ASP.NET First Thoughts

Posted At : December 17, 2009 8:45 AM | Posted By : Jon Hartmann
Related Categories: Microsoft Tools

I'd announced a few weeks back that I was taking a role as a .NET desktop developer, but I've found that I'm actually going to be doing a good bit of ASP.NET web development as well (w00t!). As such, I've shifted gears a bit and dropped my studies of WPF for ASP.NET, and although I'm certainly not an expert yet, I thought I'd share my first thoughts.

Unlike ColdFusion which is a combined language of CFML tags and CFScript, ASP.NET is broken into ASP pages, and "code-behind" compiled classes created using C#, VB, or any of the other .NET languages. I'm developing ASP.NET with C#, which I find to be a very interesting language in its own right (and deserving of its own post, which will come later), so if there are differences between how things are done with a VB code-behind and a C# one, I'm not aware of them at this point. Being backed by C# and the .NET libraries are a great starting point to a web platform, so I was excited to dig in; so far I find ASP.NET to be a bit of a mixed bag.

Master Pages

One thing I'm liking so far is the concept of a "Master" page, that establishes a wrapper for site content. It functions similarly to a layout in Rails or a file that defined a header and footer and then included the target page in CFML. What I like about how ASP.NET handles this setup is that a single master page can have multiple dynamic content areas, and a single page can define the content for multiple areas in its master page. For example, a master page might look like this:

```
<%@ Master Language="VB" CodeFile="Site.master.vb" Inherits="Site" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Untitled Page</title>
<asp:ContentPlaceHolder id="head" runat="server">
</asp:ContentPlaceHolder>
</head>
<body>
<form id="form1" runat="server">
<div class="main">
<asp:ContentPlaceHolder id="MainContent" runat="server">

</asp:ContentPlaceHolder>
</div>
<div class="LeftColumnContent">
<asp:ContentPlaceHolder id="QuickLoginUI" runat="server">

</asp:ContentPlaceHolder>
<asp:ContentPlaceHolder id="LeftColumnContent" runat="server">

</asp:ContentPlaceHolder>
</div>
</form>
</body>
</html>
```

And a given page might look like this:

```
<%@ Page Language="VB" MasterPageFile="~/Site.master" AutoEventWireup="false" CodeFile="Login.aspx.vb" Inherits="Login" Title="Untitled
Page" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" Runat="Server">
<h2>
Sign In</h2>
```

```
<p>
<asp:Login ID="Login1" runat="server" TitleText="">
</asp:Login>
</p>
</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderID="QuickLoginUI" Runat="Server">
</asp:Content>

<asp:Content ID="Content4" ContentPlaceHolderID="LeftColumnContent" Runat="Server">
<h3>Sign In Tasks</h3>
<ul>
<li>Create a New Account</li>
<li>Recover Forgotten Password</li>
</ul>
<p>TODO: Turn the above text into links...</p>
</asp:Content>
```

The basic idea is that the content of each tag is plugged into the matching tag. If you wanted to understand it in ColdFusion terms, its kind of like wrapping all of your page content in tags and then including a page that output'ed the contents of each of those sections where it needed to go.

Custom Request Handlers

Another nice feature in ASP.NET is the ability to create your own HTTP request handler, but I quickly found that it wasn't as interesting as I'd first thought.. ASP.NET comes with a couple of built in handlers that handle ASP.NET pages (.aspx), Web Services (.asmx), and simple HTML page content, but you can easily write your own to handle other HTTP requests. The problem is that you still have to explicitly tell IIS to route files with certain extensions to ASP.NET, when then does a look up against a table to find the handler... I could do that in CF if I wanted to, I just don't want to have to touch IIS for for my solutions, since on hosted environments, I don't have access to that level of IIS manipulation. This seems like a feature that I'd like to see in action before I pass final judgement, but it seems limited as is.

Drag and Drop Elements

This is more of a function of Visual Studio than of ASP.NET itself, but the ability to drag-and-drop components on to a page to create a fairly rich feature set with minimal work is very attractive. Drag a data grid onto the page, click a few buttons, and the grid is set up. The code is a little more bulky than ColdFusion's >cfgrid

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="CategoryID"
DataSourceID="SqlDataSource1" ShowFooter="true" AllowPaging="True" AllowSorting="True" OnRowCommand="GridView1_RowCommand">
<Columns>

<asp:CommandField ShowDeleteButton="True" ShowEditButton="True"/>
<asp:TemplateField HeaderText="CategoryID" InsertVisible="False" SortExpression="CategoryID">
<EditItemTemplate>
<asp:Label ID="Label1" runat="server" Text='<%# Eval("CategoryID") %>'></asp:Label>
</EditItemTemplate>
<ItemTemplate>
<asp:Label ID="Label1" runat="server" Text='<%# Bind("CategoryID") %>'></asp:Label>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="CategoryName" SortExpression="CategoryName">
<EditItemTemplate>
<asp:TextBox ID="TextBox1" runat="server" Text='<%# Bind("CategoryName") %>'></asp:TextBox>
</EditItemTemplate>
<ItemTemplate>
<asp:Label ID="Label2" runat="server" Text='<%# Bind("CategoryName") %>'></asp:Label>
</ItemTemplate>
<FooterTemplate>
<asp:TextBox ID="CategoryNameTextBox" Runat="server"></asp:TextBox>
</FooterTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="Description" SortExpression="Description">
<EditItemTemplate>
```

```
<asp:TextBox ID="TextBox2" runat="server" Text='<%# Bind("Description") %>'></asp:TextBox>
</EditItemTemplate>
<ItemTemplate>
<asp:Label ID="Label3" runat="server" Text='<%# Bind("Description") %>'></asp:Label>
</ItemTemplate>
<FooterTemplate>
<asp:TextBox ID="DescriptionTextBox" Runat="server"></asp:TextBox>
</FooterTemplate>
</asp:TemplateField>
<asp:templatefield>
<footertemplate>
<asp:linkbutton id="btnNew" runat="server" commandname="New" text="New" />
</footertemplate>
</asp:templatefield>

</Columns>
</asp:GridView>
```

Change some flags and you can do in-line editing of the grid, change some more and you can have pagination, and all kinds of other features. Unfortunately there is a trade off for this kind of power, and that brings me to my last point.

Loss of Control

While ASP.NET does give me a very nice suite of components that can be used to boot-strap a project in no time, I keep finding myself wincing when I look at the source that ASP.NET spits out for these things. For that matter, I can't really stand the source that ASP.NET spits out just to function. ASP.NET tramples on many of my expectations about web design, and does so in ways that I find questionable at best. For instance, rather than having a page with multiple forms that go to different locations, ASP.NET wraps all of your content in one giant form, and then attaches event handlers to certain buttons to determine what happens when you click a button. Behind the scenes it works fine, being completely invisible to the user, but from the development side, it confuses me a bit. Also, ASP.NET makes up IDs for its elements based on their location in the document and a number of other features, so most of the concepts you have about doing Javascript programing based on ID seem to get thrown out of the window. Finally, the HTML it produces is just ugly, with lots of stuff indented funny or not at all. Its not going to impact the program, but its ugly.

All of these things are aspects of what I feel is a bigger problem, and that is loss of control. I hate it when a platform thinks it knows better than me, without giving me a clear way of getting around it. If its a client-side tag, I can virtually guarantee that my JS code is going to beat their JS code for efficiency and simplicity any day of the week. If its generating server-side code, what happens when I want to override something? I don't use the >cflayout< tags in CF8+ for the same reason; I can do it better, and I'll be able to control it if I need to change something. This is something I see as being kind of a universal problem, rather than an ASP.NET problem; as web platforms start to transition from language only to language + framework you're going to see more of this kind of thing, and I question its real value. As for ASP.NET's implementation, I'll have to wait and see. I can't really judge the framework without more experience, but considering the shop where I work has told me that they've hand to throw out a lot of ASP.NET's built in features and write there own doesn't bode well.

Conclusion

I've got some big reservations with this

~~language~~framework, but I'm still waiting to see how it shakes out. Right now I'm looking at it and not seeing anything that I couldn't do as simply or more simply with another language, but I have to believe that I'll see some nice performance gains with heavy-lifting processes since I have the benefits of a fully OO language. That said, I'm still not convinced that an OO centric way of looking at the web really jives with other web technologies, and/or that ASP.NET's view of how OO should work out is the best.

Does any one have have other perspectives about ASP.NET that they'd like to share?