Robert L. Bogue
MS MVP, MCSE, MCSA:Security
317-844-5310
Rob.Bogue@ThorProjects.com

Understanding the SharePoint, ASP.NET Relationship

Unless you've been living under a very large rock you've probably heard about this thing called Microsoft SharePoint. And like Chicken Little, people in the SharePoint community are running around saying that the end is coming.

While there's a great deal of power that SharePoint offers it doesn't mean that you won't be writing in ASP.NET (define) any longer -- quite the contrary, SharePoint is a fairly well behaved ASP.NET 2.0 application itself -- at least in the Windows SharePoint Services 3.0/Microsoft Office SharePoint Server 2007 timeframe anyway. That being said, there are some differences when it comes to developing with SharePoint that the ASP.NET developer should be aware of. Here are some of them:

Develop on a Server (in a box)

Most ASP.NET developers run Windows XP and use the built in IIS to develop their applications. That's the model that most developers are used to. However, SharePoint can't be debugged remotely so that means developing on a Windows 2003 or Windows 2003 R2 server. Of course, most people don't want to run Windows Server 2003 directly on their desktop or laptop. That means virtualizing the server with either VMWare, or Virtual PC 2007 - which is free. This in turn means more memory than most development machines have. You'll need at least a gigabyte of RAM to develop with MOSS and at least 768 megabytes to develop with WSS. Add Windows XP overhead to that and you're effectively looking at an absolute minimum of 1GB to do SharePoint development with 2GB being more realistic, and 3GB or more being ideal. Not exactly your standard configuration.

Install, Install, Install

To develop in ASP.NET you basically install Visual Studio and you're done. Everything is in there. Your operating system already has IIS and even if it doesn't you can use the built in CASINI. Unfortunately, nothing comes out of the box with SharePoint -- not even SharePoint itself. If you want to develop on WSS you'll need to install WSS. That requires at least two downloads -- .NET 3.0 and WSS.

Getting started developing gets harder after you install Visual Studio. You also need to install three additional pieces: Visual Studio Extensions for Windows Workflow Foundation (VSeWF), Visual Studio Extensions for Windows SharePoint Services (VSeWSS), and the Windows SharePoint Services Software Development Kit (WSS SDK). The VSeWF adds the workflow capabilities to Visual Studio. VSeWSS adds some templates (including web parts and list definitions) and solutions generators. The WSS SDK adds some SharePoint specific workflow templates -- in addition to the documentation.

Attach Don't Run

When you develop in ASP.NET you press F5 -- which means build and run. When you develop in SharePoint you press Ctrl-Shift-B -- which means build solution. Then you copy files, open a command prompt, GAC and un-GAC, do a voodoo dance, and then reset IIS -- at least that's what it's going to feel like. It's getting better but there are still many manual steps you have to do with each build to be able to debug.

One of the dramatic differences is that you must attach to the IIS worker process - Visual Studio 2005 won't start the IIS process and attach to it for you. None of these individual items are overwhelming individually but taken together can make for a frustrating build cycle.

It's a Design Thing

When developing a solution we're used to developing with the support of the visual design surface as in Visual Studio 2005. We get to see what we're developing visually without running the code. Whether writing an ASPX page or an ASCX control you work quicker because you can instantly see the results of your work and because you're working with a tool that supports the design experience.

With Web parts -- and almost everything else in SharePoint -- you're going to have to do it by hand, without the support of a visual design surface. That means more time to do relatively simple things. The ability to add controls to the control tree is something every ASP.NET developer should know how to do, but it isn't something that every developer should do every day.

The ASP.NET web part framework allows you to directly use User controls -- which can be designed visually -- by providing an automatic wrapper. However, SharePoint's implementation doesn't have these features so you must develop a web part -- or provide your own web part wrapper.

Newspaper or FrontPage

Ask a handful of ASP.NET developers if they've ever used FrontPage -- what is now Expressions Web -- and you're likely to hear a chuckle. Ask how many SharePoint developers have used FrontPage -- what is now SharePoint Designer (SPD) -- and you're likely to hear a bunch of sighs. That is because there are some things -- particularly with getting a design right that are much easier to do in SPD. While ASP.NET is clearly a developer platform, SharePoint is a platform for administrators, developers, and information workers. You're not limited to the familiar interface of Visual Studio, instead you've got another set of tools to learn and use if you want to be productive with SharePoint development

Pages? We Don't Need No Stinking Pages

In ASP.NET most developers are developing a page at a time. You have a page that does one thing. In SharePoint, you can't develop pages. You have to develop web parts. The web parts are assembled together, connected together, and work together to form a page. It's a model shift from what most developers are used to. Instead of thinking about how the page should look in SharePoint you think about how the page can be assembled from individual pieces. Truthfully, assembling a page from reusable user interface components is a better way to develop -- and is something to consider whether or not you decide to use SharePoint. Done correctly it can substantially cut development time and can be used to adapt to changing needs and requirements late in the process. This solves one of the biggest issues that most projects face.

Integrate don't Create

Developing for SharePoint isn't so much that you're creating your own new functionality but rather is about integrating the existing functionality into a solution that works for your users. Whether you're leveraging the built in search, using the document storage, or taking advantage of SharePoint's ability to alert users, you spend more time learning how SharePoint works and less time writing your own code to do the same thing. The benefits to the users are a much richer set of features than are practical if you're creating your own support. Take documents for example. Rarely does it make sense to provide direct integration to Word from a custom application. It's difficult to justify multiple version control strategies including major and minor versions, check in and check out support, etc. It's equally rare to allow users to specify when they receive alerts -- on which libraries and documents. The list of "nice to have" features in SharePoint are numerous and the price of admission is simply integrating to them.

What Does This Mean to Me?

The net of all of this is that SharePoint will be different than developing for ASP.NET. Which is better? Well that depends upon your point of view. If you're looking to give your users high-touch integration with office, or rich features then SharePoint might be your answer. If you're looking to minimal features and continuing to do things as they've always been done then maybe ASP.NE is still your answer.

About The Author

Robert Bogue, MCSE (NT4/W2K), MCSA:Security, A+, Network+, Server+, INet+, IT Project+, E-Biz+, CDIA+, is president of Thor Projects LLC, which provides SharePoint Consulting services to clients around the country. He has contributed to more than 100 book projects and numerous other publishing projects. His latest book is The SharePoint Shepherd's Guide for End Users. (You can find out more about the book at www.SharePointShepherd.com.)

Bogue has been part of the Microsoft Most Valuable Professional (MVP) program for the past 5 years. He was most recently awarded for Microsoft Office SharePoint Server. Before that, Bogue was a Microsoft Commerce Server MVP and Microsoft Windows Servers-Networking MVP.

Bogue runs the SharePoint Users Group of Indiana (SPIN, www.spindiana.com), and he is also a member of the steering committees for the Indiana Windows Users Group and Indianapolis .NET Developer Association. In addition to speaking at local and regional events, Bogue speaks at national conferences. He blogs at www.thorprojects.com/blog , and you can reach him at Rob.Bogue@thorprojects.com .