



ASP+ Mobile Controls

Alex Homer

Introduction

If all the media news and hype is to be believed, we'll all be using a cellular phone or other mobile device to access the Web in the near future. OK, so this is probably a little over-optimistic, but there's no doubt that the number of visitors to your Web site that are not using a traditional PC-based browser will continue to increase.

To make coping with the various types of mobile devices easier, in particular cellular phones, ASP+ boasts a selection of server controls that provide output in WML rather than HTML. In this session, we look at the controls that Microsoft is currently developing. We'll also build a couple of our own controls to see just how easy it is. The session will also look at some of the issues involved in detecting the type of user-agent, and serving appropriate content – whether it's a mobile device, TV set-top box, or even a refrigerator.



Alex Homer is a software consultant and developer living and working in the UK. He is the CEO of Stonebroom Software, a company specializing in office integration and Internet related development that produces a range of vertical application software. Alex has worked with Wrox Press on several projects, including Professional ASP 2.0, IE5 XML Programmer's Reference, Professional ASP Techniques for Webmasters and Professional MTS MSMQ Programming with VB and ASP. Contact him on alex@stonebroom.com

What is ASP+?

Various ways exist for creating dynamic Web pages. In the past, the predominant technique was the use of executable programs that accessed the Web server via the Common Gateway Interface (CGI). However, over the past few years, new techniques that allow script code to be used within HTML pages have become more popular. Amongst these techniques are Perl, Java Server Pages (JSP), and Microsoft's Active Server Pages (ASP).

ASP started life as just a DLL that accessed the Web server through the Internet Server Application Programming Interface (ISAPI) – in other words an ISAPI DLL. This interface provides a fast and reliable connection to the Web server, as (unlike CGI executables) the code runs within the same process or memory space as the Web server. Using this technique, ASP has evolved to become the version 3.0 that is installed as part of Windows 2000.

What is .NET All About?

The fundamental design of ASP is now changing, along with the introduction of the new Microsoft **.NET Framework**. This framework is a complete departure from the language and operating system-specific structure of existing programming environments.

Today, most programming languages compile the source code directly to native code (code that runs directly on the processor), and hence are dependent upon the processor type. For example, code compiled for an Intel processor will not run on a Motorola processor.

The one major exception to this is Java, which uses a separate virtual machine to execute code that is compiled to a special processor and operating system-independent format. As long as a suitable Java virtual machine exists on the target system, the code can execute.

Microsoft has taken a broadly similar, but far more wide-ranging approach with the new .NET Framework. This provides a complete development and runtime environment within which code can execute. This code is compiled into a special **Intermediate Language (IL)**. All of the source code compilers create IL code – in theory the same code irrespective of the source language. This code is usually referred to as **managed code**.

So, code written in Visual Basic, Visual C++, JScript or the new language C#, will all be compiled to the same IL code. And these compilers are included as part of the .NET Framework. The Framework then looks after all aspects of executing the IL code – from compiling it to the final binary format for the respective processor, caching the IL and compiled code and then managing execution, to providing system services such as garbage collection, component activation and management, and error handling and reporting.

By combining the techniques of ASP with the .Net Framework, Microsoft has created a development system that provides a far better way of developing dynamic Web pages, along with a runtime environment that offers huge increases in both performance and scalability. This is called Active Server Pages+, or just ASP+.

How Does ASP+ Work?

The major new feature in ASP+ is the 'postback' architecture combined with the new application and page event models. ASP+ is part of the new managed code environment. This means that new techniques for reacting to events, such as a page loading or an application starting, can have code linked to them.

Often more useful, however, is the way that the postback architecture within each page works. Effectively this involves an HTML <FORM> and HTML controls elements, but executed on the

server within the ASP+ environment. This seemingly strange concept allows ASP+ to output content to the browser or client that not only creates the required HTML page, but also adds to it the extra 'stuff' required to maintain values in each of the controls and pass them back (that is, POST them back) to the server.

And, because ASP+ controls the way that the page behaves through the extra HTML, code, and attributes that it adds to the page, it can also do more fancy things. As the page is being created on the server, the ASP+ engine can detect how the form was submitted – that is, which button the user clicked and/or which control they activated.

This allows a whole new page-based event architecture to be used. Code can be added to the page, and will execute in response to specific events such as a click on a 'SUBMIT' button or the selection of a value in a list. In effect, the procedural model of earlier versions of ASP has been replaced by the event-driven programming concepts of modern GUI-based applications.

The special 'HTML elements' required as foundations for the new postback architecture are known as **ASP+ Server Controls**, and a comprehensive set of these controls is included with the standard ASP+ installation. You can also create your own controls, or inherit from and extend existing ones – using Visual Basic or any other of the .NET supported languages.

What are the ASP+ Mobile Controls?

One of the issues causing much concern to Web developers at the moment is the rapid growth of the 'mobile device' market. Literally hundreds of different types of device are appearing, none of which really match the physical or display characteristics of the ubiquitous PC-based Web browser. This range encompasses, but is not limited to, 'wireless devices'. As well as cellular phones and pocket PCs, we're seeing huge growth in things like Web TV, games consoles with Web access, and even esoteric devices like Web-enabled refrigerators and microwaves.

Many of these devices do not support HTML as a language, or the range of effects such as colors, screen size and resolution, sound, interactivity, etc. So, creating HTML as the output from your Web pages is not always the solution. Instead, we need to create a range of different types of output based on the type of client that we're responding to.

To provide this kind of feature, Microsoft has introduced a prototype set of **Mobile Controls** that can vary their output based on the requesting device type. If it's a cellular phone that expects pages coded in WML (Wireless Markup Language), then that's what the control will create. If it's Microsoft Mobile Explorer on a PocketPC that expects HTML, then the control will oblige with this format.

However, the point is that the controls are intelligent enough to be able to create output that offers the same (or as similar as possible) functionality irrespective of the device type or the output language. For example, a 'list' control should create the same list on a mobile phone as on a PocketPC or a traditional PC. The way it is actually rendered will depend on the capabilities and display characteristics of the device, but the functionality should be identical.

And, more to the point, the programmer should be able to do this without having to concern themselves about what the device is, and how it tackles the specific requirements of that device type. Instead, you just use the controls and leave it to them to figure out what to output to that client.

What do the ASP+ Mobile Controls do?

So, we know that the Mobile Controls are designed to create output that is dependent only on the device type and its capabilities and characteristics, while shielding the designed Web page or the programmer from this complexity. But what can the controls actually do, and how do you use them?

The Mobile Controls Range

The controls that are available in the current prototype release are:

- **MobilePage** – represents the 'deck' in WML terms
- **Panel** – represents a 'card' in WML terms
- **Form** – holds the user-input controls
- **Label** – creates text in the page
- **Command** – creates a link or command in the page
- **TextBox** – creates a user-input area in the page
- **List** – creates a list from which the user can select
- **RequiredFieldValidator** – checks for user input
- **RegularExpressionValidator** – validates user input
- **Stylesheet** – specifies output formatting

There is also a **SelectionList** control which is likely to be subsumed into the generic **List** control in future releases.

Setting Up the Mobile Controls

Setting up the Mobile Controls is simple, although the exact techniques are changing as the prototypes evolve. The controls are provided as a single DLL, which is copied into the **bin** directory of the application under development. This application must also be configured as a **virtual application directory** within the Internet Service Manager tool.

Secondly, the sample `config.web` file must be added to the application directory, or at least the `<browsercaps>` section of it copied into an existing `config.web` file. It's through the BrowserCapabilities control (an integral part of ASP+) that the mobile controls figure out what language and format to create for return to the client.

At present, the prototype versions of the Mobile Controls only output either WML for cellular phones or HTML for PocketPC devices and other clients. However, the range of available formats will include things like XHTML, HDML, cHTML, and others in the future.

Using the Mobile Controls

Using the Mobile Controls is just as easy as using any of the other ASP+ Server Controls – in fact it's easier because they will generally create the entire page. We don't even have to worry about the `<?xml ... ?>` and `<doctype ... >` elements as the controls will add these to the page as required.

To create a `<card>` within a WML `<deck>`, we use a Mobile **Panel** control – and within this place the other controls, text, WML or other content, as the following code demonstrates:

```
<Mobile:Panel runat="server" id="pnlMain">
  <Mobile:Form runat="server">
    <Mobile:Label runat="server">Enter your name:</Mobile:Label>
```

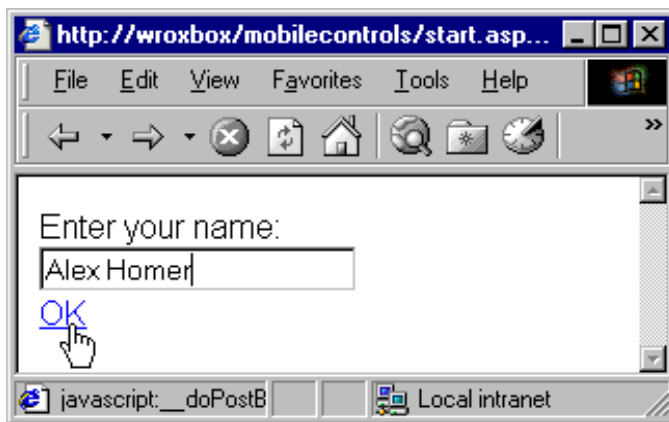
```

<Mobile:TextBox runat="server" id="NameEdit" />
<Mobile:Command runat="server" id="Button" Label="OK"
    TargetType="FormAccept" OnClick="Button_OnClick" />
</Mobile:Form>
</Mobile:Panel>

```

In the present release, every Panel control has to have a Mobile Form control that encloses the entire content. This requirement will disappear in future releases.

The result of this is a page or panel that contains a label, a text input area, and a command link. When viewed in a normal browser it looks like this:



However, when viewed in its intended target device (here demonstrated by the Nokia cellular phone emulator), it looks like this:



We're simply going to display a "Welcome" message containing the user's name, and so we add a card to the existing deck by placing another **Panel** control on the page. Within it, we use a **Label** control to display the message:

```

<Mobile:Panel runat=server id="pnlTwo">
  <Mobile:Form runat="server">
    <Mobile:Label runat="server" id="WelcomeMessage" Type="Title"/>
  </Mobile:Form>
</Mobile:Panel>

```

To display this card, we need to respond to the user selecting the **OK** option. In the Nokia phone emulator, they do this by selecting the **Options** button or by using the central "action" button (the one between the up and down buttons). Our response is achieved using some simple code within the ASP+ page, here written in Visual Basic:

```
<script language="vb" runat="server">
  Sub Button_OnClick(Sender As Object, Args As EventArgs)
    WelcomeMessage.Text = "Welcome '" & NameEdit.Text & "'"
    SetCurrentPanel (pnlTwo)
  End Sub
</script>
```

This works because the first card (or panel) contains a **Command** control that specifies the name of our Visual Basic subroutine that runs when the command is carried out:

```
<Mobile:Command runat="server" id="Button" Label="OK"
  TargetType="FormAccept" OnClick="Button_OnClick" />
```

The code in the subroutine just has to collect the user's name from the **Textbox** control in the first panel and assign the result string to the **Label** control in the second panel:

```
WelcomeMessage.Text = "Welcome '" & NameEdit.Text & "'"
```

Once this is done, it can activate the second panel by specifying the ID attribute of that panel in a call to the `SetCurrentPanel` of the page. This displays the contents of the second panel to the user:

```
SetCurrentPanel (pnlTwo)
```

The result in a normal browser looks like this:



And in the Nokia phone emulator, it looks like this:



Summary

While ASP+ is still a new and evolving technology, and in many ways a complete change from the techniques of programming in ASP 3.0 and earlier, it is already clear that the advantages it provides will make it very popular.

By building upon a universal runtime that can execute on all types of Windows machines, from Windows 2000 Server to Windows CE, Microsoft have provided a platform that makes it far easier to develop applications that have to work with various disparate types of client device.

The ASP+ Mobile Controls automatically detect the client type through entries added to the BrowserCapabilities control that is part of ASP+. Therefore, as new devices appear, the system can easily be updated to cope with them.

Building pages that work on different devices is therefore simply a matter of dropping a series of suitable ASP+ Mobile Controls onto an ASP+ page, and adding some server-side script that reacts to the events in the page (i.e. the user's actions).

While the code shown above is only a simple example, the session does demonstrate several other techniques for using the Mobile Controls, and the results with several types of client device.