows Mobile Support**

Even the earliest Windows Mobile Devices shipped with a Web Browser. However, until Windows Mobile 6 devices came to market, this browser had limited capabilities and was based upon an older version of Internet Explorer. The latest version of the browser on Windows Mobile Devices, Internet Explorer Mobile 6, is a full-featured browser that brings the same high-quality browsing experience to the user as desktop browsers. Internet Explorer Mobile 6 supports desktop-quality rendering and has the best compliance support of all versions of Internet Explorer on a Windows Mobile device to date. Several new features and improved support allows users to complete tasks quickly and easily.

* The support of Internet Explorer 8’s JScript version 5.7 in Internet Explorer Mobile 6 also enables developers to deliver desktop-consistent AJAX web experiences on Windows Mobile devices.
* Internet Explorer Mobile 6 can display both mobile customized and non-customized Web sites.
* Internet Explorer Mobile 6 can identify itself as a desktop browser or as a mobile browser depending on a user setting.
* Users have the possibility to use touch for panning and there are multiple levels of zooming available.

**Different options to view Web Sites**

When a user visits a Web Site or when the user runs a web based application, the browser on the client is identified by a UserAgent string. Depending on the contents of this UserAgent string, a web based application might provide more or less functionality (like Ajax support) to the client. The latest generation of Windows Mobile Devices have a version of Internet Explorer Mobile that can identify itself either as desktop browser or as a mobile browser. The user can easily select between these two options from inside the Internet Explorer menu on the device.



Figure - Internet Explorer Mobile 6 used in Desktop Mode and in Mobile Mode

In figure 1 you can see how the user can select between using the browser in desktop mode and in mobile mode. In the center of figure 1 you can see Internet Explorer Mobile running in mobile mode, making the viewng experience optimal for a Windows Mobile device. However, in order to get the best viewing experience, the website most be optimized for viewing on a Mobile device. On the right hand side in figure 1 you can see the same website, this time displayed in desktop mode. Even though the screen is smaller, the end user gets the same experience as visiting the same website from a desktop browser. On a device with a touch screen the user can navigate very fast over the web page by making use of panning over the screen.

**Detecting Browser Capabilities**

To determine what browser capabilities a client has, you can investigate the UserAgent string. The following UserAgent strings are returned by the latest Windows Mobile Devices, independent of the type of device (Professional or Standard):

* Mozilla/4.0 (compatible; MSIE 6.0; Windows CE; IEMobile 8.12; MSIEMobile 6.0)  
  This UserAgent string is returned when the browser is running in Mobile mode.
* Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)  
  This UserAgent string is returned when the browser is running in Desktop mode.

If the browser is running in Desktop mode, it automatically has Ajax capabilities enabled. However, if the browser is running in Mobile mode, the client has to tell the server that it can run **ie5** compatible functionality. Any browser on a Windows Mobile Device that has a IEMobile version of 6.12 or later has these capabilities. So in order to tell the server what the browser is capable of, you can make use of the following code snippet that investigates the UserAgent string.

public static bool IsIEMobileWithAjaxSupport(string input)

{

const string mobileBrowser = "IEMobile ";

bool ajaxSupported = false;

if (input.Contains(mobileBrowser))

{

string version = input.Substring(input.IndexOf(mobileBrowser) +

mobileBrowser.Length);

version = version.Remove(version.IndexOf(';'));

string[] versionNr = version.Split(new char[] { '.' });

int IEMobileMajor = Convert.ToInt32(versionNr[0]);

int IEMobileMinor = Convert.ToInt32(versionNr[1]);

ajaxSupported = IEMobileMajor > 6 || (IEMobileMajor == 6 && IEMobileMinor >= 12);

}

return ajaxSupported;

}

Listing - Determining Internet Explorer Mobile browser capabilities

If the Internet Explorer Mobile browser has a version number of 6.12 or larger, the server can be informed about the client browser capabilities with the following code snippet, as part of the FrameworkInitialize method inside the web application. Note that this is only done initially. Each time a PostBack message is received, for instance to refresh the page partially, it is not necessary to set the ClientTarget because the server already knows about its capabilities.

protected override void FrameworkInitialize()

{

base.FrameworkInitialize();

if (!IsPostBack)

{

if (IsIEMobileWithAjaxSupport(Request.UserAgent))

{

ClientTarget = "ie5";

}

}

}

Listing - Informing the server about the client capabilities

**Distinguishing between Windows Mobile Professional and Windows Mobile Standard**

Since both Windows Mobile Standard and Windows Mobile Professional devices will return the same UserAgent string, you need another mechanism to find out what kind of device is requesting a page. As part of the browser request, additional header information is also passed to the server. One of the headers contains the screen size of the device in pixels. That is the real important differentiator. Unlike Smart Client applications, browser based applications support similar User Interface controls both for Windows Mobile Standard and Windows Mobile Professional devices since they are running identical browsers. A web based application running on Windows Mobile Standard devices therefore can contain buttons. You simply can use the navigation keys to put focus on a button and can click it using the action button. The header field (“UA-Pixels”) returns the real number of pixels, meaning it will return correct values for high resolution and for normal resolution devices.

To setup the browser page for optimal viewing experiences, you can make use of the MobileOptimized meta-tag, indicating what screen size the web site is developed for. The following code inside the web application sets the MobileOptimized tag, depending on the screen width of the device:

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

if (IsIEMobileWithAjaxSupport(Request.UserAgent))

{

if (this.Header != null)

{

HtmlMeta mobileOptimized = new HtmlMeta();

mobileOptimized.Name = "MobileOptimized";

mobileOptimized.Content =

Request.Headers["UA-pixels"].Split(new char[] { 'X', 'x' })[0];

Header.Controls.Add(mobileOptimized);

}

}

}

}

Listing - Setting the 'right' width of a page to increase the end user's browsing experience

Using the Request.Header information, you can distinguish between different particular devices and optimize the viewing experience for users of all kinds of devices. In figure 2 you will see the same web page, being optimally rendered for either a Windows Mobile 6 Standard device and for a Windows Mobile 6 Professional device.



Figure - Rendering with different font sizes on different devices.

# The following code snippet shows you how you can modify the font size of Web Controls, depending on the device and especially the screen width of the device you are running the application on. Thanks to the rich functionality of ASP.NET 3.5 and thanks to the fact that you can easily modify control properties in the code behind file, you hardly have to deal with markup language.

protected void Page\_Load(object sender, EventArgs e)

{

string currentTime = DateTime.Now.ToLongTimeString();

if (!IsPostBack)

{

if (IsIEMobileWithAjaxSupport(Request.UserAgent))

{

if (this.Header != null)

{

HtmlMeta mobileOptimized = new HtmlMeta();

mobileOptimized.Name = "MobileOptimized";

mobileOptimized.Content = Request.Headers["UA-pixels"].Split(

new char[] { 'X', 'x' })[0];

Header.Controls.Add(mobileOptimized);

pageWidth = Convert.ToInt32(mobileOptimized.Content);

}

}

}

if (pageWidth < 240)

{

Label1.Font.Size = System.Web.UI.WebControls.FontUnit.Small;

Label2.Font.Size = System.Web.UI.WebControls.FontUnit.Small;

Label3.Font.Size = System.Web.UI.WebControls.FontUnit.Small;

Label4.Font.Size = System.Web.UI.WebControls.FontUnit.Small;

Label5.Font.Size = System.Web.UI.WebControls.FontUnit.Small;

DropDownList1.Font.Size = System.Web.UI.WebControls.FontUnit.Small;

Button1.Font.Size = System.Web.UI.WebControls.FontUnit.Small;

Menu1.Font.Size = System.Web.UI.WebControls.FontUnit.Small;

}

}

Listing - Adjusting control font sizes, depending on the client device.

Thanks to the fact that the new Internet Explorer Mobile supports the latest technology that is also available in the desktop version of Internet Explorer, it is possible to make use of most ASP.NET 3.5 functionality in web based applications.

**Combining Smart Client and Web functionality**

If you are developing a Smart Client application, you can make use of a browser control. The browser control in the latest generation of Windows Mobile devices makes use of the available web browser. That means that you can make use of ASP.NET 3.5 functionality, including Ajax support from inside a Smart Client application as well. Combining Web functionality with smart client functionality makes it possible for your users to continue working, even if they don’t have network connectivity. If network connectivity is available, the user can make use of the browser control, for instance to access data on a server database. Taking this approach makes it for instance relatively easy to solve conflicts between multiple users updating the same data in the database, because this can entirely be handled by the server. Another great use of this approach is running processing intensive tasks through a browser control, for instance to retrieve Virtual Earth mapping information. Instead of making use of Web Services, you can do all rendering server driven inside the browser control.

The browser control that you can add to a Smart Device application returns the same UserAgent string to the server as the stand-alone version of Internet Explorer Mobile does. However, it is not possible to set the browser control to mobile mode. Since the your Windows Mobile device has a limited screen size, you might want to create specific pages for use in combination with a browser control inside a Smart Client Application. In the following code sample, you see a complete application that detects network connectivity. If the device is connected, it is possible to browse to a web page with the information being displayed inside a web browser control. If the device is disconnected, the user can still enter new URL’s, but they can’t browse to them. Of course this is just a very simple example and a real application should offer the user more functionality.

public partial class Form1 : Form

{

private SystemState nrNetWorkConnections;

private SystemState deviceCradled;

public Form1()

{

InitializeComponent();

nrNetWorkConnections = new SystemState(SystemProperty.ConnectionsCount);

nrNetWorkConnections.Changed += new ChangeEventHandler(nrNetWorkConnections\_Changed);

deviceCradled = new SystemState(SystemProperty.CradlePresent);

deviceCradled.Changed += new ChangeEventHandler(deviceCradled\_Changed);

}

private void Form1\_Load(object sender, EventArgs e)

{

bool isCradled = (int)deviceCradled.CurrentValue != 0;

int nrNetworks = (int)nrNetWorkConnections.CurrentValue;

tbURL.Select(tbURL.Text.Length, 0);

btnGo.Enabled = isCradled || nrNetworks > 0;

}

void deviceCradled\_Changed(object sender, ChangeEventArgs args)

{

bool isCradled = (int)args.NewValue != 0;

int nrNetworks = (int)nrNetWorkConnections.CurrentValue;

btnGo.Enabled = isCradled || nrNetworks > 0;

}

void nrNetWorkConnections\_Changed(object sender, ChangeEventArgs args)

{

bool isCradled = (int)deviceCradled.CurrentValue != 0;

int nrNetworks = (int)args.NewValue;

btnGo.Enabled = isCradled || nrNetworks > 0;

}

private void btnGo\_Click(object sender, EventArgs e)

{

webBrowser1.Navigate(new Uri(tbURL.Text));

}

}

Listing - Complete Windows Mobile Application that detects network connections

Combining smart client functionality with web functionality brings the best of two worlds together. You can combine user productivity with processing power on the server. Also, if data is stored centrally, it is easy to download a local copy of that data, maybe even through a web service. Then you can use the data on the device as a local cache and upload it again to the server once the user is done working on it. This simplifies conflict resolution, and the amount of data available on the device can be limited to just that data that the user needs to work on. This not only preserves costly resources on the device, but it also guarantees that less company sensitive data is available on the device at any time.



Figure - Web Application inside a browser and as part of a Smart Client Application

# **Additional resources and references**

* [Window Mobile Developer Center](http://msdn.microsoft.com/en-us/windowsmobile/default.aspx)
* [IE Mobile Team Blog](http://www.microsoft.com/events/series/detail/webcastdetails.aspx?seriesid=86&webcastid=5596)
* [Windows Mobile 6.5 – What’s in it for developers](http://msdn.microsoft.com/en-us/library/bb158571.aspx)
* [Video: How do I Determine Whether a Windows Mobile Device Supports ASP.NET AJAX](http://msdn.microsoft.com/en-us/windowsmobile/bb872388.aspx)